1 Introduction

The advent of wireless communication and the proliferation of handheld devices has significantly advanced the growth of nomadic communications. The capability of these handheld mobile devices to self-organise themselves on-the-fly in the absence of an infrastructure, and to extend their communications beyond their wireless radio range has potentially led to the development of ***Mobile Ad hoc Networks (MANET)***. Mobile devices in these networks are commonly referred to as *nodes* and are predominantly deployed in conditions that include emergency scenarios, such as earthquakes and other natural disasters, unmanned terrain explorations and defence related applications etc. Furthermore, the self-organised, multi-hop and infrastructure-less features have evolved the MANET into being the basis for sensor networks [97, 115, 229, 259, 295, 364, 390, 409], *Vehicular Ad hoc Networks (VANET)* [325], peer-to-peer wireless networks [81, 345], pervasive networks [99] and mesh networks [6, 63, 121].

However, the successful deployment of civilian and commercial MANET is still in its infancy stages, because the same features that support the development of MANET emerge as a hindrance for their deployment. In other words, these features give rise to a range of issues, such as, (a) *broken and sporadic links that result from a mobility-induced dynamically-changing topology* [322], (b) *insecure and promiscuous wireless communications* [173], (c) *self-organised, multi-hop and infrastructure-less features of MANET being reliant on the cooperation between the mobile nodes* [168], and (d) *slow advancement in battery technology causes battery power to be a constrained resource among the heterogeneous mobile nodes* [98]. Extensive research has been carried out to date to address these issues; the nuclei of this research focus on security [121, 427, 434], *Quality of Service (QoS)* and reliability [158, 232, 264, 295, 363, 377, 392, 436, 454], mobility management and topology control [4, 336], network connectivity and routing [157, 206, 219, 223, 267, 364, 375, 410, 422], multicasting [49, 109, 117, 263, 291], power management [95, 126, 139, 197, 281, 402], and localisation and node auto-configuration [50, 198, 232, 276, 414, 415].

1.1 State of Art

Security is one of the most indispensable research areas and plays a central role in determining the success of civilian and commercial MANET [418]. Unfortunately, security solutions that have been proposed for wired networks are not directly inheritable into the MANET, because of the variant attack patterns and the new types of adversary models. In other words, mobile nodes struggle to enlist trusted intermediaries for communication with various destinations, because trusted intermediaries are a prerequisite for keeping those communications alive and free from active attacks. This has strongly influenced the development of security solutions for the routing layer so that secure end-to-end services between applications can later be realised over a trustworthy secure routing layer.

Several secure routing protocols [76, 105, 154, 155, 159, 161–164, 179, 180, 215, 260, 282, 284, 337, 367, 397, 399, 445, 446] have been proposed to assist a mobile node in discovering secure routes to various destinations. The protocols meet the objective by devising a set of rules and deploying a suitable cryptographic mechanism. The latter is used to authenticate intermediate nodes and to verify the integrity of the discovered routes. Hence, it is noted that the functionality of secure routing protocols relies heavily on the existence of a robust key management service. Key management is responsible for initialising and distributing keys (or secret associations) between nodes, and there after refreshing those keys and performing key revocations. It is a rather difficult service to attain in the MANET, at least, for the complexities involved in scaling the service to indeterminate MANET. It has been noted that such an indeterminate characteristic of the MANET results from the autonomous feature of the nodes to enter and leave the network at any time after deployment. Furthermore, the self-organised feature of the MANET contradicts the assumption of a *Centralised Authority (CA).* To add-on broken and sporadic links that are the by-products of a mobility-induced dynamically-changing topology inhibits the basic structure of the distributed online trusted authorities. Since key management and secure routing protocols are only designed to defend against predefined active attacks and also to act as a prevention system (or a first-layer of defence for the MANET), they are neither tailored to stimulate cooperation among the mobile nodes nor designed to support a mobile node in defending against either selectively misbehaving nodes or emerging attacks.

Incentive-based payment systems [77, 79, 102, 103, 151, 248, 318, 320, 349, 441, 443, 452, 453, 456] step-in to motivate cooperation among mobile nodes by, (a) *providing incentives for mobile nodes that forward the packets of other nodes*, and (b) *forcing mobile nodes to pay to forward self-generated packets.* Nevertheless, most if not all payment systems are custom-made to defend against only a special category of adversary known as *selfish nodes*[1] and, consequently, only to redress the non-cooperation that result from packet dropping. In addition, most of these payment systems are either

[1]Selfish nodes are specific to MANET owing to the heterogeneous battery resources and computational capability of the nodes. Selfish nodes with low battery power attempt to save power by not forwarding those packets generated by other mobile nodes. Alternatively, selfish nodes with high battery power tend to hijack the wireless medium for transmitting only self-generated packets.

reliant on a tamper-proof hardware or a CA to meet their objective. Finally, they also struggle to manage pricing issues and the distribution of incentives.

The success of *Intrusion Detection Systems (IDS)* as a second-layer of defence in wired networks has potentially influenced the migration or development of such systems in the MANET [11–13, 16, 73, 94, 104, 153, 195, 262, 286, 289, 290, 361, 362, 370, 371, 388, 389, 400, 403, 421]. The IDS can assist a mobile node in detecting instances of active attacks by analysing the evidence that was collected from the past actions of other mobile nodes. For instance, the simplest form of IDS in the MANET is based on promiscuous monitoring. Although IDS can collect evidence corresponding to the dynamically-changing behaviours of the other nodes, it fails to facilitate a mobile node to measure the trustworthiness of other mobile nodes and hence to react to them accordingly. In this manner, it remains analogous with secure routing, key management and payment systems, all of which have eventually led to the development of trust management systems for the MANET [39, 82, 83, 107, 112, 116, 125, 137, 140, 169, 178, 190, 204, 220–222, 235, 252, 272, 273, 301–305, 308–311, 327, 335, 347, 372, 373, 379, 380, 401, 404, 412, 425, 461, 468]. Although trust management systems lack consensus on the definition of *trust* at a more fundamental level, they adhere to most of the following characteristics to act as a detection-reaction system for the MANET, (a) *they collect evidence based on the behaviour of the other mobile nodes through an IDS and also from the opinions (which are often regarded as recommendations) received from other nodes*, (b) *they then subjectively evaluate and rate the evidence and use those ratings to predict the future behaviour of those mobile nodes* and (c) *finally they make routing decisions based on the policies defined for that corresponding routing context and the behaviour anticipated for those mobile nodes involved in that context.*

Likewise, in other security systems, solutions that adhere to the trust management systems are not free from issues. Most of these solutions have been proposed as an alternative to prevention based systems and, hence, tend to rely solely on observable and classifiable evidence that may be prone to masquerading. On the other hand, the dissemination of recommendations in these solutions opens the door to issues such as *honest-elicitation*[2], *free-riding*[3], and *recommender's bias*[4]. In most of these solutions, evidence-to-opinion mapping process struggles to solve the following, (a) *how to handle selectively-behaving benign and malicious nodes*, (b) *whether to represent the trust metric in either discrete or continuous values* and, (c) *whether to give more weightage to new evidence with respect to old evidence or vice versa.* Whenever a mobile node that deploys one of these trust management based solutions is separated from a one-hop node because of mobility and meets again with the same one-hop node after separation, the returning mobile node misses representing its ignorance on the one-hop node's change in behaviour by not revising its opinion for the one-hop node. Generally, the ignorance results from the uncertainty that the one-hop node might have either retained its old behaviour or changed its behaviour caused by compromise (or redemption) after the separation. Finally, most of these trust management-based solutions outlay only the basic policies to make trust-based decisions for routing; but fail to consider the option of outlaying comprehensive policies that can be used to enhance the security decisions of the prevention based systems.

1.2 Proposed Approach

Few integrated security systems exist for MANET with the following compositions, (a) *secure routing and trust management systems* [241, 425], (b) *secure routing and IDS* [15, 195, 246, 368, 429], (c) *key management and trust management systems* [3, 107] and, (d) *incentive-based payment and trust management systems* [461]. To the best

[2]We collectively refer false-praise and false-accusations as honest-elicitation. For example, a benign node may forward a high recommendation (false praise) in favour of an adversary so that it can prevent the adversary from reacting with a low recommendation (false accusation) in the future. Likewise, the adversary may also forward a high recommendation (false praise) for a colluding adversary or a low recommendation (false accusation) to defame a benign node.

[3]Free-riding is seen as a non-cooperation exhibited at a meta-level, where a free-riding node accepts recommendations from other nodes, but fails to reciprocate with recommendations when requested by them.

[4]In comparison with honest-elicitation, those recommendations are provided by an honest recommender but derived from an optimistic or pessimistic evaluation of collected evidence.

of our knowledge, a two-layered (first-layer of prevention systems and second-layer of detection-reaction systems) trust enhanced-security architecture is yet to be seen for the MANET. The two-layered approach is required for all prevention and detection-reaction systems to complement each other to deliver a trustworthy, secure and operational routing layer amidst those previously-mentioned inherent issues. Therefore, the objective of our research is twofold, (a) *to systematically study the security threats, attacks and adversaries in the MANET and accordingly analyse the strength of solutions that have been proposed within the different categories of security systems* and, (b) *to propose novel practical techniques to meet the shortcomings of the proposed security solutions and, thence, to develop a practical two-layered* ***Trust Enhanced security Architecture for the MANET (TEAM)***. Next, § 1.3 presents an overview of our thesis organisation.

1.3 Thesis Overview

In Chapter 2, we will present different types of security threats and attacks that are both passive and active in nature. We will then study the network layer oriented adversaries that include internal and external attackers, and compromised and selfish nodes. We will also introduce the basic security concepts to set up a platform for analysing various security systems, such as, key management, anonymous, secure routing, incentive-based payment, intrusion detection and trust management systems. In sequent, we will present our analysis and insights into the functionalities, advantages and shortcomings of these different security systems. Finally, we will establish the security requirements that are fundamental to our proposed trust enhanced security architecture.

In Chapter 3, we will introduce our obligation-based *fellowship* model to motivate cooperation among the mobile nodes. The fellowship technique obligates a mobile node to forward the packets of other mobile nodes and to not flood them with a high rate of self-generated packets to receive similar network services from those mobile nodes. We will then detail the main components of our fellowship model, which are (a) *rate-limitation*, (b) *enforcement*, and (c) *restoration*. The rate-limitation component defends against flooding attacks, while the enforcement component mitigates packet drop attacks. Fellowship enables a mobile node to determine the threshold for the rate-limitation component depending on the number of available one-hop nodes and the bandwidth of the wireless channel. Furthermore, the rate-limitation component allows a mobile node to establish threshold values with one-hop nodes without introducing additional control packets. On the other hand, the enforcement component takes into account the past behaviour of the one-hop nodes (that is, the *packet forwarding ratio (PFR)*[5]) and the availability of the channel before deeming those one-hop nodes as packet droppers. Whenever a node commits to forwarding a packet for its previous-hop, but fails to do so owing to unforeseen conditions (such as contention and congestion), then the restoration component extends the node's commitment in favour of its previous-hop's subsequent packet. All these components rely on a promiscuous monitoring-based IDS to detect the abnormalities. We will also present the operation of our fellowship technique in the presence of mobile nodes with heterogeneous battery power and its defence against *Denial of Service (DoS)* attacks. Unlike the incentive-based payment system, the fellowship technique eliminates pricing issues and does not rely on any tamper-proof hardware or a CA to motivate the selfish nodes to cooperate with other mobile nodes. Finally, we will demonstrate the performance of our fellowship technique against flooders and packet droppers, and its strengths and limitations, using large-scale Network Simulator 2 (NS2) [316] simulations. The simulation setup investigates the performance metrics such as the *packet delivery ratio (PDR)*[6] for four scenarios that vary, (a) *the percentage of malicious nodes in the network*, (b) *the maximum velocity of the mobile nodes*, (c) *the pause time between the mobility of nodes* and, (d) *the simulation area of the network to impact the density of the mobile nodes.* In our simulations, the PDR of the fellowship and the Dynamic Source Routing (DSR) [181] nodes is evaluated independently against the packet droppers for all scenarios to

[5]The ratio of the total number of packets sent to the next-hop node and the total number of packets forwarded by the next-hop node.

[6]The average ratio of the total number of packets forwarded to a destination by a source node to the total number of packets at the destination.

demonstrate the better performance of the fellowship nodes. In the case of a flooding attack, the PDR of the flooders is evaluated independently against the fellowship and DSR nodes to demonstrate the effective defence of the fellowship nodes.

In Chapter 4, we will propose our trust management-based *Secure MANET Routing with Trust Intrigue (SMRTI)* to enhance the security decisions of the DSR protocol in the MANET. The SMRTI enables a node to formulate its opinion for (or to establish trust relationship with) one-hop and multi-hop nodes by, (a) *collecting behavioural evidence from one-to-one interactions with one-hop nodes*, (b) *observing the interactions that pertain between one-hop nodes* and, (c) *receiving recommendations for multi-hop nodes.* The evidence collected from the one-to-one interactions enables the SMRTI to classify one-hop nodes as either benign or malicious. Also, the behavioural evidence collected by observing the interactions between the one-hop nodes allows the SMRTI to shortlist those malicious one-hop nodes that are likely to misbehave in future interactions. On the other hand, the SMRTI not only exploits the benefits of the recommendations by locating multi-hop benign nodes even before interacting with them, but also eliminates the issues inherent in recommendations (such as recommenders bias, honest-elicitation, free-riding and additional overhead) by deploying a novel approach to communicate the recommendations. It is important to note that the approach eliminates the need for the dissemination of additional packets or headers that are required to communicate the recommendations. We will then present how the evidence-to-opinion mapping function formulates the evidence into an opinion, where the latter is referred as *reputation* in the SMRTI. The evidence-to-opinion function also measures the relative rate of change in the behaviour of other mobile nodes so that selectively-behaving benign and malicious nodes are classified into separate categories and are therefore excluded from sensitive communications. The function prevents selectively-behaving nodes from taking a greater advantage based on their past benign behaviours and assures that mistakenly evaluated evidence does not conceal the impact of future evidence. The evidence-to-opinion function accounts for the uncertainty introduced into a node's opinion (or the trust relationship established with) about another node because of the mobility-induced dynamically-changing topology. We will also show on how the SMRTI enables a node to predict the future behaviour of other nodes by measuring their trustworthiness through different types of opinions (direct, observed, and recommended) held for them. In the SMRTI, the evidence collected from personal experiences (direct interactions and observations) takes a higher precedence over the recommendations received from others and direct interactions precede the interactions observed between neighbours. The SMRTI enhances the security of the communications by making efficient decisions based on the policies defined for routing contexts and the trustworthiness of the nodes involved in those contexts. Following are the routing contexts for which decisions are made, such as whether, (a) *to accept or reject a route from a route discovery*, (b) *to record or ignore a route from a forwarded packet*, (c) *to forward or discard a packet*, (d) *to forward a packet for a previous-hop*, (e) *to send a packet to a next-hop* and, (g) *which route to choose for the communication.*

In Chapter 5, we will present an adversary model to demonstrate the strengths and limitations of the SMRTI, and its performance using extensive NS2 simulation results. The simulation setup of the SMRTI is similar to the fellowship, except it evaluates additional performance metrics, (a) *latency*[7] and (b) the *successful route discovery (SRD)*[8] for the SMRTI and DSR nodes along with the PDR. The total number of nodes in network is fixed at 100 and, on average, 20 communications are kept alive between randomly-chosen sources and destinations. The adversary is designed to modify the routing headers such that this leads to the disruption of the route discovery and the data flow. The SMRTI demonstrates a PDR of above 50% for up to 70% of the malicious nodes.

In Chapter 6, we will present the integration of the fellowship and the SMRTI models to arrive at the detection-reaction layer of our envisaged TEAM. The integration effects two main changes in the fellowship, (a) *forwarding the evidence for packet dropping and flooding attacks to the SMRTI* and, (b) *utilising SMRTI's trust-enhanced*

[7]The average time taken by the packets to reach from source to destination.

[8]The average ratio of the total number of routes discovered to the total number of route discovery cycles initiated.

decisions to defend against malicious nodes and also to enforce selfish nodes to cooperate with other mobile nodes. Initially, the rate-limitation component of the fellowship uses the SMRTI to measure the trustworthiness of the one-hop nodes and thus customises its threshold value for each of those one-hop nodes. Similarly, the enforcement component of the fellowship considers the PFR of the next-hop nodes and also queries the SMRTI to measure their trustworthiness before forwarding packets to them. Additional evidence received from the fellowship allows the SMRTI to further deepen the separation between the selfish and malicious nodes and, in turn effectively assist the fellowship to force the selfish nodes to cooperate with other mobile nodes.

In Chapter 7, we will introduce our identity-based multi-service secure routing protocol which we refer as *Scasec* as a prevention layer for our envisaged TEAM. We will then present the operation of our TEAM and, in particular, the integration of the prevention and detection-reaction layers. Since route discovery through controlled flooding has led to the inevitability of broadcast authentication, we resort to a public-key based system and choose identity-based multi-service public key system among them to facilitate the different levels of sensitivity requirement for communications. The Scasec provides the following, (a) *it allows intermediate nodes to authenticate the source and the destination nodes*, (b) *it facilitates intermediate nodes to establish symmetric-key based secret associations with source and destination nodes* and, (c) *enables the intermediate nodes to forward security headers to the source and destination based on the established security associations.* Note that the symmetric-key based security associations reduce the overhead compared with the public-key based systems. Since self-organised mobile nodes only believe in themselves, they rely on the SMRTI to measure the trustworthiness of other mobile nodes and to decide whether or not to establish secret associations with those mobile nodes. Furthermore, the notion of using an identity-based multi-service public-key system reflects the notion of different levels of trustworthiness in the communications. Alternatively, the Scasec forwards the evidence of the fabrication and modification attacks to the SMRTI and then uses the SMRTI to choose trustworthy routes for secure communications.

In Chapter 8, we will analyse the flooding and packet drop attacks in the MANET that support anonymous communications. We will then extend our fellowship technique to defend against flooding attacks in anonymous MANET. The anonymous routing protocol chosen for our extension is the *Anonymous Secure Routing (ASR)* [460]. Simplicity, light-weight, a higher degree of identity anonymity, location privacy, route anonymity and the assured security of established routes are the factors that led to the choice of the ASR. Our approach efficiently identifies and isolates a malicious node that floods the network. Also, it considers the benign behaviour of an expelled node and rejoins that node into the network. Because our adapted fellowship-based technique does not require any additional packets to communicate the behaviour of the flooders, it does not incur any additional overhead. Finally, we will validate the performance of our adapted technique through detailed NS2 simulations. In the first scenario, the performance of the fellowship-extended and ASR nodes is analysed against the persistent flooders by varying the threshold limit. The second scenario imitates a similar setup, except that the flooders are allowed to exhibit a selective flooding behaviour.

In Chapter 9, we will incorporate subjective logic into the SMRTI to resolve the notion of uncertainty in the trust relationships established between newly-joining and existing mobile nodes. In the MANET, the existing nodes may not have a record of past evidence to trust or distrust a newly joining node. By pessimistically assigning a low level of trust or by optimistically assigning a neutral or high level of trust to a new node poses several issues. The purpose of the pessimistic approach is to compel a newly-joining node to exhibit a consistent benign behaviour from the moment it enters the network. However, in this approach, it is not always clear how the less trusted newly-joining node is selected for communications when nodes with high trust values already exist in the network. If a new node is not preferred for communications because of its low level of trust, it then lacks the opportunity to gain the trustworthiness of the existing nodes. Alternatively, with the optimistic approach, the aim is to promptly identify whether a newly-joining node exhibits a malicious behaviour from the moment it enters into the network. Prompt identification is feasible, because the neutral or high level of trust assigned for a newly-joining node decreases rapidly once the malicious behaviour increases. However, the optimistic approach fails to discriminate a newly-joining node from the existing nodes, whose dynamically-changing selective behaviour has warranted the same level of trust. Because the aspect of ignorance and the associated uncertainty are intrinsic to the establishment of trust relationships in the MANET, the failure to represent such an uncertainty in a trust relationship between two mobile nodes may not always reflect their actual relationship and, consequently, the executed decision may not always be accurate. We will then present the sound mathematical foundation of subjective logic to explicitly represent and manage a node's ignorance in its trust relationships with other nodes. We will also show our improvements to the subjective-logic to adapt it to the MANET. Finally, we will present our adapted version of subjective-logic based SMRTI to the *Ad hoc On-demand Distance Vector (AODV)* [296] and demonstrate its performance against the modifications, packet dropping and flooding attacks, using NS2 simulations.

In Chapter 10, we will summarise our thesis by highlighting the design and operations of our models and, their extensions. We will then outline our future work mainly from the perspective of simulations and the reasons are, (a) *to further understand the impact of the parameters of the models, their extensions and the network*, (b) *to evaluate additional performance metrics*, (c) *to compare the performance results with the related models.* We will then present our plan to explore the related security areas in the MANET. We expect to extend the secure routing protocol's capability to deliver secure data transfer and also to further confirm its operations through simulation results. Similar to the fellowship's extension to the MANETs that support anonymous communications, we will be examining the possibility of extending the SMRTI and the TEAM to the anonymous communications. In sequent, we will also discuss the possibility of developing a policy manager for the purpose of coupling our architecture to the variety of applications at the top layer. Furthermore, we will present our plan to couple a tamper-proof hardware as the bottom layer to defend against strong-identity related issues.

2 Related Work

2.1 Introduction

In this chapter, we first describe the security issues that are prevalent in the MANET and the security systems that have been proposed to mitigate those security issues, and finally their limitations. Although the features such as, (a) *wireless communication*, (b) *mobility*, (c) *lack of infrastructure* and (d) *absence of power cables*, serve as the substratum for the development of MANET, they ironically open door to several security issues. As explained below, these issues have severe impact on the design of security systems for the MANET and, hence, prevent them from inheriting the security solutions that have been proposed for wired networks.

To begin with, wireless networks that are well-known for eliminating wired cables are vulnerable to eavesdropping, whereas wired networks necessitates an adversary to gain physical access to the cables to achieve the same. This confirms the absence of clear line of defence in the MANET as in the case of wired networks. Furthermore, the adversaries would be able to take advantage of the promiscuous wireless medium to inject modified or spoofed packets. All these demands for an efficient security system at every mobile node so that those mobile nodes can defend against such threats.

Second, mobility is the factor that facilitates nodes to locate potential network services. Nodal mobility when combined with the wireless medium allows nodes to enter and leave the network at their will. Ironically, all these lead to a dynamically changing topology in the MANET that is characterised by lack of shape and size. Although mobility has been shown to increase the capacity of MANET [114], it is irrefutable that mobility-induced dynamically changing topology results in broken links and sporadic connections. This in turn opens door for adversaries to report valid links as broken links for the sake of disrupting genuine communications. Also, the lightweight and portable characteristics of mobile nodes make them vulnerable to capture and compromise.

Since the self-organised, infrastructure-less and multi-hop communication translates into the elimination of centralised servers and absence of dedicated devices for routing and network management, all mobile nodes are required to co-operate with each other for the correct execution of routing and network management. Interestingly, this becomes a pre-requisite for the establishment and sustainability of the MANET. In turn, such intrinsic requirement manifests as a vulnerability, and accordingly, several simulation results [210, 254, 357, 449] have demonstrated on how adversaries would be able to degrade the network performance dramatically by exploiting the vulnerability. Notably, any security system that is designed to mitigate such exploitation or in other words designed to motivate cooperation among mobile nodes has to consider the fact that a centralised CA is infeasible in the self-organised and infrastructure-less MANET.

Finally, battery-powered mobile are portrayed to eliminate the usage of power cables in order to complement the nomadic communication. However, the heterogeneity that prevails in the MANET inspires adversaries to maximise their battery or other resources by either not forwarding packets on the behalf of other nodes or only transmitting self-generated packets. Such a behaviour has a profound effect in the network performance, and at extreme conditions, it may also lead to a partitioned or an inoperative MANET. In synopsis, the heterogeneity in the battery, storage, and computational resources of the MANET challenge the design of any security system for the MANET.

To further understand the impact of these inherent features on the design of security systems, we briefly summarise different types of attacks that are prevalent in the context of MANET and then some fundamental security concepts for the design of security system in the following §2.2. Since exhaustive research has been carried out in the security of MANET, we concentrate only on few well-reviewed approaches. Initially, we focus our study on prevention-based approaches, then detection-based approaches and finally reaction-based approaches. Although the main objective of our thesis is to develop a two-layered security architecture that can defend against prevailing and emerging active attacks, we discuss few well-reviewed approaches that enable the MANET to defend against passive attacks and, thereof to support anonymous communications in §2.3. In §2.4, we detail some well-known secure routing protocols to iterate their role in preventing multi-hop communications against conventional attacks. We then discuss the significance of key management systems in complementing the functionality of secure routing protocols in §2.5. Section 2.6 will introduce incentive and game-theory based systems that focus on stimulating cooperation among nodes. We will then conclude the section with a discussion on their limitations and how they fail to complement the prevention-based systems. In §2.7, we will discuss the recent trend of adopting defence-in-depth approach, such as, including the IDS in the security design of MANET. In sequent, we will introduce the reputation-based systems and then the trust management systems that are developed to address the shortcomings of prevention-based systems. As a part of the literature review, we will introduce the concepts and methodology confined to the traditional trust management systems and also the limitations of trust management systems. Section 2.8 will summarise our literature study in terms of security requirements that will be fulfilled in the design of our two-layered security system for the MANET.

2.2 Attacks and Adversaries

In general, attacks can be broadly classified into *internal* and *external* based on their origin. The external attacks are launched by nodes that are not part of the network, while internal attacks are originated by malicious or compromised nodes that are part of the network. In reality, it is hard to defend against internal attacks when compared with the external attacks. Orthogonally, attacks can also be classified as either *passive* or *active* depending on their nature. Although we concentrate only on the security systems that have been proposed to defend against the routing-layer specific active attacks, we briefly discuss the scope of passive attacks in the following, and then the security solutions that have been proposed to defend against passive attacks in § 2.3 for the sake of completeness.

2.2.1 Passive Attacks

In [201], Kong *et al.* demonstrated how external and internal adversaries can conduct passive attacks in the MANET. In particular, they showed how external adversaries are capable of launching passive attacks even in the presence of an end-to-end encryption because of the routing headers being in plaintext, at least for the purpose of routing. In addition, they described how external adversaries are also capable of performing flow-based timing analysis for extracting information when group encryption is employed. However, they pointed out that such restrictions fail to hold against the internal attackers and only levy additional overhead to the network. For instance, internal attackers can passively eavesdrop to trace the motion pattern and location of monitored nodes. In addition, they can also analyse multiple routes and therefore, partially visualise the dynamically changing topology of the network. Furthermore by colluding with other adversaries, such internal attackers can effectively understand, (a) *where the monitored mobile nodes are heading to and at what rate those monitored nodes are moving*, (b) *how many mobile nodes are relatively stationary to their location and the corresponding list of their neighbours*, (c) *the rate at which those monitored nodes communicate with the destination and especially through their neighbours*, (d) *to which destinations* *do those monitored nodes communicate*, (e) *how far those destinations and monitored nodes are positioned from each other in terms of hop-count*, and (f) *the type of communication between those monitored nodes and destinations.* This situation aggravates if the internal adversaries can integrate geographical information into their analyses and note that such analyses are critical in the MANET that support defence-related applications.

2.2.2 Active Attacks

In the MANET, active attacks are highly prominent in the MAC [84, 135, 143, 145, 207, 208, 319, 331] and routing layers [1, 7, 274, 277, 307, 406, 419]. Given that our objective is to secure the routing layer, we classify routing layer specific active attacks into three broad categories, (a) *modification*, (b) *fabrication* and (c) *interruption of routing packets.*

- **Modification Attack**: In this case, the adversary alters the contents of routing messages and injects them back into the network to disrupt either the routing or data flow phase (for example, injecting a route request packet with maliciously increased sequence number).
- **Fabrication Attack**: Here the adversary generates routing messages with the identity of other nodes and injects them into the network to disrupt the routing phase (for example, injecting spoofed packets).
- **Interruption Attack**: In this case, the adversary interrupts the routing or network management by not performing the basic routing operations. (for example, packet dropping).

Depending on the objective of an adversary, active attacks can be also alternatively classified as given below, and such a classification usually blends modification, fabrication or interruption attacks. In our thesis, we interchangeably refer to mobile nodes that perform active attacks as *malicious nodes* and collectively active attacks as *misbehaviours*.

- In a *routing loop attack*, an adversary modifies the route discovery packets so that it can cause the intermediate nodes to dissipate their valuable battery resources by making them to forward those modified packets along a loop.
- Adversaries may also alter routing packets to attract the packets of other nodes towards them and then to drop those attracted packets. Otherwise, they may selectively drop those attracted packets to conceal themselves from being identified. The former is known as *blackhole attack*, while the later is known as *grayhole attack*.
- *Spoofing attack* is one of the most vulnerable attacks that permits a node to become an authorised entity in the network and allows it to take advantage of the authorised services.
- *Detour attack* routes packets along a sub-optimal path to ensure that one set of nodes are never reachable. In *gratuitous attack* (*i.e.* variation of the detour attack), the adversary modifies the routing header to lengthen the route so that the lengthened route is not chosen for the data flow. Such attacks allow the adversary to flee from forwarding packets for other nodes.
- In hop-count based on-demand routing protocol such as AODV, the adversary can suppress the propagation of valid route requests by quickly propagating a route request that contains the bypassed hop-count increment. Since the adversary rushes the route request ahead of valid route requests by not incrementing the hop-count, it is known as *rushing attack*.
- In the absence of authentication, the adversary can corrupt another node's legitimate information through *blackmail attack*.
- *Wormhole attacks* are *tunnelling attacks*, in which the adversary tunnels routing packets to another adversary that is positioned at the other end of the network. The latter disseminates the received routing packet to corrupt the routing table of its neighbours and hence prevents them from discovering valid routes.

- *Route salvaging attack* allows the adversary to invoke falsified route salvations so that they can disrupt the network functioning.
- Similarly, *falsified route error generation attack* causes the source node to re-initiate the route discovery by generating route errors that falsely report valid links as broken links. Alternatively, *valid route error suppression attack* is a variation of the above, where valid route errors are suppressed from reaching the source node when the link is actually broken.
- *Flooding attacks* cause intermediate nodes to burn their battery resources by driving them to forward flooded packets. In some cases, it also blows out the routing table as a result of overflow.
- *Packet dropping* is similar to blackhole and grayhole attacks since the adversary drops the received packets, except that the adversary fails to modify the routing headers to attract those packets towards it.

Apart from malicious nodes, there is another type of adversary in the MANET that is known as *selfish nodes*. These nodes are the resultant of the heterogeneity in battery, computational, and storage resources in the MANET. Selfish nodes that are powered with low-battery resource attempt to save their resources by not forwarding packets on the behalf of other nodes (*i.e.* packet dropping). Similarly, selfish nodes that are powered with high-battery resource aim to hijack the wireless medium by boycotting the MAC layer's contention resolution mechanism and then to maximise the network utility by propagating only self-generated packets. Note that although selfish nodes differ from malicious nodes in terms of their intention, the effect of their intentions is synonymous with the malicious nodes.

2.2.3 Security Concepts

Similar to traditional wired networks, the security services that are envisaged to defend against adversaries in the MANET are, (a) *confidentiality*, (b) *integrity*, (c) *authentication*, (d) *non-repudiation*, (e) *availability*, and (f) *authorisation*.

- **Confidentiality** ensures the privacy of information such that the information is only delivered to the intended recipient. This is generally achieved by employing symmetric and asymmetric-key based cryptographic mechanism.
- **Integrity** maintains the information intact and assures the recipient that the information is free from any modification. Although symmetric and asymmetric-key based cryptographic mechanisms can be employed, one-way hash based cryptographic mechanism is preferred over other mechanisms.
- **Authentication** enables a node to verify the identity of the claimant and hence prevents an adversary from masquerading as a trusted mobile node.
- **Non-repudiation** ensures that a mobile node cannot deny the sent message in future. Public-key based certificates are traditionally employed to render non-repudiation service.
- **Authorisation** policies at the routing layer is not as comprehensive as the application layer, for instance, it assists a source node to shortlist intermediate nodes according to the sensitivity and type of communications. Trust management systems incorporate such policies to enhance the routing decisions.
- **Availability** assures that network services are available to the mobile nodes in the presence faulty mobile nodes, issues inherent to the MANET or adversaries. Related proposals measure the trustworthiness of nodes based on their behavioural pattern and, hence, efficiently route around the nodes that disrupt the availability of network service.

2.3 Anonymous Secure Routing in MANET

Although passive attacks are traditionally mitigated by anonymous routing, any security design that incorporates anonymous routing is forced to rely on the following, (a) *well-reviewed cryptographic mechanism*, (b) *robust and scalable key management* *mechanism*, and (c) *one-time valid routing information to defend against traffic analysis*. In general, most of the solutions that have been proposed for the MANET to deploy anonymous routing are inspired from the Chaum's *mix-net* technique [92, 93] or public-key based protocols. In the mix-net, the traffic from a source to a destination is passed through one or more *mixes*. Unlike traditional routers, a mix reorders and re-encrypts the incoming packets to separate the relation between the incoming and outgoing packets. In [176, 177], Jiang *et al.* have extended the concept of mixes to the MANET, while *Anonymous On Demand Routing (ANODR)* protocol [199, 200] assigned pseudonym using mixes to conceal the identities of communicating nodes. In [352], Shokri *et al.* established a framework such that nodes in a path are virtually bounded to their immediate neighbours like a chain. Hence they are not required to know any other information about source, destination and other parts of the chain. Later, they deployed an on-demand routing protocol known as *Chain-based Anonymous Routing (CAR)* over the established framework to discover untraceable routes. However, the computational overhead levied by CAR makes it impractical for the MANET. *Anonymous Secure Routing (ASR)* protocol [460] is similar to DSR, except that ASR neither contains the source nor destination addresses. In addition, the intermediary nodes do not append their identities to the messages; however they retain the state information of those messages for later response. Pairing-based key agreements [36] have been employed in [450] to propose an anonymous authentication protocol for neighbours so that neighbours can authenticate each other without revealing their identities. The secret keys and pseudonyms that are established during the *neighbourhood authentication protocol* are then used to perform anonymous on-demand routing known as *MASK*. In [348], Seys and Preneel have proposed *Anonymous Routing Protocol for MANET (ARM)* that provides higher efficiency than the ANODR and the ASR. Also, the ARM addresses the drawback of the MASK (*i.e.* the tight synchronisation of keys and pseudonyms between neighbouring nodes) using random padding and time-to-live values. Other noteworthy research works in the area of anonymous communications in the MANET can be referred at [9, 59–61, 374].

2.4 Secure Routing in MANET

Given that routing protocols [2, 42, 44, 52, 87, 132, 219, 244, 258, 267, 333, 345, 422] do not, (a) *authenticate intermediate nodes*, (b) *verify routing messages*, and (c) *protect data transmissions*, the routing layer opens door to wide range of passive and active attacks. Secure routing protocols [21, 75, 76, 105, 120, 136, 150, 154, 155, 159, 161–164, 179, 180, 215, 250, 260, 282, 284, 337, 367, 397, 399, 439, 440, 445, 446, 455] step-in to defend against active attacks that are related to modification and fabrication of routing messages. They heavily rely on a robust key management system to utilise the secret associations established between the mobile nodes. They define appropriate steps and employ suitable cryptography mechanism, and then utilise the secret associations to authenticate the intermediate nodes or (and) verify the integrity of routing messages. As a result, they are able to assist mobile nodes in discovering secure paths, and then data transmissions are protected by employing end-to-end encryption to the payload using the secret associations [57, 236, 283, 285]. Several research papers [19, 78, 146, 160, 174, 192, 193, 256, 265, 321, 384, 442] from the literature discuss the strengths and limitations of these approaches and, we detail a few well-reviewed secure routing protocols in the following.

2.4.1 Authenticated Routing for Ad-hoc Networks (ARAN)

In *ARAN* [337], certificates are used to bind a node's IP address to its public-key. The source node provides authenticity and freshness to the route request by signing the, (a) *IP address of the destination*, (b) *nonce*, and (c) *timestamp*. At every hop, the intermediate nodes wrap their signature over the signature provided by the source node before broadcasting the route request. Hence, the intermediate nodes accept the route request only after verifying their previous-hop's signature, and then verifying the authenticity and freshness of the route request using the signature provided by the source node. For example in Table 2.1, node $\mathcal{C}$ broadcasts the route request received from its previous-hop $\mathcal{B}$ only after verifying the authenticity of $\mathcal{B}$, and then the authenticity and freshness of the route request using the signature provided by $\mathcal{S}$. Note that the intermediate node next to the source node verifies only the authenticity and freshness of the route request.

Route Request Phase	
$\mathcal{S} \rightarrow * :$	$\left\{ \left(RREQ, \mathcal{D}, \text{Cert}_{\mathcal{S}}, \text{Nonce}, \text{Timestamp} \right)_{K_{\mathcal{S}}} \right\}$
$\mathcal{A} \rightarrow * :$	$\left\{ \left[\left(RREQ, \mathcal{D}, \text{Cert}_{\mathcal{S}}, \text{Nonce}, \text{Timestamp} \right)_{K_{\mathcal{S}}} \right]_{K_{\mathcal{A}}}, \text{Cert}_{\mathcal{A}} \right\}$
$\mathcal{B} \rightarrow * :$	$\left\{ \left[\left(RREQ, \mathcal{D}, \text{Cert}_{\mathcal{S}}, \text{Nonce}, \text{Timestamp} \right)_{K_{\mathcal{S}}} \right]_{K_{\mathcal{B}}}, \text{Cert}_{\mathcal{B}} \right\}$
$\mathcal{C} \rightarrow * :$	$\left\{ \left[\left(RREQ, \mathcal{D}, \text{Cert}_{\mathcal{S}}, \text{Nonce}, \text{Timestamp} \right)_{K_{\mathcal{S}}} \right]_{K_{\mathcal{C}}}, \text{Cert}_{\mathcal{C}} \right\}$
Route Reply Phase	
$\mathcal{D} \rightarrow \mathcal{C} :$	$\left\{ \left(RREP, \mathcal{S}, \text{Cert}_{\mathcal{D}}, \text{Nonce}, \text{Timestamp} \right)_{K_{\mathcal{D}}} \right\}$
$\mathcal{C} \rightarrow \mathcal{B} :$	$\left\{ \left[\left(RREP, \mathcal{S}, \text{Cert}_{\mathcal{D}}, \text{Nonce}, \text{Timestamp} \right)_{K_{\mathcal{D}}} \right]_{K_{\mathcal{C}}}, \text{Cert}_{\mathcal{C}} \right\}$
$\mathcal{B} \rightarrow \mathcal{A} :$	$\left\{ \left[\left(RREP, \mathcal{S}, \text{Cert}_{\mathcal{D}}, \text{Nonce}, \text{Timestamp} \right)_{K_{\mathcal{D}}} \right]_{K_{\mathcal{B}}}, \text{Cert}_{\mathcal{B}} \right\}$
$\mathcal{A} \rightarrow \mathcal{S} :$	$\left\{ \left[\left(RREP, \mathcal{S}, \text{Cert}_{\mathcal{D}}, \text{Nonce}, \text{Timestamp} \right)_{K_{\mathcal{D}}} \right]_{K_{\mathcal{A}}}, \text{Cert}_{\mathcal{A}} \right\}$

Table 2.1: Secure Route Discovery using ARAN.

The same procedure is carried-out in reverse for the route reply, except that the route request is a broadcast and the route reply is a unicast. The main drawback in ARAN is that fails to address how a CA is deployed in a self-organised MANET and therefore exposed to the corresponding issues, such as, (a) *issuing new certificates*, (b) *refreshing* and (c) *revoking certificates.*

2.4.2 Secure Ad-hoc On-demand Distance Vector (SAODV)

SAODV [445, 446] achieves the following such as, authentication, integrity, confidentiality and non-repudiation using the public-key based certificates. Hence, the source and destination are able to verify and authenticate each other by signing the mutable fields of the route request and route reply respectively. The signature also contains the seed of a one-way hash chain to prevent modification of the hop-count. At every hop, the intermediate nodes increase the hop-count by hashing the previous hash chain value. Therefore, the one-way nature of the hash chain prevents the intermediate nodes from reducing the hop-count.

For instance, if the hash chain values $h_0, h_1, \cdots, h_i, \cdots, h_n$ are generated, such that, $h_i = H[h_{i+1}]$, then the hop-count authenticator h_i corresponds to a hop-count of $n - i$. This can be verified using the hash chain anchor h_0 and length n from the signature. Let us consider the route request phase in Table 2.2, where node $\mathcal{B}$ verifies the hop-count using the hash chain authenticator h_{n-2}, anchor h_0 and length n, because $h_0 = H_{n-2}[h_{n-2}]$.

Route Request Phase	
$\mathcal{S} \rightarrow * :$	$\left\{\left(\mathit{RREQ}, \text{RouteID}, \mathcal{S}, \text{Seq}_{\mathcal{S}}, \mathcal{D}, \text{OldSeq}_{\mathcal{D}}, h_0, n\right)_{K_{\mathcal{S}}}, 0, h_n\right\}$
$\mathcal{A} \rightarrow * :$	$\left\{\left(\mathit{RREQ}, \text{RouteID}, \mathcal{S}, \text{Seq}_{\mathcal{S}}, \mathcal{D}, \text{OldSeq}_{\mathcal{D}}, h_0, n\right)_{K_{\mathcal{S}}}, 1, h_{n-1}\right\}$
$\mathcal{B} \rightarrow * :$	$\left\{\left(\mathit{RREQ}, \text{RouteID}, \mathcal{S}, \text{Seq}_{\mathcal{S}}, \mathcal{D}, \text{OldSeq}_{\mathcal{D}}, h_0, n\right)_{K_{\mathcal{S}}}, 2, h_{n-2}\right\}$
$\mathcal{C} \rightarrow * :$	$\left\{\left(\mathit{RREQ}, \text{RouteID}, \mathcal{S}, \text{Seq}_{\mathcal{S}}, \mathcal{D}, \text{OldSeq}_{\mathcal{D}}, h_0, n\right)_{K_{\mathcal{S}}}, 3, h_{n-3}\right\}$
Route Reply Phase	
$\mathcal{D} \rightarrow \mathcal{C} :$	$\left\{\left(\mathit{RREP}, \mathcal{D}, \text{Seq}_{\mathcal{D}}, \mathcal{D}, \text{lifetime}, \mathcal{S}, h_0, n\right)_{K_{\mathcal{D}}}, 0, h_n\right\}$
$\mathcal{C} \rightarrow \mathcal{B} :$	$\left\{\left(\mathit{RREP}, \mathcal{D}, \text{Seq}_{\mathcal{D}}, \mathcal{D}, \text{lifetime}, \mathcal{S}, h_0, n\right)_{K_{\mathcal{D}}}, 1, h_{n-1}\right\}$
$\mathcal{B} \rightarrow \mathcal{A} :$	$\left\{\left(\mathit{RREP}, \mathcal{D}, \text{Seq}_{\mathcal{D}}, \mathcal{D}, \text{lifetime}, \mathcal{S}, h_0, n\right)_{K_{\mathcal{D}}}, 2, h_{n-2}\right\}$
$\mathcal{A} \rightarrow \mathcal{S} :$	$\left\{\left(\mathit{RREP}, \mathcal{D}, \text{Seq}_{\mathcal{D}}, \mathcal{D}, \text{lifetime}, \mathcal{S}, h_0, n\right)_{K_{\mathcal{D}}}, 3, h_{n-3}\right\}$

TABLE 2.2: Secure Route Discovery using SAODV.

SAODV struggles to address the issues related to public-key based certificates and is prone to *same distance fraud*, where the forwarding node fails to increment the hop-count to make the route appear shorter in length. Also, it is vulnerable to malicious increment of hop-count, which may be carried out by selfish nodes to discourage the route from being chosen for the data flow.

2.4.3 Ariadne

Ariadne [164] secures DSR by employing a broadcast-based authentication mechanism known as *TESLA* [297–299], which relies on loosely synchronised clocks and delayed key disclosures. Note that TESLA allows intermediate nodes to authenticate other intermediate nodes that are enlisted in the route. Also, Ariadne calculates message authentication code ($HMAC$) against the, (a) *identities of source and destination*, and (b) *sequence number* using the secret key shared between the source and destination and, this prevents the modification of routing messages.

Route Request Phase	
$\mathcal{S}:$	$h_0 = HMAC_{K_{\mathcal{SD}}}(RREQ, \mathcal{S}, \mathcal{D}, id, \text{time})$
$\mathcal{S} \rightarrow *:$	$\{RREQ, \mathcal{S}, \mathcal{D}, id, \text{time}, h_0, (), ()\}$
$\mathcal{A}:$	$h_1 = H[\mathcal{A}, h_0];\quad M_{\mathcal{A}} = HMAC_{K_{\mathcal{A}_{time}}}(RREQ, \mathcal{S}, \mathcal{D}, id, \text{time}, h_1, (\mathcal{A}), ())$
$\mathcal{A} \rightarrow *:$	$\Big\{RREQ, \mathcal{S}, \mathcal{D}, id, \text{time}, h_1, (\mathcal{A}), \big(M_{\mathcal{A}}\big)\Big\}$
$\mathcal{B}:$	$h_2 = H[\mathcal{B}, h_1];\quad M_{\mathcal{B}} = HMAC_{K_{\mathcal{B}_{time}}}\Big(RREQ, \mathcal{S}, \mathcal{D}, id, \text{time}, h_2, (\mathcal{A}, \mathcal{B}), \big(M_{\mathcal{A}}\big)\Big)$
$\mathcal{B} \rightarrow *:$	$\Big\{RREQ, \mathcal{S}, \mathcal{D}, id, \text{time}, h_2, (\mathcal{A}, \mathcal{B}), \big(M_{\mathcal{A}}, M_{\mathcal{B}}\big)\Big\}$
$\mathcal{C}:$	$h_3 = H[\mathcal{C}, h_2];\quad M_{\mathcal{C}} = HMAC_{K_{\mathcal{C}_{time}}}\Big(RREQ, \mathcal{S}, \mathcal{D}, id, \text{time}, h_3, (\mathcal{A}, \mathcal{B}, \mathcal{C}), \big(M_{\mathcal{A}}, M_{\mathcal{B}}\big)\Big)$
$\mathcal{C} \rightarrow *:$	$\Big\{RREQ, \mathcal{S}, \mathcal{D}, id, \text{time}, h_3, (\mathcal{A}, \mathcal{B}, \mathcal{C}), \big(M_{\mathcal{A}}, M_{\mathcal{B}}, M_{\mathcal{C}}\big)\Big\}$
Route Reply Phase	
$\mathcal{D}:$	$M_{\mathcal{D}} = HMAC_{K_{\mathcal{SD}}}\Big(RREP, \mathcal{D}, \mathcal{S}, \text{time}, (\mathcal{A}, \mathcal{B}, \mathcal{C}), \big(M_{\mathcal{A}}, M_{\mathcal{B}}, M_{\mathcal{C}}\big)\Big)$
$\mathcal{D} \rightarrow \mathcal{C}:$	$\Big\{RREP, \mathcal{D}, \mathcal{S}, \text{time}, (\mathcal{A}, \mathcal{B}, \mathcal{C}), \big(M_{\mathcal{A}}, M_{\mathcal{B}}, M_{\mathcal{C}}\big), M_{\mathcal{D}}, ()\Big\}$
$\mathcal{C} \rightarrow \mathcal{B}:$	$\Big\{RREP, \mathcal{D}, \mathcal{S}, \text{time}, (\mathcal{A}, \mathcal{B}, \mathcal{C}), \big(M_{\mathcal{A}}, M_{\mathcal{B}}, M_{\mathcal{C}}\big), M_{\mathcal{D}}, \big(K_{\mathcal{C}_{time}}\big)\Big\}$
$\mathcal{B} \rightarrow \mathcal{A}:$	$\Big\{RREP, \mathcal{D}, \mathcal{S}, \text{time}, (\mathcal{A}, \mathcal{B}, \mathcal{C}), \big(M_{\mathcal{A}}, M_{\mathcal{B}}, M_{\mathcal{C}}\big), M_{\mathcal{D}}, \big(K_{\mathcal{C}_{time}}, K_{\mathcal{B}_{time}}\big)\Big\}$
$\mathcal{A} \rightarrow \mathcal{S}:$	$\Big\{RREP, \mathcal{D}, \mathcal{S}, \text{time}, (\mathcal{A}, \mathcal{B}, \mathcal{C}), \big(M_{\mathcal{A}}, M_{\mathcal{B}}, M_{\mathcal{C}}\big), M_{\mathcal{D}}, \big(K_{\mathcal{C}_{time}}, K_{\mathcal{B}_{time}}, K_{\mathcal{A}_{time}}\big)\Big\}$

Table 2.3: Secure Route Discovery using Ariadne.

Route requests are broadcasted by the source with an expected time of arrival to the destination. In sequent, the intermediate nodes authenticate other nodes that are enlisted in the route by using the valid TESLA key of those nodes. Intermediate nodes drop a route request only if they have either seen the route request already or failed to confirm the validity of a TESLA key. Otherwise, they append themselves to the route and add the HMAC value of the header to the HMAC list of the route request. In addition, they also replace the previous hash value with the hash value generated using their identity and previous hash value (*i.e.* $h_i = H[Node_{id}, h_{i-1}]$). The hash value and HMAC list prevent intermediate nodes from altering the route. As shown in Table 2.3, destination $\mathcal{D}$ accepts the route request only after verifying the hash value $h_i = H\Big[Node_{id}, H\Big[Node_{id-1}, H\big[Node_{id-2}, \cdots, H[h_0]\big]\Big]\Big]$, where h_0, is the HMAC generated by the source. The authors call this process as *per-hop hashing* since no single intermediate node is omitted from the route request. Optionally, destination may also wait for the intermediate nodes to release their TESLA keys before responding back with the route reply. Note that the route reply carrying the HMAC list is unicasted along the reverse path to the source.

Authors of Ariadne claim that TESLA remains secure irrespective of the end-to-end delay. However, the delay parameter and the assumption in TESLA that a node should be able to determine which keys a sender might have already published impact the network performance. Furthermore, it has been shown that solutions based on loosely synchronised clock that require re-synchronisation to avoid the clock drift are prone to a special type of DoS attack known as *desynchronisation attack* [417]. The authors have also proposed a solution known as *packet leashes* [162] to defend against wormhole attack by using similar technique.

2.4.4 Secure Routing Protocol (SRP)

SRP [282] delivers a secure path from source to destination based on the assumption that source and destination have a secure association prior to the deployment. Unlike other secure routing protocols, SRP does not levy heavy computations at the intermediate nodes. As shown in Table 2.4, source maintains a pair of identifier for each route request, (a) *query sequence number* and (b) *random query identifier.* The HMAC computed against the, (a) *source*, (b) *destination* and (c) *two query identifiers* using the secure association is utilised to verify the integrity and authenticity of the route request. The query sequence number ensures the freshness of the route request and the random query identifier assists the intermediate nodes to identify queries that have been already forwarded. If the route request is valid, the destination then responds back with a route reply in the same manner. The source node discards the route replies that fail to match with the pending query identifiers and, verifies the integrity of the route reply using the HMAC generated by the destination.

Route Request Phase	
$\mathcal{S}:$	$M_{\mathcal{S}} = HMAC_{K_{\mathcal{SD}}}\big(RREQ, Q_{\mathcal{D}_{Seq}}, Q_{ID}, \mathcal{S}, \mathcal{D}, ()\big\}$
$\mathcal{S} \rightarrow *:$	$\big(RREQ, Q_{\mathcal{D}_{Seq}}, Q_{ID}, \mathcal{S}, \mathcal{D}, (), M_{\mathcal{S}}\big\}$
$\mathcal{A} \rightarrow *:$	$\big(RREQ, Q_{\mathcal{D}_{Seq}}, Q_{ID}, \mathcal{S}, \mathcal{D}, (\mathcal{A}), M_{\mathcal{S}}\big\}$
$\mathcal{B} \rightarrow *:$	$\big(RREQ, Q_{\mathcal{D}_{Seq}}, Q_{ID}, \mathcal{S}, \mathcal{D}, (\mathcal{A}, \mathcal{B}), M_{\mathcal{S}}\big\}$
$\mathcal{C} \rightarrow *:$	$\big(RREQ, Q_{\mathcal{D}_{Seq}}, Q_{ID}, \mathcal{S}, \mathcal{D}, (\mathcal{A}, \mathcal{B}, \mathcal{C}), M_{\mathcal{S}}\big\}$
Route Reply Phase	
$\mathcal{D}:$	$M_{\mathcal{D}} = HMAC_{K_{\mathcal{SD}}}\big(RREP, Q_{\mathcal{D}_{Seq}}, Q_{ID}, \mathcal{D}, \mathcal{S}, (\mathcal{A}, \mathcal{B}, \mathcal{C})\big\}$
$\mathcal{D} \rightarrow *:$	$\big(RREP, Q_{\mathcal{D}_{Seq}}, Q_{ID}, \mathcal{D}, \mathcal{S}, (\mathcal{A}, \mathcal{B}, \mathcal{C}), M_{\mathcal{D}}\big\}$
$\mathcal{D} \rightarrow *:$	$\big(RREP, Q_{\mathcal{D}_{Seq}}, Q_{ID}, \mathcal{D}, \mathcal{S}, (\mathcal{A}, \mathcal{B}, \mathcal{C}), M_{\mathcal{D}}\big\}$
$\mathcal{D} \rightarrow *:$	$\big(RREP, Q_{\mathcal{D}_{Seq}}, Q_{ID}, \mathcal{D}, \mathcal{S}, (\mathcal{A}, \mathcal{B}, \mathcal{C}), M_{\mathcal{D}}\big\}$
$\mathcal{D} \rightarrow *:$	$\big(RREP, Q_{\mathcal{D}_{Seq}}, Q_{ID}, \mathcal{D}, \mathcal{S}, (\mathcal{A}, \mathcal{B}, \mathcal{C}), M_{\mathcal{D}}\big\}$

Table 2.4: Secure Route Discovery using SRP.

2.4.5 Building Secure Routing out of an Incomplete Set of Security Association (BISS)

BISS [397] extends Ariadne to networks where there is an incomplete set of security associations by using certificates that were obtained from an off-line CA. The rate of convergence depends on various factors such as *node density*, *mobility*, and *network connectivity*. As shown in Table 2.5, BISS facilitates a source to establish secure association with a destination using its certificate, provided there is a secure association between every intermediate node and the destination. The intermediate nodes and destination utilise the certificate appended by the source to authenticate the route request. In addition, the destination authenticates the intermediate nodes using the secret associations that were previously established with them. On successful authentication of the source and intermediate nodes, the destination responds with a signature protected route reply. In addition, the intermediate nodes that do not have secure associations with the source node append their certificates to the route reply. The source node authenticates the destination and the intermediate nodes by using either the certificate or security association depending on the integrity fields. Although BISS increases the security associations, it also increases the overall network overhead.

Route Request Phase ($\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ have secret associations with $\mathcal{D}$)	
$\mathcal{S}:$	$\sigma_{\mathcal{S}} = \{RREQ, \mathcal{S}, \mathcal{D}, id, ()\}_{K_{\mathcal{S}-private}}$
$\mathcal{S} \rightarrow *:$	$\{RREQ, \mathcal{S}, \mathcal{D}, id, (), (\sigma_{\mathcal{S}}), (\text{Cert}_{\mathcal{S}})\}$
$\mathcal{A}:$	$M_{\mathcal{A}} = HMAC_{K_{\mathcal{AD}}}\{RREQ, \mathcal{S}, \mathcal{D}, id, (\mathcal{A})\}$
$\mathcal{A} \rightarrow *:$	$\{RREQ, \mathcal{S}, \mathcal{D}, id, (\mathcal{A}), (\sigma_{\mathcal{S}}, M_{\mathcal{A}}), (\text{Cert}_{\mathcal{S}})\}$
$\mathcal{B}:$	$M_{\mathcal{B}} = HMAC_{K_{\mathcal{BD}}}\{RREQ, \mathcal{S}, \mathcal{D}, id, (\mathcal{A}, \mathcal{B})\}$
$\mathcal{B} \rightarrow *:$	$\{RREQ, \mathcal{S}, \mathcal{D}, id, (\mathcal{A}, \mathcal{B}), (\sigma_{\mathcal{S}}, M_{\mathcal{A}}, M_{\mathcal{B}}), (\text{Cert}_{\mathcal{S}})\}$
$\mathcal{C}:$	$M_{\mathcal{C}} = HMAC_{K_{\mathcal{CD}}}\{RREQ, \mathcal{S}, \mathcal{D}, id, (\mathcal{A}, \mathcal{B}, \mathcal{C})\}$
$\mathcal{C} \rightarrow *:$	$\{RREQ, \mathcal{S}, \mathcal{D}, id, (\mathcal{A}, \mathcal{B}, \mathcal{C}), (\sigma_{\mathcal{S}}, M_{\mathcal{A}}, M_{\mathcal{B}}, M_{\mathcal{C}}), (\text{Cert}_{\mathcal{S}})\}$
Route Reply Phase ($\mathcal{B}$ has secret association with $\mathcal{S}$)	
$\mathcal{D}:$	$\sigma_{\mathcal{D}} = \{RREP, \mathcal{S}, \mathcal{D}, id, (\mathcal{A}, \mathcal{B}, \mathcal{C})\}_{K_{\mathcal{D}-private}}$
$\mathcal{D} \rightarrow \mathcal{C}:$	$\{RREP, \mathcal{S}, \mathcal{D}, id, (\mathcal{A}, \mathcal{B}, \mathcal{C}), (\sigma_{\mathcal{D}}), (\text{Cert}_{\mathcal{D}})\}$
$\mathcal{C}:$	$\sigma_{\mathcal{C}} = \{RREP, \mathcal{S}, \mathcal{D}, id, (\mathcal{A}, \mathcal{B}, \mathcal{C})\}_{K_{\mathcal{C}-private}}$
$\mathcal{D} \rightarrow \mathcal{B}:$	$\{RREP, \mathcal{S}, \mathcal{D}, id, (\mathcal{A}, \mathcal{B}, \mathcal{C}), (\sigma_{\mathcal{D}}, \sigma_{\mathcal{C}}), (\text{Cert}_{\mathcal{D}}, \text{Cert}_{\mathcal{C}})\}$
$\mathcal{B}:$	$M_{\mathcal{B}} = HMAC_{K_{\mathcal{BS}}}\{RREP, \mathcal{S}, \mathcal{D}, id, (\mathcal{A}, \mathcal{B}, \mathcal{C})\}$
$\mathcal{B} \rightarrow \mathcal{A}:$	$\{RREP, \mathcal{S}, \mathcal{D}, id, (\mathcal{A}, \mathcal{B}, \mathcal{C}), (\sigma_{\mathcal{D}}, \sigma_{\mathcal{C}}, M_{\mathcal{B}}), (\text{Cert}_{\mathcal{D}}, \text{Cert}_{\mathcal{C}})\}$
$\mathcal{A}:$	$\sigma_{\mathcal{A}} = \{RREP, \mathcal{S}, \mathcal{D}, id, (\mathcal{A}, \mathcal{B}, \mathcal{C})\}_{K_{\mathcal{A}-private}}$
$\mathcal{A} \rightarrow \mathcal{S}:$	$\{RREP, \mathcal{S}, \mathcal{D}, id, (\mathcal{A}, \mathcal{B}, \mathcal{C}), (\sigma_{\mathcal{D}}, \sigma_{\mathcal{C}}, M_{\mathcal{B}}, \sigma_{\mathcal{A}}), (\text{Cert}_{\mathcal{D}}, \text{Cert}_{\mathcal{C}}, \text{Cert}_{\mathcal{A}})\}$

TABLE 2.5: Secure Route Discovery using BISS.

2.5 Key Management in MANET

It is evident that secure routing protocols are heavily reliant on the secret associations that are established between the mobile nodes to bootstrap an attack-resistant network service. Therefore, a robust key management service becomes a pre-requisite to establish the secret associations between the mobile nodes so that the service is, (a) *established on the fly without exposing the keying material to unauthorised nodes or compromised nodes*, (b) *capable of performing key refreshment and revocation dynamically*, and (c) *competent to scale well with varying network size and densities*. Numerous solutions have been proposed based on the public-key cryptography [18, 41, 53, 91, 96, 147, 152, 167, 216, 224, 225, 238, 385, 396, 407, 420, 438] and especially based on the threshold cryptography [101, 218, 268, 344] and the identity-based cryptography [54, 55, 110, 196, 431] mechanisms. Alternatively, some solutions focus on a particular aspect of key management such as, key establishment [37, 119, 124, 175, 230, 253, 306, 313, 346, 360, 395, 398, 426, 466], key distribution [88, 89, 110, 324, 329, 365, 413], group key [20, 40, 62, 118, 122, 123, 194, 231, 269, 293, 464], authentication [86, 341, 387, 405], certificate discovery [166] and certificate revocation [18, 100]. As noted in [90, 130, 343], the self-organised feature of the MANET opposes the on-line CA based key management service. Alternatively, the need for one-to-many verification of broadcasted routing packets discourages pairwise keys at least to avoid unique signatures for each receiver. Hence, we confine to distributed certificate and identity-based public-key schemes in the following.

2.5.1 Partially-Distributed Public-Key System

Zhou and Hass [458] distributed the private-key of a CA to a set of nodes known as *server nodes* and then made the public-key of the CA available to all the nodes in the network. The server nodes are capable of generating partial certificates using their own share of the CA's private-key, and k such partial certificates can then generate a valid certificate based on the threshold cryptography. One of the server nodes combines these partial certificates to generate the certificate and hence known as the *combiner*. Server nodes possess their own separate public/private-key pair and assumed to know the public-key of other servers, so that they can communicate the shares of CA's private-key among themselves. Also, the server nodes are responsible for storing the certificates of all the nodes in the network.

In this approach, a mobile node possesses a valid certificate when it joins the network and this certificate is valid only for a certain period of time. Once the certificate expires, the mobile node has to renew its certificate by obtaining shares from k servers and then submitting them to a combiner. The combiner, after joining the partial certificates, checks the validity of the generated certificate to avoid the inclusion of the invalid partial signature generated by any compromised server. Since k shares of CA's private-key are used to generate the certificate, the threshold cryptography can tolerate up to $k - 1$ compromised servers.

Yi and Kravets [437] extend the above solution to delegate the role of certificate construction to the mobile nodes and to specify a certification protocol for retrieving the shares of CA's private-key from the server nodes known as *MObile Certificate Authority (MOCA)*. Hence they eliminate the need for combiner nodes in the network. Similar to DSR, the certification protocol involves a broadcast certification request (*CREQ*) initiated by a mobile node, and the corresponding certification reply (*CREP*) generated by a MOCA that contain the share of CA's private-key. In order to reduce the flooding of CREQ, a mobile node unicasts $\beta - specific$ MOCAs based on the cached routing and also to increase the probability of receiving at least k responses, that is, $\beta = (k + \alpha)$. If a mobile node fails to collect enough shares, then it resorts to the default approach, that is, flooding.

Secure and Efficient Key Management (SEKM) [420] forms a multicast group using MOCAs, in which a mobile node forwards CREQ to a MOCA, and the MOCA in turn forwards CREQ to additional $(k + \alpha)$ MOCAs. However, SEKM fails to state how a MOCA is aware that it is the first server to receive the CREQ. Although MOCA proposes a simple certification revocation list, partially-distributed CA based mechanism struggle to synchronise the servers. The key strength of threshold cryptography *(n, k)* rests on the choice of k. It is unclear on how to choose the parameter *(n, k)* because it is a trade-off between the security and availability of the servers. The factors that are likely to be considered in choosing *(n, k)* are at least the bandwidth, mobility, node density, nature of wireless connectivity, and physical security of the nodes.

2.5.2 Completely-Distributed Public-Key System

Luo *et al.* [240] and Kong *et al.* [203] extend the capability of server nodes to all the mobile nodes in the network. Thus the private key of the CA is shared among all the nodes and a valid signature is generated by combining k such shares according to the threshold cryptography mechanism. They limit the share requests to one-hop, and in parallel enforce mobile nodes to monitor the behaviour of their neighbours to decide whether to provide a share or not. Although the design saves bandwidth and accommodates scalability, it is unrealistic to assume that each mobile node has k neighbours to renew a certificate. Otherwise, it suffers from the earlier-mentioned drawbacks.

2.5.3 Identity-based Public-Key System

In identity-based public-key system, the identity serves as the public-key in order to remove the necessity for certificates, and also to eliminate the overhead and scalability. Identity-based cryptography consists of a *Private Key Generator (PKG)* to generate the private-key signatures corresponding to the user identities, *i.e.* the public-keys. A detailed description of identity-based cryptography can be found at [56]. In [196], Khalili *et al.* divide the PKG of identity-based cryptography into k thresholds based on the threshold cryptography mechanism discussed earlier. Hence, a mobile node collects k shares from the PKG nodes to derive a private-key for its identity. However, the proposal fails to demonstrate the integration of identity-based cryptography into threshold cryptography, and struggles to address the previously-mentioned issues such as key revocation, PKG update, *etc.*

2.5.4 Transitive-Chain based Public-Key System

In [396], Capkun *et al.* suggest an approach similar to *Pretty Good Privacy (PGP)* [467]. The mobile nodes generate their own public/private-key pairs and issue certificates to trustworthy nodes similar to PGP. However, they store certificates rather than publishing the certificates in a centralised repository. Whenever two mobile nodes anticipate to validate their certificates, they merge their local certificate repositories to find a valid certificate chain. Therefore, the strength of this proposal primarily rests on the construction of local repositories. The authors propose various algorithms to construct local repositories and to analyse the performance of their proposed algorithms.

In *Maximum Degree Algorithm*, the authors consider each node to store several directed and mutually disjointed paths of certificates in its local repository such that each path leads to nodes that have the highest number of disjointed paths of certificates. In *Shortcut Hunter Algorithm*, certificates are stored into the local repositories based on the number of short chain certificates that are connected to the nodes. Although the proposal addresses simple certification revocation, and renewal, it is unclear on how synchronisation is achieved among the mobile nodes. Although the proposed approach suits with the self-organised characteristic of the MANET by not assuming the existence of an on-line or off-line CA for bootstrap, the proposed approach is only probabilistic and not deterministic, and strong as its weakest link. If anyone of the mobile nodes is compromised in the chain then it eventually leads to a *false authentication.* Otherwise, the chain itself is open to the issues of trust transitivity, where there is no requirement for a mobile node to trust a recommended node in the same manner as it trusts the recommender. The same authors have also shown how mobility can be taken advantage to establish secret associations in [398].

2.5.5 Limitations

Although the above-mentioned public-key based systems eliminate the limitations of a CA, they struggle to address some aspects of key management, such as, key refreshment, revocation, and certificate synchronisation in the MANET. This is so important given that mobile nodes are prone to physical capture and compromise. Perhaps the difficulty arises because of the sporadic connections induced by dynamically changing topology and autonomous structure of the network where nodes are allowed to enter and leave the network at their will. Hence any vulnerability in the key management service can permit adversaries to disintegrate the strength of the secure routing protocols. Furthermore, key management and secure routing systems are incapable of defending against a set of adversaries that disrupts either the key refreshment or revocation by interrupting the operation in the form of packet dropping or flooding. The same holds true whenever a genuine link is reported as a broken link. Since secure routing protocols are only designed to prevent against predefined attacks and also assume that all the nodes in the network would carry-out the routing and network management, they overlook the correct execution of the critical network functions such as packet forwarding. For this reason, secure routing protocols also fail to enforce cooperation among nodes and, therefore, allow a separate category of adversary known as selfish nodes to maximise their utility in the resource constrained MANET.

2.6 Incentive and Game-theory based Systems

Recall that incentive-based [51, 77, 79, 102, 103, 127, 151, 209, 248, 278, 318, 320, 334, 349, 408, 416, 441, 443, 452, 453, 456] and game-theory based [10, 14, 80, 172, 255, 257, 358, 359, 391, 457] systems have been proposed to stimulate cooperation among the mobile nodes. In the following, we briefly discuss the working of *Nuglets* [77], and *Sprite* [456] for the sake of completeness.

In [77], virtual currencies known as *nuglets* are used to stimulate cooperation among the mobile nodes. These nuglets can be deployed in one of the following ways, (a) *Packet Purse Model (PPM)* in which the source of the packet is charged and (b) *Packet Trade Model (PTM)* in which the destination of the packet is charged. The proposal assumes a tamper-resistant hardware at each node to assure a fair usage of nuglets. In PPM, the source node propagates a packet with sufficient count of nuglets so that the intermediate nodes can debit the nuglets from the packet for forwarding that packet. However, if the packet runs out of nuglets then that packet is dropped. The key advantage of this approach is that it forces the originator to pay for each propagated packet thereby defending against junk transmissions. The main drawback of this approach is the necessity for the originator to pre-determine the count of nuglets. Alternatively in the PTM, each hop buys the packet and sells it to the next-hop. If the receiving node could not sell a packet, it tries for another node, otherwise it drops the packet. The main advantage is that there is no need for the source to predict the required count of nuglets for transmission, but on the other hand, the approach fails to prevent the network from being overloaded with unwanted transmissions.

Sprite eliminates the usage of a tamper-resistant hardware by providing incentives to stimulate cooperation among the selfish nodes. However, the proposed approach relies on the presence of a CA known as the *Credit Clearance Service (CCS)* to charge the mobile nodes that generate packets and credit the intermediate nodes that produce receipts for forwarding those packets. Hence they discourage the destination to pay for the propagation of a packet in order to defend against DoS attacks and also to discourage the source node from colluding with the intermediate nodes.

In general, incentive-based or game-theory systems assume nodes to be economically rational, and consider either a tamper-proof hardware or trusted third parties to make incentives unforgeable. Furthermore, incentive-based systems struggle to address the issue of pricing [165], and other related issues, such as, how to deal with the deprivation of incentives, *etc.* They express poor insight with respect to boundary nodes, especially when boundary nodes are willing to cooperate with other nodes to earn incentives but unable to do so due to the lack of opportunities. In addition, most of the incentive-based systems confine only to selfish nodes that attempt to retain their battery resource by not forwarding packets for other nodes. They fail to defend against malicious nodes that disrupt the transmissions of other nodes by either dropping or flooding packets.

2.7 Reputation and Trust Management Systems

Since history of security has shown that a completely intrusion-free system is infeasible [428] no matter how carefully the prevention mechanism is designed, therefore there is an urge to opt for defence-in-depth strategy in the MANET. The main reason for the shortcoming of secure routing and incentive based systems is that they either fail to capitalise the available knowledge or fall short to dynamically detect the misbehaviours. This has led to the development of several detection-based systems [13, 17, 104, 111, 213, 237, 251, 286, 288, 323, 342, 366, 371, 400, 411, 430] for the MANET and, most of them based on the anomalies [11, 12, 369, 370, 388, 389] and signatures [16, 46, 361]. Few of those detection-based systems are inspired by the natural immune system of vertebrates [58, 300, 338–340]. Realising the importance of merging prevention and detection systems, some proposals integrate IDS into secure routing approaches [195, 246, 368, 429], while few others combine IDS with cooperation based approaches [8, 73, 189, 289, 290, 362, 403]. However, these proposals fall short by failing to measure the trustworthiness of nodes using the dynamically changing behaviours of those nodes so that they could effectively react to those behaviours accordingly. This has eventually led to the growth of *trust management systems* [3, 38, 39, 64–71, 82, 83, 107, 112, 113, 116, 125, 134, 137, 140, 169, 178, 190, 204, 212, 214, 220–222, 233–235, 239, 241, 242, 252, 261, 270–273, 292, 294, 301–305, 308–311, 327, 328, 335, 347, 372, 373, 379, 380, 401, 404, 412, 424, 425, 461, 468] in the MANET, which are synonymously referred as *detection reaction*, and *reputation systems* in the literature.

Since trust management systems proactively detect and reactively isolate (or select) malicious (or benign) nodes, these systems are also known as *self-policing systems* [72]. Although these systems lack consensus on the definition of *trust* at a more fundamental level, they adhere to all or some of the following steps. First, these systems tend to collect evidence for the behaviours of other nodes through passive monitoring, acknowledgements, or IDS [8, 94, 191, 444]. They may also consider evidence from other nodes and define suitable approach for the evidence distribution. This is often regarded as *recommendations*. Second, trust management systems subjectively evaluate and rate the evidence, and then utilise those ratings to predict the future behaviour of other nodes. Finally, they make decisions based on the policies defined for the context and the behaviour anticipated for those nodes that are involved in the context. In this section, we investigate the concepts of *trust* and *reputation*, and the methodology in trust management systems from the perspective of distributed computing to establish the groundwork for our analysis and their significance in the MANET.

2.7.1 Concepts – Trust and Reputation

Trust management and trustworthy computing are becoming increasingly significant in a distributed environment as they assist in making sensible interactions with unknown parties by providing a basis for more detailed and automated decisions [332]. The concepts, *trust* and *reputation* are closely related in trust management systems [266] and they are firmly rooted in sociology and psychology. Although there is no universal definition for these concepts due to their rich connections to different disciplines, we confine to their meaning within the computing discipline.

In traditional trust management systems, trust enables a trustor to reduce uncertainty in its future interactions with a trustee, who is beyond the control of trustor but whose actions are of interest to the trustor and also affects the state of the trustor. In other words, trust is a subjective probability which enables the trustor to take a binary decision by balancing between the known risks and the opinion held for the trustee. Here, only known risks are considered for making decisions as it is difficult to prove unknown risks, and therefore, the opinion presents the trustor's relationship with the trustee based on the trustor's experiences. The other factors influencing the decision are time and context, where context accounts for the type of interaction between the trustor and trustee, and the nature of application.

In reputation systems, reputation is defined as the opinion held by the trustor towards the trustee depending on its past experiences with the trustee [266]. In other words, reputation represents the trustor's direct relationship with the trustee. Similarly, the trustor's relationship with a second trustee based on its direct relationship with a first trustee and the first trustee's direct relationship with the second trustee is known as *indirect relationship.* These relationships are established by enabling nodes to share their opinions in the network.

Although trust and reputation are used interchangeably in the MANET, we precisely define them in the following because they complement each other based on the above inference. Therefore, trust can be defined as the prediction of a node's future action in a context such as forwarding routing messages without modification, while reputation then becomes the opinion held for the node based on the node's past actions and the one that influences the prediction [233]. For this reason, we consider the trust definition in [22] to be more appropriate and timely, "*Trust is the firm belief in the competence of an entity to act as expected such that this firm belief is not a fixed value associated with the entity but rather subject to the entity's behaviour (*i.e. *the reputation held for the entity) and applies only within the context and at a given time*".

2.7.2 Components of Trust Management Systems

Trust management systems are characterised by the following components, (a) *evidence manager to collect and classify evidence*, (b) *mathematical model to formulate the evidence into opinion, and then to apply them to predict the result of future interaction*, and (c) *policy manager to define decision policies for making decisions.* Traditional trust management systems were monotonic as they were only designed to authenticate users for granting access to services using digital certificates. Here, the digital certificate is the evidence, and the cryptography mechanism evaluates the certificate and makes necessary opinion regarding the identity of the user. Finally, the policy manager defines simple decision policy such as, granting access to the user whose identity matches with the digital certificate. Such a simple trust management system may be warranted in a closed organisation with a CA, but the same fails to hold in a distributed system where no hierarchical relationships exists between entities. Although the policy system [47, 48] later expanded significantly, the decisions were yet based on the credentials presented by the user.

Meantime the development of IDS to meet the shortcoming of prevention systems has led the researchers to consider IDS to contribute behavioural evidence for the trust management systems. The reason for the merge arises from the inflexibility of IDS to take full advantage of the behavioural evidence and the inability of the trust management systems to adapt to peer-to-peer distributed networks. Accordingly, Lin *et al.* [226–228] refer such behavioural evidence as *soft evidence* and credentials used for authentication as *hard evidence*. Other approaches expanded their focus to different dimensions whilst evaluating the trustworthiness of a system, such as the context [382, 383], formal framework [138], social resemblance [138] and ignorance [183–185, 187, 188].

2.7.3 Significance in the MANET

Given the concepts and methodologies in trust management systems have been briefed above along the lines of few well-established proposals [5, 22, 45, 106, 129, 133, 141, 142, 144, 148, 149, 186, 211, 245, 266, 317, 332, 351, 376, 386, 393, 394, 432, 462, 463] from the literature, let us now consider their significance in the MANET. Majority of the trust management systems proposed for the MANET are decentralised and deployed at every node to make subjective decisions. Most of these systems consider only behavioural evidence, and collect the evidence through passive monitoring, acknowledgements, or IDS. Note that the behavioural evidence collected by passive monitoring are based upon the assumption that the evidence are observable and classifiable. Alternatively, the feature to accommodate IDS to collect behavioural evidence facilitates trust management systems to handle new type of attacks in the MANET. This is in contrast to secure routing and payment systems, where the protocol design has to be changed for every new type of attack. Furthermore, few trust management systems [64–72] consider the evidence related to a system failure as malicious behaviour in order to realise a reliable and trustworthy MANET. Most of these trust management systems [3, 39, 64, 66, 68–71, 82, 83, 107, 112, 116, 125, 134, 137, 140, 169, 178, 190, 204, 214, 220–222, 233–235, 239, 241, 242, 252, 261, 270–273, 292, 294, 301–305, 308–311, 327, 335, 347, 372, 373, 379, 380, 401, 404, 424, 425, 461, 468] also facilitate mobile nodes to exchange opinions with other nodes (in the form of reputation ratings) so that they can enhance their decisions. However, they neither rely on higher authorities nor required to reach consensus with other nodes to make a decision. The reputation ratings that are disseminated to communicate opinions with other nodes as referred as *recommendations* and they facilitate the nodes to establish indirect relationships.

Trust management systems in the MANET employ various mathematical models such as graph-theory based models [234], entropy-based models [372, 373], Bayesian-logic based models [70, 71], *etc*, to quantify the evidence into opinion and represent the resulting opinion as a relationship between the nodes. Finally, these systems execute simple decision policies, such as, whether to, (a) *accept or reject a newly discovered route*, (b) *send or forward a packet on behalf of other nodes, accept or ignore a recommendation from another node*, (c) *delete or retain paths containing misbehaving nodes*, (d) *abstain or warn others about the misbehaving nodes*, and (e) *which path to choose for communication.* Capra [83] models trust management systems based on human behaviour; however, it is not certain whether such system will meet the requirements of the MANET because humans do not seem to always make fully rational trust decisions [182]. In the following, we choose few recently proposed and some well-reviewed systems for analysis from the literature. In our analysis, we use the terms, *opinion*, *reputation* and *relationship* interchangeably based on the above-established hypothesis.

2.7.4 State of the Art

2.7.4.1 Quantifying Trust in MANET

Virendra *et al.* [401] have proposed a trust model in which trust relationships are utilised to enable nodes to establish keys with other nodes. In their proposal, they eliminate the following assumptions, such as, (a) *centralised trusted authority to establish keys*, and (b) *nodes are non-malicious at the time of key establishment.* The decentralised trust model at every node adopts a five-phased approach to establish and maintain trust relationship with other nodes, and the phases are, (a) *initiation and monitoring*, (b) *query and evaluation*, (c) *updating*, (d) *restructuring*, and (e) *re-establishment.* The model implements a metric to represent the opinions or relationships in the range of 0 to 1, and further sub-divides the range into three regions,

(a) *bad*, (b) *uncertain*, and (c) *good* respectively. However, the model lacks sound mathematical proof for the chosen metric.

During the initiation and monitoring phase, a node fixes its relationship with other nodes to the median of the uncertain region. It promiscuously monitors the packet transmissions of neighbours to collect the evidence of trustworthiness for their benign and malicious behaviours. Also, it abstains from sending sensitive data to neighbours until keys are established with them, for which its relationship with them has to be in the good region. The query and evaluation phase facilitates the node to collect recommendations from neighbours. However, recommendations are collected only from the neighbours for which the relationship is in the good region. Also, recommendations are treated secondary to the evidence collected through promiscuous monitoring. The updating phase regularly evaluates the relationship depending on the evidence incoming through monitoring and recommendations. As the relationship shifts towards the good region, the periodicity of evaluation is reduced. The restructuring phase addresses the uncertainty that may enter into the relationship, whenever, (a) *neighbours move out of the transmission range*, and (b) *neighbours had previously moved out of the environment are now back into the transmission range.* Instance when the neighbours are out of range, the node exponentially decays its relationship to the floor of the region (bad, uncertain, or good) to which the trust relationship belongs to. Alternatively, the decayed relationship is resumed when the neighbours come back into the node's vicinity. Finally, the re-establishment phase is an optional phase in which a malicious neighbour re-establishes relationship with the node by performing a linear benign behaviour.

The node then utilises its trust relationship to establish pair-wise and group-keys, where keys are established with the neighbours for which trust relationship is in the good region. Pair-wise keys are established with neighbours using mechanism such as *duckling* [360]. In group-key establishment, one of the nodes in the group invites all the neighbours to form a cluster, based on the trust relationships which is known as *Physical Logical Trust Domains (PLTD).* Rest of the neighbours may accept the invitation to join the group or in turn invite the node to join their PLTD. The model also permits a node to participate in more than one PLTD.

The model's novelty rests on its scheme to employ behavioural and recommendation based trust relationships to establish pair-wise and group-keys among nodes. However, it is unclear how the model defends against nodes that perform malicious attacks by spoofing packets with the identities of other nodes before the key establishment phase. Other drawbacks are the model's inability to handle recommendation issues such as honest-elicitation, free-riding , and recommender's bias. Finally during the re-establishment phase, it is unclear on how nodes select a malicious node for forwarding packets while trustworthy neighbours are in the neighbourhood.

2.7.4.2 Establishing Trust in Pure Ad-hoc Networks

Pirzada and McDonald [305] have proposed a trust model to improve reliability in the pure MANET. The model performs three tasks, namely (a) *trust derivation*, (b) *trust quantification*, and (c) *trust computation*. Trust derivation is synonymous to the evidence manager and collects evidence for each type of packet (*e.g.* route request and route reply packets) in all the categories such as forwarding data packets, forwarding routing packets without modification, providing gratuitous route replies, and salvaging route errors. Here, the opinion is referred as trust and measured in continuous values between -1 and +1. Similar to [401], the model fails to provide a sound mathematical proof for the chosen trust metric. Trust quantification formulates opinion for each category by combining the trust metric held for different packet types within the category. To achieve this, the operation assigns a weight for each packet type such that it represents the concepts, (a) *utility* and (b) *importance*, as referred by Marsh in [245]. The value for the weight ranges from 0 (unimportant) to +1 (important). Here, the utility refers to the cost and benefits associated with the context, and importance refers to the significance of the context based on time. Finally, the trust computation is responsible for computing the trustworthiness of a node by combining the trust quantified for each category. For this, it adopts the same strategy of assigning weights to each category before combining them together.

Unlike the previous model [305], the system design composes of only evidence manager and weak computational logic. It overlooks one of the trust management components, decision manager, because it fails to take advantage of the established trust relationships between nodes. It is not clear whether the evidence manager, *i.e.* the trust derivation considers recommendations, and also how the accuracy of gratuitous route replies and salvaged route errors are verified especially in a pure MANET where there is no pre-established relationship among nodes. This is because nodes can maliciously report a genuine link as broken link and introduce tampered routes through gratuitous route replies and salvaged route errors. Similar to [305], Pirzada and McDonald also fail to demonstrate the strengths and limitations of their proposed models through simulation results.

2.7.4.3 Propagating Trust in Ad-Hoc Networks for Reliable Routing

Pirzada *et al.* enhanced their earlier trust model [305] in [301] to address the question of decision manager and recommendations. Their latter model incorporates the previous model as a trust agent to collect the behavioural evidence and in sequent to compute the *direct trust* for a node. Here, reputation agent is responsible for sharing direct trust values as recommendations with other nodes, and the opinion derived from recommendations is known as *derived trust*. To achieve this, the recommendation is scaled with the direct trust held for the recommender. In order to keep fallacious recommendation requests and recommendations at bay, the authors suggest *HashCash*[1]. The combiner then combines direct and derived trusts from trust and reputation agents respectively to arrive at the *aggregate trust*. Finally, trustworthy routes are chosen by assigning weights to each link in the route based on the aggregate trust and then employing shortest path algorithm using the assigned weights.

Pirzada *et al.* further extended their latest model in [309] to elaborate the dissemination of recommendations and demonstrated the effectiveness of their extended

[1]A CPU cost function that computes a proof-of-effort token using a cryptographic hash function. The requester tries different combinations of trial string to find a token, and then sends the token to the recommender, which can easily verify the token by performing a single hash operation.

model through simulation results. In their extended model, nodes disseminate recommendations by appending the direct trust values for other nodes along with the data packets. This successfully reduces the overhead that may otherwise occur by forwarding separate packets to disseminate recommendations. The simulation results show that DSR coupled with the trust model perform 10% better than the normal DSR. Since the model excludes malicious nodes by resorting to only trustworthy routes, it is interesting to know how the model will include repenting malicious nodes in the future. Also, it is speculative whether a node appends its direct trust for all the nodes along with the data packet or only for a few selected nodes. Finally, the notion of disseminating recommendations opens door to honest-elicitation, free-riding, and bias of the recommender.

2.7.4.4 Performance Analyses of the CONFIDANT Protocol (Cooperation Of Nodes: Fairness In Dynamic Ad-Hoc NeTworks)

Buchegger and Boudec adopted defence-in-depth strategy to enhance the strength of prevention mechanism by proposing a trust management system known as *CONFIDANT* [64, 67]. They utilise the trust model to defend against attacks, such as, (a) *traffic deviation*, (b) *route salvaging*, (c) *unusual frequent route updates*, (d) *silent route tampering*, (e) *lack of error messages*, and (f) *no forwarding of control or data packets.* The motivation for their proposal is derived from Richard Dawking's *The Selfish Gene* [108], in which *suckers* and *cheaters* are demonstrated as losers, while *grudgers* as winners.

The CONFIDANT collects evidence from direct experiences, and recommendations. Later, relationships are established based on the evidence that are then used for trust decisions. These operations are achieved through four interdependent modules, (a) *monitor*, (b) *reputation system*, (c) *path manager*, and (d) *trust manager*. Monitor collects evidence by passively monitoring the transmission of a neighbour after forwarding a packet to the neighbour. It then reports to the reputation system that consists of a rating-list and blacklist, only if the collected evidence represents a malicious behaviour. Reputation system changes the rating for a node if the evidence for a node's malicious behaviours exceeds the pre-defined threshold value. It is important to note that the evidence collected for a malicious behaviour from a direct experience has more influence than the evidence collected from a recommendation. In sequent, the path manager makes a decision to delete the malicious node from the path. Also, the path manager assists the node in making decision such as whether to forward a received packet by cross-checking the upstream node's identity (previous-hop) in the blacklist. The reputation system calls for a time-out operation to handle false accusations, fault rating and list blow-up. Finally, the trust manager is responsible for forwarding and receiving recommendations to and from trustworthy nodes. Here, the recommendations are known as ALARM messages and the trustworthy nodes are referred as *friends*. The ALARM messages received from friends are evaluated for trustworthiness before being sent to the reputation system. The trust decisions pertaining to the trust manager are, (a) *providing and accepting routing information*, (b) *accepting a node as a part of route*, and (c) *taking part in a route originated by some other node*. The authors demonstrate the effectiveness of the model through extensive simulation results. Interestingly, CONFIDANT nodes drop around 100 packets in the presence of 90% malicious nodes, while DSR nodes drop 10000 packets in the presence of 10% malicious nodes. Also, CONFIDANT nodes deliver 75% of data packets in the presence of 60% malicious nodes.

Similar to the models discussed earlier, a friend's ALARM message may not reflect the actual status as the friend may be biased. In order to address honest-elicitation, the authors have extended the model in [70, 71] to accommodate recommendations that are in agreement with their opinions and accepted from friends whose trustworthiness are measured separately for forwarding honest recommendations. However, such a design fails to consider recommendations that report unusual behaviour because of the disagreement. Furthermore, the model fails to handle free-riding behaviour. The notion of refreshing the rating-list and blacklist using time-outs are reconsidered in their later revision [68] to avoid the re-entry of malicious nodes.

2.7.4.5 Information Theoretic Framework of Trust Modeling and Evaluation for the MANET

Yan Lindsay *et al.* have developed a trust model for improving the security of MANET routing protocols in [373]. Similar to related models, the proposed trust model establishes trust relationships using the evidence collected through monitoring and recommendations. The established trust relationships are then used to choose trustworthy routes and to isolate malicious nodes. The model differs from related model by laying axioms for measuring, propagating, and combining trust. This is because the authors believe that it is hard to compare trust metrics from various trust models and believe that the fundamental definition of trust is incomplete. Similar to [190, 220–222, 379, 380], they define trust as uncertainty and measure the uncertainty using entropy. The evidence collected for malicious and benign behaviours are probabilistically mapped by following a modified Bayesian approach. The modification allows the model to give low importance to past evidence and more importance to recent evidence. The probabilistic estimate of Bayesian approach is then mapped to entropy.

In the case of recommendations, the authors ascertain that the trust established through a recommendation should not be more than the trust held for the recommender or the recommendation. Also, they state that the trust built using the multiple recommendations received from the same node should not be greater than the trust built using the recommendations received from independent nodes. To prevent nodes from knowing a requesting node's favourite recommenders, the node disseminates a *Trust Recommendation Request (TRR)* that contains a chosen list of nodes for which its recommendation trust is greater than the threshold, and also includes few other nodes for which it is interested to update its recommendation trust. The requesting node receives recommendations only from the list of nodes that are specified in the TRR packet and restricted by the defined time-to-live (TTL). Note that it is up to the recommender to respond back to the TRR. Later the recommendation trust of the recommenders is updated depending on the interaction with the node for which the recommendation was received.

The authors have performed intensive simulations using three types of malicious nodes (a) *packet droppers*, (b) *recommenders that are liars* and (c) *blend of liars and packet droppers*. Simulation results show that malicious nodes do not form clusters because once nodes are identified for malicious behaviours then there will be no more routes through them. In the case of selectively malicious behaviours, few of the benign nodes are mistakenly considered as malicious nodes. Finally, the authors observe mobility to produce false negatives against benign nodes for dropping packets. However, mobility is believed to decrease the overhead of requesting more recommendations because nodes move around and meet other nodes frequently. Interestingly, the paper lays definition for trust and specifies the properties for propagating trust. However, recommendations are prone to recommender's bias, honest-elicitation and free-riding problems. In addition, it is unclear how TRR and TTL are protected against modification attacks.

2.7.5 Limitations

In this section, we summarise our analysis of various trust models that have been presented above. Although trust management systems in the MANET have introduced a defence-in-depth strategy to enhance the security of communications, they are still incomplete due to some certain limitations and shortcomings. We discuss these limitations and shortcomings and point towards the possible future directions. Interestingly, trust models [128, 217, 326, 447] that have been proposed in last couple of years are still prone to some of the limitations discussed in the following.

2.7.5.1 Behavioural Evidence

Most of the trust management systems in the MANET consider only behavioural evidence since they are observable and classifiable. However, it is not possible collect evidence for all observable behaviours or the collected evidence may not be accurate due to, (a) *ambiguous collisions*, (b) *receiver collisions*, and (c) *limited transmission power*. Ambiguous collisions occur when a node receives a packet while it passively monitors the transmission of another node. Alternatively, receiver collision occurs when a passively monitored node transmits a packet to its next-hop, at the same time the next-hop is receiving another packet from some other node. In such situation, even if the passively monitored node fails to retransmit the packet, it can increase its reputation at the monitoring node. Finally, a selfish node can circumvent the monitoring node by varying its transmission power such that the signal reaches the monitoring node but not to the selfish node's next-hop. Note that the malicious node requires to know the transmission power range to reach each of its neighbours.

The severity increases when trust management systems collect only behavioural evidence and fail to consider credentials or secret associations to authenticate nodes. In the absence of authentication, evidence may be collected from a spoofed packet and, hence, the corresponding trust relationships and resulting decisions may be inaccurate. This clearly advocates the need to authenticate intermediate nodes and to verify routing messages using secure routing protocols and only few models [3, 107, 241, 425] acknowledge such a requirement. For this reason, it is necessary to collect both hard and soft evidence so that trust management systems can also be employed to assist secure routing protocols in enlisting only trustworthy nodes to the routes.

2.7.5.2 Recommendations

In the MANET, if nodes happen to rely only on the evidence collected from direct interactions, then they can defend against malicious nodes only after interacting with them. For this reason, mobile nodes exchange their opinions for other nodes with their neighbours in the form of recommendations. In this manner, they partially avoid encountering malicious nodes without interacting with them. However, recommendations are effective only if the trust model is capable of deploying some strategies to eliminate or at least cope with, (a) *false accusations or false praise, which we collectively refer as honest-elicitation*, (b) *free-riding in which neighbours fail to reciprocate with recommendations* and (c) *recommender's bias resulting from recommender's subjective evaluation of evidence.*

A straightforward defence against honest-elicitation in the literature is to consensus the received recommendations so that valid recommendations are differentiated from falsified recommendations. However, this approach is vulnerable to collusion attack, if the recommendations generated by colluding malicious neighbours surpass the recommendations disseminated by benign neighbours. Hence, it becomes important not to penalise any recommender as honest recommendations may fail to emerge due to the consensus approach. Alternatively, a mobile node can defend against honest-elicitation by only considering recommendations that are in agreement with its belief. Such an approach would overlook the recommendation that may otherwise report the infrequent malicious behaviour of a selectively-benign neighbour. Recommender's bias is another issue pertaining to recommendations. Unlike honest-elicitation, here the recommender may disseminate an honest recommendation which is derived from either an optimistic or pessimistic evaluation of collected evidence. Such recommendations are critical in transitive trust relationships, where a node receiving a recommendation believes that the recommender has evaluated the trustworthiness of recommended node in the same manner it evaluates the trustworthiness of recommender. Free-riding confronts the advantages of recommendations as it adheres to the specifications of routing protocol for forwarding packets but fails to follow the specification outlined for the trust model. One of the solutions to thwart free-riding is to opt for recursive design so that nodes are enforced to communicate recommendations with other nodes as they are enforced to forward packets for other nodes. Finally propagating recommendations opens door to many questions such as whether to, (a) *restrict recommendations to single or multiple hops*, and (b) *attach opinions for all nodes or report only malicious nodes.* The answer towards such questions rests on the balance between the overhead incurred in disseminating recommendations and the importance of informing other nodes regarding malicious nodes.

These issues lead us to question whether recommendations are necessary at all in the trust management systems. However, the benefit of defending against malicious nodes even without interacting with them prioritises the necessity to consider recommendations. Perhaps, the direction to look for solution is to investigate the trade-off between the advantage of considering recommendations to establish trust relationships with unknown nodes and the benefit of ignoring recommendations as a result of their vulnerabilities. In comparison with the related trust models, we emerge to establish the balance by addressing the issues pertaining to recommendations and at the same time taking advantage of the benefits of communicating recommendations in Chapter 4.

2.7.5.3 Modelling Ignorance in Trust Relationships

Most if not all trust models, often fail to represent the aspect of ignorance and the associated uncertainty, which are intrinsic to the establishment of trust relationships in the MANET. Hence, a trust relationship between two nodes may not always reflect the actual relationship and consequently the executed decision may not always be accurate. For instance, consider a new node joining the network. The existing nodes in the network may not have a record of past evidence to trust or distrust a newly joining node. Assigning an arbitrary level of trust for the new node poses several issues. The trust models address this issue either by pessimistically assigning a low level of trust [294] or by optimistically assigning a neutral [401] or high level of trust [301] to the new node. The purpose of pessimistic approach is to compel the new node to exhibit a consistent benign behaviour from the point it enters the network. However, in some of these models, it is not always clear as to how the less trusted new node is selected for communications when nodes with high trust values exist in the network. If a new node is not preferred for communications due to its low level of trust, then it lacks the opportunity to gain trust with the existing nodes. Alternatively, with the optimistic approach, the aim is to promptly identify whether the new node exhibits malicious behaviour from the point it enters into the network. Prompt identification is feasible because the neutral or high level of trust assigned for a new node decreases rapidly as the malicious behaviour increases. However, the optimistic approach favours existing malicious nodes to re-enter the network with a new identity. These issues arise as trust models explicitly fail to represent an existing node's ignorance about a newly joining node's behaviour. The issue also extends to nodes that have already established trust relationship with one another depending on their interactions, and recommendations. For example, when a node moves away from a neighbour (due to mobility), it is unclear whether to consider the neighbour with the same level of trust or distrust during the next interaction (when it returns). It may be that the neighbour could retain its current behaviour which may be either benign or malicious. Considering the neighbour to be benign, there is a chance for the neighbour to be compromised prior to the next interaction. Alternatively, if the neighbour is considered to be malicious, then it may be repenting and expecting for an interaction to improve its relationship. Another possibility is that the neighbour may be malicious as a result of compromise, and it may be redeemed prior to the next interaction. These issues are often addressed by either increasing or decreasing the trust or distrust of the nodes in proportion to the duration for which they are out of communication. The seriousness of this approach is that a node's ignorance of other nodes is represented by either increasing or decreasing its trust or distrust for them, which indeed denotes to their benign or malicious behaviour respectively. Hence the failing to explicitly represent the notion of ignorance and the associated uncertainty has a fundamental impact on the trust model. Recently few trust models [82, 220, 373, 379] have considered this issue by modelling ignorance in the established trust relationships. However, they are vulnerable to at least one of the issues related to the recommendations.

2.7.5.4 Selective Malicious Behaviours

Majority of the trust models consider mobile nodes to exhibit persistent malicious or benign behaviours. This is not true in the MANET where nodes are either prone to capture and compromise or redeemed after being compromised. Modelling trust management systems to handle such selective malicious behaviours enable prompt detection and isolation of compromised benign nodes. Alternatively, it enables the detection of malicious nodes that may perform benign behaviours to remain unidentified or to gain trustworthiness in its neighbourhood for a future objective. Similar approach can be employed to detect selfish nodes that aim for maximum utility by selectively dropping the packets received from neighbours. Although few models [64, 66, 68–70] measure the relative rate of such selective malicious behaviours, they fail to take advantage of their measurement by policing such nodes into a separate category. By defining efficient policies, these nodes can then be included in a communication only if there is no available path through other nodes and it is an non-sensitive communication.

2.7.5.5 Evidence to Opinion Mapping

Only few models [39, 64, 66, 68–70, 82, 220, 233, 234, 270, 271, 373, 373, 379] are built on strong mathematical proofs to map evidence to opinion. Remaining models provide weak basis for evidence to opinion mapping and also fail to provide valid reasoning for the continuous or discrete values chosen for the trust metric. This consequently leads to inconsistencies in the evaluations and introduces difficulty in comparing the results with other models. The other issue in managing opinions is, whether to give more weightage to recent or past behaviours. Past behaviours are given more weightage in order to avoid recent wrong observation to overshadow past benign behaviours; however, it is also important to prevent nodes from taking leverage of past benign behaviours. Since trust evolves continuously in a dynamically changing MANET, it is advantageous to give more importance to recent behaviours. In such case, even if a benign node's behaviour has been mistakenly evaluated for malicious behaviour, the model should be designed to allow benign nodes to regain their reputation through their future benign behaviours. In other words, the logic should not reverse the existing opinion upside down. Another important factor that needs consideration while choosing the mathematical logic is the ability of the logic to prioritise different types of opinion. For example, opinion formulated from one-to-one interactions should take more weightage than the opinion formulated from recommendations.

Finally, not many models [242, 247] enable nodes to call for a strict decision to exclude malicious nodes in their communication and to deny the requests received from malicious nodes. Such models has to be designed to take advantage of the established trust relationships so that they could efficiently make decisions to, (a) *accept or reject a discovered route*, (b) *record or ignore a route from a forwarded packet*, (c) *to forward or discard a packet*, (d) *to forward a packet for a previous-hop*, (e) *to send a packet to a next-hop*, (f) *refresh or revoke a secret association with a node*, and (g) *which route to choose for the communication.* In summary, dynamic and efficient trust management systems can be realised only if the decision policies evolve with the evidence manager that has been designed already to accommodate IDS for detecting new type of attacks.

2.8 Motivation and Security Requirements

In this section, we summarise the security requirements upon which our two-layer trust enhanced security architecture will be built. Note that these security requirements are motivated from the literature study of related prevention and detection-reaction systems. We envisage to meet the following requirements in our proposed architecture.

- The systems and sub-systems of the architecture must complement and operate within the assumptions and inherent limitations of the MANET. For example, the systems must consider mobility and dynamically changing topology, resource heterogeneity, promiscuous wireless medium and self-organised nature of the MANET in their design. On the other hand, these systems are expected to be designed on realistic and achievable assumptions. Furthermore, they are anticipated to scale and adapt to the dynamically changing environment of the MANET. For instance, taking into account the internal and external factors, such as, congestion and contention in their operations. Finally, they must be decentralised, independent and subjective, *i.e.* node centric to complement the heterogeneity factors of the MANET.

- The designed systems must resolve the existing problems for which they are proposed rather than introducing new vulnerabilities to the architecture. Also, they must not rely on any online centralised or distributed trusted authorities for their successful operation, but they are permitted to take leverage of an offline trust entity to bootstrap authentication. These credentials can be used to establish secure associations and thereof to establish secure data communications.

- Given that it is hard to eliminate the dissemination of additional control messages in the case of prevention systems (such as, secure routing protocols), the detection and reaction systems are anticipated not to add further overhead to the architecture. In other words, the detection and reaction systems must be designed to communicate the control information without disseminating additional messages for those control information. As per the label, detection and reaction systems must be designed to identify and isolate malicious nodes until those malicious nodes are observed to be repenting for their past misbehaviours. Therefore, these detection and reaction systems are also expected to accommodate compromised mobile nodes that have been reset.

- In the case of trust management system, all available, observable and classifiable evidence must be contributed towards the formulation of opinions. Unlike most of the related models, the trust model must be integrated with other models and/or systems for the collection of evidence. This includes the flow of information from secure routing protocols, IDS, cooperation based systems, *etc.* Such evidence must be used in identifying and excluding malicious nodes even before interacting with them. Similarly, the opinions formed from the collected evidence must be used in classifying the nodes and choosing them for sensitive communications. For example, selectively misbehaving nodes can be explicitly excluded in the case of sensitive communications and can be included in a non-sensitive communication, only if there is no other alternative route. As discussed in the previous section, the trust model must consider the notion of ignorance whilst establishing trust relationships and also when the relationships change as a result of unforeseen factors, such as, mobility. Furthermore, the trust model must resolve the issues related to recommendations (*i.e.* free-riding, honest-elicitation and recommender's bias) and take advantage of the second-hand information to identify malicious and benign nodes. Unlike most of the related models, the proposed approach must rest on a strong mathematical foundation for mapping evidence into opinions and also in measuring the trust. To meet the non-monotonic requirement of trust, the model must give importance to the recent evidence but not to the point that it over-writes the opinion. Furthermore, the evidence emerging from one-to-one experience must be taken at the highest priority in shaping the opinion, whilst the second-hand evidence plays the subsequent role. Finally, we

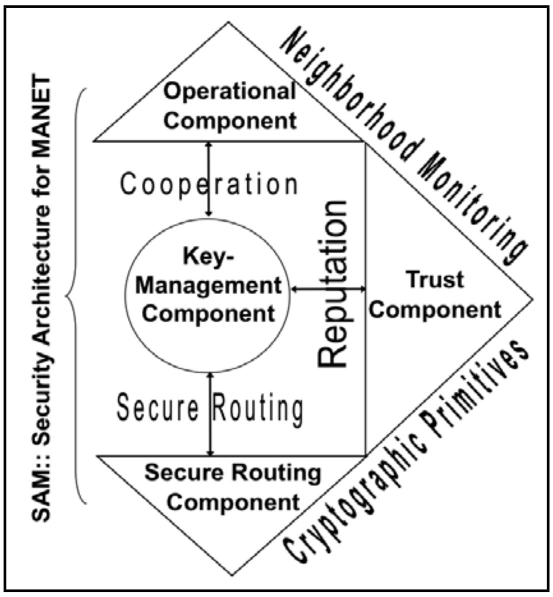


Figure 2.1: Components of Envisaged Trust Enhanced Security Architecture.

anticipate the trust model to incorporate a policy manager that can assist various components and/or systems of our architecture to make better decisions depending on the context and time. For example, secure routing protocol can piggyback trust model to enlist trusted intermediate nodes to the route, key management systems can resort to the trust model during key re-establishment, *etc.*

As shown in Figure 2.1, we identify the key management component as the centre of our architecture since it plays a significant role in bootstrapping authentication between mobile nodes. However, in a pure MANET such privilege may not be available and mobile nodes may rely on self-generated certificates to achieve the same goal. As expected, the secure routing component is tightly coupled with the key management and together form the prevention system of the architecture. The prevention system heavily relies on the cryptographic mechanisms to attain their objectives. On the other hand, the operational and trust components take advantage of the secure associations provided by the key management systems to authenticate other mobile nodes. These together form the detection and reaction system, in which the operational component focuses on achieving cooperation among the mobile nodes and therefore aims to produce a functional MANET. The trust component interfaces with all these components to collect evidence and then to assist these components in return to make better decisions. The detection and reaction system will be built upon a basic IDS, *i.e.* neighbourhood monitoring, to detect and gather evidence for malicious and benign behaviours.

In our design, we will interchangeably refer one-hop nodes as *neighbours* and the neighbourhood as *environment.* All the different types of MANET-oriented adversaries will be commonly referred to as *malicious nodes* and their behaviours as *misbehaviours.* We assume that a *bi-directional link* exists between mobile nodes and all the nodes in the network are equipped with an *omni-directional antenna.* Furthermore, the mobile nodes in our design are expected to exercise an *uniform transmission range*, although the mobility may vary from a node to node. In the following chapters, we will introduce and detail the design and working of our architecture.

2.9 Conclusion

In this chapter, we have successfully presented the impact of MANET's inherent features on the design of security systems and the different types of attacks that are prevalent in the same context. We have also established the basis for the design of security systems by introducing the fundamental concepts of security. Given that exhaustive research has been carried out in the security of the MANET, we focussed only on few well-studied approaches. To begin with, we discussed the impact of passive attacks and the solutions that have been proposed to render anonymous communications in the MANET. We then presented our study on prevention-based approaches, followed by detection-based approaches and reaction-based approaches. We have detailed some well-known secure routing protocols to iterate their significance in preventing multi-hop communications against conventional attacks and their reliance on robust key management systems to attain their objective. We then presented the incentive and game-theory based systems that focused on stimulating cooperation among nodes and, concluded the section with a brief discussion on their limitations and how they fail to complement the prevention-based systems. We have detailed the defence-in-depth approach (such as, including the IDS in the security design of MANET) and then presented the notion of reputation and trust management systems to address the shortcomings of prevention-based systems. We have established the context for trust management systems in the MANET by exploring the concepts and methodology in traditional trust management systems. Various trust models including recently proposed and few well-reviewed have been analysed and compared to study the state of trust management system in the MANET. We have presented the analysis of these trust models and discussed their limitations and shortcomings. Finally, we summarised our study in terms of security requirements that will be fulfilled in the design of our two-layered security system for the MANET.

3

Fellowship: Mitigating Packet Drop and Flooding Attacks

3.1 Introduction

Regardless of few fundamental challenges, secure routing protocols [21, 75, 76, 105, 120, 136, 150, 154, 155, 159, 161–164, 179, 180, 215, 250, 260, 282, 284, 337, 367, 397, 399, 439, 440, 445, 446, 455] only focus on securing a functional MANET. In other words, flooding and packet drop attacks that deter the availability of network services can successfully over-ride such secure routing protocols. Therefore it is apparent that the sole design of secure routing protocols (to keep unauthorised intermediate nodes at bay and to ensure the integrity of discovered routes) is inadequate to guarantee cooperation among mobile nodes. The fact that secure communications thrive only when a network is available clarifies well the severity of packet drop and flooding attacks, and therefore the urgency to enforce cooperation among mobile nodes. The severity of such attacks can further be realised from the dependency chain, where the success of secure routing protocols relies on a robust key management service, which, in turn, is defined by the self-organised feature of the MANET.

Several incentive-based payment systems [51, 77, 79, 102, 103, 127, 151, 209, 248, 278, 318, 320, 334, 349, 408, 416, 441, 443, 452, 453, 456] have been proposed to motivate cooperation among mobile nodes. As discussed in Chapter 2, these systems introduce incentives to motivate the mobile nodes to forward packets for other nodes and, in turn, demand the same mobile nodes to pay incentives to other nodes for propagating their packets. It appears that incentive-based payment systems shift the problem to another plane rather than solving the original problem. It is for this reason that these systems either rely on a tamper-proof service to protect and authenticate incentive tokens or struggle to manage the pricing and distribution of the incentive tokens. Proposals [18, 41, 91, 100, 166, 216, 385, 395, 438] that rely on an off-line or on-line CA to protect incentive tokens remain analogous with the issues that hinder the deployment of a robust key management service. Nevertheless, distributed on-line CAs does not change the state of the art significantly, because they, too, are restricted by the same issues for which the incentive-based payment systems have been designed. Although tamper-proof hardware emerges as a likely candidate to protect incentive tokens, it is unclear on how well it will fit to resolve the economic theory-based questions, such as, (a) *how are tokens priced* (b) *how are tokens distributed and made available across the network* (c) *how the deprivation of the tokens is prevented*, and so on. Even if incentive-based payment systems adopt a suitable economic model to realise the potential of tamper-proof hardware, it is too early to conclude that these systems would be the final. At least for this reason, incentive-based payment systems solve only the non-cooperation that results from packet drop behaviour, and only when such behaviour is exhibited by the selfish nodes. Furthermore, we agree with Huang *et al.*'s argument [165] that there is no need for an incentive system at all, especially at the early stages of the MANET, where excessive complexity may inhibit the development of the MANET.

Game-theory based systems [10, 14, 80, 172, 255, 257, 358, 359, 391, 457] have also been proposed to bring about cooperation among the mobile nodes. Although such approaches appear to be effective, their feasibility becomes questionable owing to their heavy reliance on the assumption that mobile nodes are rational. It is plausible to design mobile nodes that engage mobile users[1] in every rational decision; however, such an approach is discouraged as a viable solution because it would become a causal factor for security to be overlooked in the long-run with competitive functionality and usability. At least in the case of the unmanned MANETs, it is apparent that game-theory based systems are not the forerunners. Nevertheless, it is important to acknowledge the fact raised by game-theory based systems that packet droppers and/or flooders are the symbolic representation of rational attackers.

The emergence of IDS [13, 17, 104, 111, 213, 237, 251, 286, 288, 323, 342, 366, 371, 400, 411, 430] as a second-layer of defence has provided the platform for the development of various reputation and trust management systems. Given the fact that IDS only contributes to a detection vector, reputation systems step in to rate and classify the detections, whilst trust management systems take advantage of those reputation ratings and predefined security policies to make decisions for future events. Since reputation and trust management systems identify themselves as a *reactive layer* (in which they are expected to collect evidence from all security events to make best possible decision in the future), it is more appropriate to decouple them from preventive approaches. Such a modular approach has already proved its effectiveness; for example, secure routing and key management mechanisms independently focus on securing communications and managing security credentials, respectively. For this reason, we believe in designing a preventive layer that can enforce cooperation among the mobile nodes that can later be coupled with reputation and trust management systems to improve the overall security stance of the mobile nodes.

[1]Mobile users are likely to operate mobile devices in the case of commercial and civilian-based MANET deployments, although the same does not hold true in the case of unmanned deployments.

In this chapter, we propose an obligation-based mechanism known as the ***fellowship*** to enforce cooperation among the mobile nodes. The main focus of the fellowship is to explore and demonstrate the feasibility of mitigating packet drop and flooding attacks within the capability of the MANET. In other words, the approach targets complementing the self-organised feature of the MANET by not imposing an additional layer of communication for managing or exchanging information and, in parallel, keeping the packet droppers and flooders at bay. This contrasts with related models where an additional layer of communication exists to materialise cooperation among the mobile nodes. Furthermore, our approach is decentralised in the sense that it does not require any off-line or on-line CA for bootstrapping or administration. It also differs from distributed approaches [51, 77, 79, 102, 103, 255, 257, 435, 453, 456] that make the final decisions based on the consensus reached by the mobile nodes. With this perspective, the fellowship-enabled mobile nodes execute their instance of fellowship model that is totally independent of the execution of the fellowship instance at the other nodes.

In the following §3.2, we will present an attacker model. Section 3.3 will introduce and detail our fellowship model. In §3.4, we will demonstrate the effectiveness of the fellowship model against packet droppers and flooders and, thereby its role in motivating cooperation among mobile nodes (by means of detailed simulation results). We will then discuss some of the limitations of the fellowship model in §3.5. Section 3.6, then provides some concluding remarks and summarises the chapter.

3.2 Adversary Model

Our adversary model consists of two types of mobile nodes, *malicious* and *selfish*. Again, each of these nodes is capable of performing one of the following attacks, *packet drop* and *flooding*. Although the objectives of the malicious and selfish nodes may be different, their behaviours produce the same results. In the case of a packet drop attack, both malicious and selfish nodes are destined to take the role of an intermediate node to materialise the attack. On the other hand, they undertake the role of a source node to launch a flooding attack.

Malicious nodes primarily concentrate on disrupting the availability of the network services by draining the resources of other nodes (*through packet drop attacks*) or preventing other nodes from accessing the transmission channel (*through flooding attacks*). In particular, a class of malicious nodes can exhibit intermittent or selective attack behaviour; such malicious nodes are known as *selectively-misbehaving malicious nodes*. These nodes were noted for dropping packets that were received from a few specific node(s), while forwarding packets received from other mobile nodes. On the other hand, selectively-misbehaving malicious nodes can also exhibit time-dependent sporadic behaviour towards most of the nodes, and persistent behaviour towards a few selective node(s). They are also known for flooding packets to a few specific node(s) to block the availability of the wireless channel to those specific node(s). As noted earlier, selectively-misbehaving malicious nodes can also sporadically congest the channels of all neighbouring nodes. Recall that a flooding attack is different from a jamming attack, where the former either congests a node's bandwidth using unicast packets or an entire neighbourhood through broadcast packets, and the latter makes the wireless channel unavailable by over-powering the *Radio Frequency (RF)* signal at the physical layer.

Selfish nodes are unique to the MANET because of the heterogeneity of battery and computing resources among the mobile nodes. However, they echo the behaviour of malicious nodes by either dropping the packets that were received from other mobile nodes (for retaining their battery resource) or hijacking the transmission channel using their high battery resource (for propagating their high volume of self-generated packets). To hijack the transmission channel, selfish nodes skip the Medium Access Control (MAC) layer's contention resolution mechanism. Although selfish nodes are motivated to carry out their own communications, their total dependency on the network services implicitly reveals that their higher priority would be to avoid exclusion from the network. As seen from this perspective, selfish nodes differ from malicious nodes and hence it is apparent that they can be forced to comply with any preventive measure that is proposed to enforce cooperation among the mobile nodes. From here onwards, we will refer to both selfish and malicious nodes as *adversaries*, until a situation arises to resolve them independently.

3.3 Fellowship Model

Our model is designed to operate at the interface of the MAC and network layers so that it is independent of the routing protocols. It is founded on the principle that every mobile node in the network is *obliged* to contribute services toward the network to derive such services back from the network. Such an obligation not only enforces the mobile nodes to forward packets on behalf of the other nodes but also assures them to expect the same from those nodes for forwarding their packets. Since the obligation satisfactorily restricts the mobile nodes from over-riding the shared wireless channel, in return it guarantees the availability of the wireless channel to them. Mobile nodes that are not obliging with other nodes are progressively isolated from deriving further network services from those nodes. Since the obligation lays the groundwork for the mobile nodes to cooperate regardless of their objective, the model is referred to as *fellowship*.

3.3.1 Fundamental Concepts

Since fellowship is designed to enforce cooperation among the mobile nodes, it focuses on identifying the mobile nodes that fail to comply with its design. Depending on whether a mobile node absorbs an offered service or reciprocates with a similar service after using the offered service, it is feasible to classify the mobile node as either cooperating or non-cooperating. Therefore, the fellowship persuades the mobile nodes to commit a proportion of their resources for discovering and isolating adversaries with certainty. In other words, adversaries are trapped, based on the behaviour exhibited by them towards the offered service. Note that it is the same resource that acts as the causal factor for the heterogeneity among the mobile nodes and gives rise to a special class of adversary known as selfish nodes. More precisely, fellowship uses battery resources to discover the mobile nodes that forfeit their obligation towards the network.

Note that it is feasible to bring in any other resource or combination of resources as the measurement factor; however such an investigation is beyond the scope of this thesis. Although fellowship does not strictly focus on defining the proportion of the resources committed by a mobile node for other nodes in the network, it affirms that the payoff is proportional to the committed resource, that is, on how well the mobile node can be assured of the behaviour of those nodes in network. Such a design not only complements the resource heterogeneity of the mobile nodes, but also supports the self-organised feature of the MANET, especially in the *pure MANET*[2]. However, the same may not be true in the case of the *managed MANET*[3] where a mobile node's resource commitment for discovering the behaviour of other nodes can be dictated at a granular level. This is because the total count of the mobile nodes is most likely to be known ahead of deployment. Hence those mobile nodes can be either programmed to commit a proportion of their resource equally to all the remaining nodes or to prioritise their committed resource to specific nodes. However, for scenarios where incremental deployment is adopted, those mobile nodes can be directed to refine and define their resource commitment whenever new nodes become available in the network.

In addition, the fellowship's obligation-based approach is free from both the authenticity and integrity concerns that confine the related models, where control messages are the key factor for operation. Remember that related models evaluate packet droppers either by incrementing or decrementing their metrics based solely on the count of the packets forwarded or dropped by those adversaries. Also, recall from § 3.2 that selfish nodes look for possible vulnerabilities so that they can exploit or over-ride other nodes for network services, and alternatively aim to evade isolation and remain in the network. We envisage that such selfish nodes will comply with related models but will also exploit the defence by intelligently deriving more resources from other nodes. For instance, they can forward packets for other nodes and in return force those nodes to forward packets that have larger payloads. Such an imbalance can allow these selfish

[2]A pure network has no pre-established infrastructure and the network is created on the fly.

[3]A managed network excludes unauthorised nodes via pre-deployed keys or certificates, but does not eliminate authenticated peers that may be malignant.

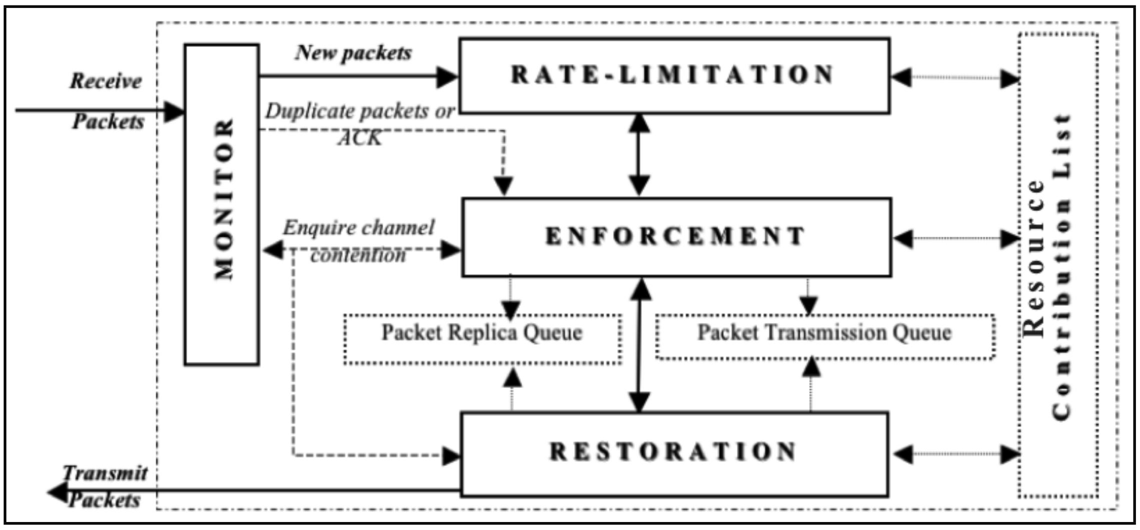


FIGURE 3.1: Components of Fellowship Model.

nodes to gain more network service than might otherwise have been gained by dropping packets that were received from those nodes. Note that the fellowship's approach for measuring the network services that have been used in terms of resources in turn allows them to defend against such intelligent exploitation.

3.3.2 Overview of Components

As shown in Figure 3.1, the fellowship comprises three components, *rate-limitation*, *enforcement*, and *restoration*. The rate-limitation component is responsible for defending against flooding attacks that are conducted by non-obliging nodes, that is, those mobile nodes that fail to share the wireless channel with neighbours. In such a condition, the rate-limitation component reduces the resource committed for those flooders in proportion to their flooded packets. The enforcement component is accountable for mitigating the packet drop attacks that are launched by the non-obliging nodes, that is, those mobile nodes that fail to forward packets for other nodes. It takes into account the past forwarding ratio of those non-obliging nodes, and the availability of shared wireless channel before reducing the resource committed for those packet droppers in proportion to the dropped packets. The enforcement component relies on an IDS-based monitor [248] to confirm whether a next-hop has forwarded packets on its behalf. The scenario where the mobile nodes are obliged to forward packets on behalf of the other nodes but are unable to do so owing to unforeseen conditions (such as a *hot-spot*[4] [375], and the overflow of the packet transmission queue), they extend their obligation towards the future packets that will be received from those nodes (provided the channel becomes available) with the help of the restoration component. In other words, the fellowship increments the resource commitment for those nodes in proportion to the packets that are discarded as a result of contention and congestion.

3.3.3 Data Structures

The data structures of fellowship are, the *resource-contribution list*, *packet-replica queue*, and *packet-transmission queue*. The following discusses the significance and role of these data structures.

For instance, the resource-contribution list of a fellowship-enabled node $\mathcal{A}$ is a container of multi-attributed tuples, where each of these tuples uniquely maps to one of the other known nodes in the network and is identified by the nodal-identity (such as, node $\mathcal{I}$). In each of these tuples, one of the attributes, known as the *contribution-share* (${}^{\mathcal{A}}\eta_{\mathcal{I}}$), holds the value of the resource committed by $\mathcal{A}$ for discovering the behaviour of the mapped node $\mathcal{I}$. We will present the significance of remaining attributes in the next §3.3.4. The following presents the approach adopted by $\mathcal{A}$ when arriving at ${}^{\mathcal{A}}\eta_{\mathcal{I}}$. As given in (3.1), if $E_{\mathcal{A}}(t_0)$ is the battery energy of $\mathcal{A}$ at the time of deployment (t_0), then $C_{\mathcal{A}-Self}$ presents the proportion of energy that $\mathcal{A}$ has dedicated for self operations (such as receiving packets for which its destination, transmitting own packets, mobility, reciprocating network services, *etc*), while $C_{\mathcal{A}-Contribution}$ denotes the proportion of energy that $\mathcal{A}$ has allocated for investigating the behaviour of the remaining nodes in the network. Assuming that $\mathcal{A}$ has discovered n other nodes in the network, then for every $\mathcal{I} \in n$ such that $\mathcal{I} \neq \mathcal{A}$, it populates ${}^{\mathcal{A}}\eta_{\mathcal{I}}$ with an allocated share of resource that is given by $(C_{\mathcal{A}-Contribution} \cdot E_{\mathcal{A}}(t_0))/n$. In the case of the pure MANET, ${}^{\mathcal{A}}\eta_{\mathcal{I}}$ may vary dynamically as $\mathcal{A}$ discovers more nodes over a period of time.

[4]Hotspots appear when excessive contention exists, prompting congestion when insufficient resources are available to handle the increased traffic load.

$$\begin{aligned} E_{\mathcal{A}}(t_0) &= \Big\{ \big[C_{\mathcal{A}-Self} \cdot E_{\mathcal{A}}(t_0)\big] + \big[C_{\mathcal{A}-Contribution} \cdot E_{\mathcal{A}}(t_0)\big] \Big\}; \\ 1 &= (C_{\mathcal{A}-Self} + C_{\mathcal{A}-Contribution}) \end{aligned} \tag{3.1}$$

Apart from the resource-contribution list, node $\mathcal{A}$ also uses a variable known as the *contribution-common* (χ) for managing its overall resource contribution to all the remaining nodes. The variable χ acts as a sink for penalised resources and a source for bonus resources. In other words, whenever $\mathcal{A}$ observes $\mathcal{I}$ to be a packet dropper or flooder, then the rate-limitation or enforcement components of its fellowship model reduces ${}^{\mathcal{A}}\eta_{\mathcal{I}}$ in proportion to the flooded or dropped packets, and sequentially increments χ with the reduced proportion. Similarly, whenever $\mathcal{A}$ has to discard $\mathcal{I}$'s packet owing to congestion, it then increments ${}^{\mathcal{A}}\eta_{\mathcal{I}}$ in proportion to the discarded packet, and sequentially decrements χ with the corresponding proportion. Next §3.3.4 presents a detailed discussion on how ${}^{\mathcal{A}}\eta_{\mathcal{I}}$ is managed, based on the notifications received for increment and decrement from the fellowship's components.

Node $\mathcal{A}$ stores recently-transmitted packets at the packet-replica queue, so that its enforcement component can use the queue to confirm whether a next-hop $\mathcal{J}$ has forwarded packets on its behalf. The duration for which the packets are retained at the packet-replica queue is proportional to the average time taken to discover a route, which is governed by the routing protocol. For instance, where the link-error notification is received that node $\mathcal{J}$ is unreachable, node $\mathcal{A}$'s enforcement component iterates through the packet-replica queue and purges those packets for which the next-hop is $\mathcal{J}$. Node $\mathcal{A}$'s enforcement component checks the availability of the shared wireless channel to determine the contention rate and the packet-transmission queue to determine the congestion rate, before deeming that the next-hop $\mathcal{J}$ is a packet dropper. The same approach is adopted by the restoration component before dispatching a packet. We will explain later that these are not the only checks that are performed by node $\mathcal{A}$'s enforcement component before deeming the next-hop $\mathcal{J}$ is a packet dropper.

3.3.4 System Design

In this section, we present a detailed description of the fellowship's components and then their combined operations. The operations are demonstrated from the perspective of a pure MANET with the anticipation that meeting the demands of a pure MANET will implicitly convene the requirements of a managed MANET.

3.3.4.1 Rate-limitation Component

The objective of a mobile node's rate-limitation component is to affirm that its neighbours oblige in sharing the wireless channel and should those neighbours fail to do so, it then exerts non-cooperation towards those neighbours in the future.

The fellowship's rate-limitation component at a mobile node is activated whenever the mobile node receives packets for forwarding on behalf of the other nodes. Depending on the contention rate deduced for the shared wireless channel and the congestion rate inferred from the packet transmission queue, the mobile node can specify an unanimous *reception-threshold* (τ) for the other nodes, that is, the rate at which it can receive packets for forwarding on behalf of those nodes in a given time interval. Given that the transmission rate of the other nodes is within τ, the mobile node's rate-limitation transfers those received packets to the enforcement component for forwarding. It is important to note that the mobile node's rate-limitation component disregards both the source and upstream route details for those received packets, regardless of being operated at the network layer. This is because the mobile node holds the upstream nodes as responsible for segregating or prioritising the packets before they request it to forward those packets on their behalf.

It is also possible for a mobile node to allocate a different τ for every other node in the network by taking into account the past behaviours of those nodes. Note that such a design would require the fellowship model to integrate with a reputation-based trust model so that its rate-limitation can take the feedback from the trust model and assign the node-specific τ. Chapters 4 and 5 present our trust model, which is then integrated with the fellowship to realise a trust enhanced defence against the packet droppers and flooders. Chapter 6 presents the integration of the fellowship with our trust model. Until then, let us assume that a fellowship-enabled mobile node $\mathcal{A}$ assigns a common τ for other nodes, and it is denoted as ${}^{\mathcal{A}}\tau_{\mathcal{I}}$ for node $\mathcal{I}$. However, as discussed below, there are instances where a mobile node would assign a specific ${}^{\mathcal{A}}\tau_{\mathcal{I}}$ for node $\mathcal{I}$ even in the absence of integration with a trust model.

Although the self-organised feature of the MANET allows a mobile node $\mathcal{A}$ to opt for a subjective ${}^{\mathcal{A}}\tau_{\mathcal{I}}$ based upon the contention and congestion values, it is unfair to expect node $\mathcal{I}$ to be aware of ${}^{\mathcal{A}}\tau_{\mathcal{I}}$ in the absence of any explicit notification. Therefore, node $\mathcal{A}$'s rate-limitation introduces another threshold known as the *negotiation threshold* (ρ) through which it indirectly notifies node $\mathcal{I}$ or, alternatively, enables node $\mathcal{I}$ to adaptively learn the value of ${}^{\mathcal{A}}\tau_{\mathcal{I}}$. Whenever node $\mathcal{I}$'s packet transmission rate exceeds ${}^{\mathcal{A}}\tau_{\mathcal{I}}$, node $\mathcal{A}$ then discards those packets that exceed ${}^{\mathcal{A}}\tau_{\mathcal{I}}$ during the given time interval. However at the next interval, node $\mathcal{A}$ continues to accept $\mathcal{I}$'s packets for forwarding, provided those packets are within ${}^{\mathcal{A}}\tau_{\mathcal{I}}$. This indirect signalling operation continues for ρ intervals, so that node $\mathcal{I}$ can infer node $\mathcal{A}$'s ${}^{\mathcal{A}}\tau_{\mathcal{I}}$ and synchronise its packet transmission rate with ${}^{\mathcal{A}}\tau_{\mathcal{I}}$. Note that the approach allows node $\mathcal{I}$ to infer the value of ${}^{\mathcal{A}}\tau_{\mathcal{I}}$ even in the absence of a CA and the lack of an exchange of additional control packets. Let us now perceive on how node $\mathcal{I}$ responds to node $\mathcal{A}$'s advertised ${}^{\mathcal{A}}\tau_{\mathcal{I}}$, provided node $\mathcal{I}$ has fellowship enabled, and alternatively how it reacts if it is one of those selfish or malicious nodes.

3.3.4.1.1 Respondent – Fellowship-enabled Mobile Node Assuming node $\mathcal{I}$ to be benign, in which case it must have the fellowship enabled, it is expected to trace its packet transmission rate with the help of its enforcement component. Note that the enforcement component interfaces with an IDS-based monitor to promiscuously capture node $\mathcal{A}$'s transmissions. Since node $\mathcal{I}$ would retain a copy of all transmitted packets at the packet-replica queue for a duration equal to route discovery, its enforcement component would be able to verify those packets that have been forwarded on its behalf by node $\mathcal{A}$ and, similarly those packets that have been discarded by node $\mathcal{A}$. From this information, node $\mathcal{I}$ would then be able to infer node $\mathcal{A}$'s packet reception rate (node $\mathcal{A}$'s ${}^{\mathcal{A}}\tau_{\mathcal{I}}$ for node $\mathcal{I}$), and tailor its packet transmission rate according to ${}^{\mathcal{A}}\tau_{\mathcal{I}}$. To begin with, even if node $\mathcal{I}$'s enforcement component fails to trace the rate of packets that were forwarded and subsequently discarded by node $\mathcal{A}$ as a result of collisions at the physical interface, the continual exhibition of this operation by node $\mathcal{A}$ for ρ intervals assures that node $\mathcal{I}$ can successfully infer ${}^{\mathcal{A}}\tau_{\mathcal{I}}$.

In addition, node $\mathcal{I}$ records node $\mathcal{A}$'s reception rate (${}^{\mathcal{A}}\tau_{\mathcal{I}}$) as the *reciprocate reception* (${}^{\mathcal{I}}\tau_{\mathcal{A}}^{recp}$) in its contribution-list as another attribute together with ${}^{\mathcal{I}}\eta_{\mathcal{A}}$ for $\mathcal{A}$. This ensures that node $\mathcal{I}$ reciprocates with the same reception rate (${}^{\mathcal{I}}\tau_{\mathcal{A}} = {}^{\mathcal{I}}\tau_{\mathcal{A}}^{recp} = {}^{\mathcal{A}}\tau_{\mathcal{I}}$) when forwarding packets on the behalf of $\mathcal{A}$ in the future, and therefore restricts $\mathcal{A}$ from gaining any momentum by lowering ${}^{\mathcal{A}}\tau_{\mathcal{I}}$. Regardless of node $\mathcal{I}$'s uniform τ for other nodes (including for $\mathcal{A}$), it would resort to ${}^{\mathcal{I}}\tau_{\mathcal{A}}^{recp}$ to reciprocate $\mathcal{A}$'s ${}^{\mathcal{A}}\tau_{\mathcal{I}}$. Therefore as noted earlier, mobile nodes do not necessarily allocate a uniform τ for the other nodes even in the absence of integration with a trust model. However if channel contention and congestion would prevent node $\mathcal{I}$ from honouring ${}^{\mathcal{I}}\tau_{\mathcal{A}} = {}^{\mathcal{I}}\tau_{\mathcal{A}}^{recp}$ for node $\mathcal{A}$ (and the same also holds true for the requests received from the other nodes), then node $\mathcal{I}$ would resort to a common τ that can be lower than ${}^{\mathcal{I}}\tau_{\mathcal{A}}^{recp}$ (the same applies for other nodes too). Observe that node $\mathcal{I}$ would maintain a constant value for ρ intervals for all nodes, unlike the reception rate that is likely to change from one node to another, depending on the rate offered by those nodes. Nevertheless, our design does not restrict an implementation from tailoring the value of ρ based on the behaviour or relationship held with other nodes.

We defer our discussion on how node $\mathcal{I}$'s enforcement component semantically differentiates node $\mathcal{A}$'s *packet discards* from the *packet drops* to §3.3.4.2. Although discarding a packet is syntactically the same as dropping a packet, it is differentiated in our model, based on the semantics whether a node drops a packet owing to unforeseen conditions such as the contention and congestion or for other reasons, such as for saving the battery resource and/or disrupting the network services.

3.3.4.1.2 Respondent – Malicious Node Unlike the previous scenario, considering node $\mathcal{I}$ to be of type malicious leaves us with the probability that $\mathcal{I}$ is likely to disregard $\mathcal{A}$'s ${}^{\mathcal{A}}\tau_{\mathcal{I}}$. This is because malicious nodes are engineered to disrupt the availability of the shared wireless channel. For instance, a malicious node $\mathcal{I}$ can be expected to flood $\mathcal{A}$ by means of unicast packets with an aim of propagating the impact of the flooding to node $\mathcal{A}$'s downstream nodes. In this case, node $\mathcal{I}$'s transmission rate is supposed to exceed $\mathcal{A}$'s ${}^{\mathcal{A}}\tau_{\mathcal{I}}$ for more than ρ intervals; otherwise node $\mathcal{I}$ would not be behaving much different from a fellowship-enabled node to substantiate its maliciousness. Recall that node $\mathcal{A}$ would discard all those packets that exceed ${}^{\mathcal{A}}\tau_{\mathcal{I}}$ during every interval for up to ρ intervals, but ensuring the transference of packets that comply ${}^{\mathcal{A}}\tau_{\mathcal{I}}$ within every interval to the enforcement component. Once node $\mathcal{A}$ realises that the indirect negotiation with node $\mathcal{I}$ has failed, it takes the stance that $\mathcal{I}$ is an adversary that is targeting to disrupt its access to the shared wireless channel. Therefore as a preventive measure, node $\mathcal{A}$ continues to discard $\mathcal{I}$'s packets that exceed ${}^{\mathcal{A}}\tau_{\mathcal{I}}$ at every interval following the ρ^{th} interval. In addition, node $\mathcal{A}$ decrements its ${}^{\mathcal{A}}\eta_{\mathcal{I}}$ for $\mathcal{I}$ for every discarded packet (P_k) by using a *demerit-factor* (λ) and the energy ($E(P_k)$) spent in receiving P_k. In parallel, node $\mathcal{A}$ increments χ with the proportion of resource decremented from ${}^{\mathcal{A}}\eta_{\mathcal{I}}$. All these are given in (3.2),

$$\begin{aligned} {}^{\mathcal{A}}\eta_{\mathcal{I}}(t_{a+1}) &= \Big\{ {}^{\mathcal{A}}\eta_{\mathcal{I}}(t_a) - \big[\lambda \cdot E_{\mathcal{A}}(P_k)\big] \Big\}; \quad \lambda > 1; \\ \chi(t_{a+1}) &= \Big\{ \chi(t_a) + \big[\lambda \cdot E_{\mathcal{A}}(P_k)\big] \Big\}; \quad t_{a+1} > t_a; \end{aligned} \tag{3.2}$$

Thereafter, node $\mathcal{A}$ continues to discard those packets that exceed ${}^{\mathcal{A}}\tau_{\mathcal{I}}$ during every interval until node $\mathcal{I}$ tunes its packet transmission rate to $\mathcal{A}$'s ${}^{\mathcal{A}}\tau_{\mathcal{I}}$. Since node $\mathcal{A}$ decrements ${}^{\mathcal{A}}\eta_{\mathcal{I}}$ in proportion to the amount of discarded packets, it is possible for ${}^{\mathcal{A}}\eta_{\mathcal{I}}$ to run into a negative value. In such a situation, node $\mathcal{A}$ records the timestamp in a variable known as the *blacklist timestamp* (${}^{\mathcal{A}}T_{\mathcal{I}}$), which is another attribute of the contribution-list. Although the rate-limitation takes the necessary measure to forward node $\mathcal{I}$'s packets that are received within ${}^{\mathcal{A}}\tau_{\mathcal{I}}$ and hence transfers those packets to the enforcement component, the latter makes the final decision on whether to forward those packets on the behalf of node $\mathcal{I}$ by looking into ${}^{\mathcal{A}}\eta_{\mathcal{I}}$. As we will explain later in detail, the enforcement component prevents node $\mathcal{A}$ from forwarding packets on behalf of node $\mathcal{I}$ when ${}^{\mathcal{A}}\eta_{\mathcal{I}}$ is negative. In summary, the combined effect of node $\mathcal{A}$ to discard packets that exceed ${}^{\mathcal{A}}\tau_{\mathcal{I}}$ and to decrement ${}^{\mathcal{A}}\eta_{\mathcal{I}}$ for every discarded packet ensures that node $\mathcal{I}$ obliges to ${}^{\mathcal{A}}\tau_{\mathcal{I}}$. Otherwise, node $\mathcal{I}$ would be draining all its energy in flooding, where node $\mathcal{A}$ will be discarding those packets that exceed ${}^{\mathcal{A}}\tau_{\mathcal{I}}$ following the ρ^{th} interval or all packets once ${}^{\mathcal{A}}\eta_{\mathcal{I}}$ becomes negative. Not only node $\mathcal{A}$ prevents the impact of flooding from being propagated beyond a hop, that is, by containing the flooded packets at the point of origin, but also isolates node $\mathcal{I}$ for the exhibited behaviour. Given that mobile nodes may be compromised and used to perform such malicious actions and that later they may be redeemed, the design allows an infinitesimal growth of ${}^{\mathcal{A}}\eta_{\mathcal{I}}$ as given in (3.3). The time period for which node $\mathcal{I}$ remains benign with node $\mathcal{A}$ determines the growth of ${}^{\mathcal{A}}\eta_{\mathcal{I}}$ into positive, and any sign of flooding during the transition is adequate to push node $\mathcal{I}$ to reinvent the cycle.

$$\begin{aligned} {}^{\mathcal{A}}\eta_{\mathcal{I}}(t_{a+1}) = &\left[e^{t\delta} + {}^{\mathcal{A}}\eta_{\mathcal{I}}({}^{\mathcal{A}}T_{\mathcal{I}})\right]; \qquad 10^{-6} < \delta < 10^{-9}; \\ &\exists t \in T,\ \big(t = t_{a+1} - {}^{\mathcal{A}}T_{\mathcal{I}}\big) \wedge \big(t_{a+1} > {}^{\mathcal{A}}T_{\mathcal{I}}\big) \wedge (\mathcal{I} \neq malicious)\,; \end{aligned} \tag{3.3}$$

Therefore, it is feasible to anticipate node $\mathcal{I}$'s compliance with node $\mathcal{A}$'s ${}^{\mathcal{A}}\tau_{\mathcal{I}}$, but to behave different from a fellowship-enabled node by persistently throttling $\mathcal{A}$'s bandwidth with a packet transmission rate equal to ${}^{\mathcal{A}}\tau_{\mathcal{I}}$. We argue that node $\mathcal{I}$ cannot take much leverage from the above scenario, because ${}^{\mathcal{A}}\tau_{\mathcal{I}}$ is dependent on various factors, (a) *node $\mathcal{A}$'s contention and congestion rate*, (b) ${}^{\mathcal{A}}\tau_{\mathcal{I}}^{recp}$, *i.e. ${}^{\mathcal{I}}\tau_{\mathcal{A}}$ offered by node $\mathcal{I}$ towards $\mathcal{A}$ in the past*; even if there was no interaction between $\mathcal{A}$ and $\mathcal{I}$ in the past, future response of $\mathcal{I}$ is adequate to initialise ${}^{\mathcal{A}}\tau_{\mathcal{I}}^{recp}$, (c) *the influence of total count of $\mathcal{A}$'s neighbours in determining ${}^{\mathcal{A}}\tau_{\mathcal{I}}$, which is discussed below*, and (d) *node $\mathcal{I}$'s packet dropping behaviour that influences ${}^{\mathcal{A}}\eta_{\mathcal{I}}$ and is explained in detail in §3.3.4.1.3.*

So far we have seen only one category of malicious behaviour, where a malicious node overwhelms a specific mobile node and therefore the downstream nodes with unicast packets. In the next category, a malicious node adopts a similar approach but targets its entire neighbourhood with broadcast packets with an aim to block a section of network. Nevertheless, fellowship-enabled mobile nodes (that are positioned in the malicious node's environment) activate their rate-limitation component on receiving the broadcast packets and proactively prevent the propagation of those flooded broadcast packets. Although these fellowship-enabled nodes take independent measures to mitigate the propagation of flooded broadcasts, their indirect synchronisation not only isolate the malicious node but also causes an unproductive battery drain and therefore a shorter lifetime for the malicious node. In the case of selective flooding, the only difference that can be observed at a fellowship-enabled node is the time taken by its rate-limitation to isolate the selectively flooding malicious node, where the isolation delay primarily results from the malicious node's selective behaviour and not from the fellowship's design.

3.3.4.1.3 Respondent – Selfish Node Let us now perceive node $\mathcal{I}$ as being selfish with the objective of understanding the possible response that a selfish node would exhibit against the fellowship node's reception rate (node $\mathcal{A}$'s ${}^{\mathcal{A}}\tau_{\mathcal{I}}$ in this scenario). The selfish node $\mathcal{I}$ is expected to be highly resourced (owing to the rationale that it hijacks the shared wireless channel from its neighbours) to disseminate a high volume of packets by boycotting the MAC's contention resolution mechanism. Note that node $\mathcal{I}$ would be making the maximum benefit from the imbalanced environments, for instance, a neighbourhood that is occupied by the less-resourced nodes, such as *Personal Digital Assistants (PDAs)*, and IP-enabled cellular devices.

Since selfish nodes are tailored to exploit network services but at the same time to do their best to avoid being isolated, node $\mathcal{I}$'s behaviour is anticipated to fit in between the behaviour spectrum of the fellowship-enabled and the malicious nodes. Therefore node $\mathcal{I}$ is aware of the consequence of exceeding ${}^{\mathcal{A}}\tau_{\mathcal{I}}$, because it not only witnesses those packets that exceed ${}^{\mathcal{A}}\tau_{\mathcal{I}}$ being discarded by node $\mathcal{A}$, but also receives a penalty proportional to those flooded packets, which will thereafter affect its future requests to node $\mathcal{A}$. As a result, node $\mathcal{I}$ is expected to throttle node $\mathcal{A}$ only with a packet transmission rate that equals to ${}^{\mathcal{A}}\tau_{\mathcal{I}}$, especially after learning node $\mathcal{A}$'s reception rate within ρ^{th} interval. In other words, node $\mathcal{I}$ is enforced to comply with ${}^{\mathcal{A}}\tau_{\mathcal{I}}$ to avoid from being detected as selfish by node $\mathcal{A}$. As explained earlier, node $\mathcal{I}$ cannot influence node $\mathcal{A}$'s choice of ${}^{\mathcal{A}}\tau_{\mathcal{I}}$ unless it had lured $\mathcal{A}$ by contributing a high ${}^{\mathcal{I}}\tau_{\mathcal{A}}$ in the past, which is likely to contradict with its objective. Even offering a high ${}^{\mathcal{I}}\tau_{\mathcal{A}}$ is inadequate for node $\mathcal{I}$ to be assured that node $\mathcal{A}$ would reciprocate with the same reception rate ($i.e.$ ${}^{\mathcal{A}}\tau_{\mathcal{I}} = {}^{\mathcal{A}}\tau_{\mathcal{I}}^{recp} = {}^{\mathcal{I}}\tau_{\mathcal{A}}$) for the reasons mentioned earlier.

In summary, selfish nodes are pushed to imitate either the behaviour of a fellowship-enabled or a malicious node. They are likely to imitate the former, given that no mileage is derived from imitating the behaviour of a malicious node. The same argument holds true for selectively misbehaving selfish nodes.

3.3.4.1.4 Selection of Reception-Threshold (τ) In this section, we demonstrate how the fellowship's design can be extended to enable a mobile node to determine the value for τ depending on the number of neighbours in the environment and the channel bandwidth that is available in the absence of contention and congestion. Recall that mobile nodes are capable of detecting the total count of active neighbours at any point in time from the requests received for forwarding packets, recent link-layer acknowledgements, and fresh reports received for broken links. The advantage of this design is that it allows a mobile node, for example node $\mathcal{A}$, to make use of the available bandwidth more efficiently, especially when the environment is packed with either high or low density of neighbours. For instance, node $\mathcal{A}$ would be able to honour node $\mathcal{I}$'s ${}^{\mathcal{I}}\tau_{\mathcal{A}}$ with ${}^{\mathcal{A}}\tau_{\mathcal{I}}^{recp}$ at a low neighbour density (or equally split the bandwidth for its neighbours at high node density). Therefore the possible values that node $\mathcal{A}$'s ${}^{\mathcal{A}}\tau_{\mathcal{I}}$ can take for node $\mathcal{I}$ are, $0 \leqslant {}^{\mathcal{A}}\tau_{\mathcal{I}} \leqslant Bandwidth(\mathcal{A})$. At one extremity, node $\mathcal{A}$ may be completely clogged with a congested packet transmission queue, thus preventing it from rendering any service to node $\mathcal{I}$. At the other extremity, it may be able to render its entire share of channel for forwarding node $\mathcal{I}$'s packets.

Although we assume a one-to-one mapping between the MAC and the IP addresses in pure MANETs, it is feasible for mobile nodes to spoof the MAC addresses in pure

MANETs where the appropriate mechanism is absent for preventing nodes from spoofing their identity. We include some optional policies to the fellowship's rate-limitation to minimise the effect caused by the adversaries that would randomly change their MAC address and therefore their IP address. The first policy allows rate-limitation to separate the neighbours into two categories, *known* and *unknown*. The policy then restricts the rate-limitation to assigning a minimum reception rate ($\tau > 0$) to unknown or new neighbours, and to advocate a progressive increase in τ for those neighbours depending on the factors influencing the choice of τ. The policy is based on the rationale that known or old neighbours are privileged to reap a high τ for their demonstrated benign behaviour, while new neighbours are expected to demonstrate their behaviour to earn a high τ. The second policy imposes rate-limitation to restrict the *count of new neighbours* (ϕ) considered at any interval. Given the limited acceptance of ϕ and restricted τ for each new neighbour, malicious and selfish nodes are disadvantaged from spoofing their identity. Even if those adversaries opt to spoof their identity, the advantage that they can derive is extremely constrained by the additional policies of fellowship and thereof mitigating the impacts of flooding.

3.3.4.2 Enforcement Component

The objective of a mobile node's enforcement component is to affirm that its neighbours oblige in forwarding its requests and if those neighbours fail to cooperate then it exerts non-cooperation towards those neighbours in the future.

On receiving a packet from rate-limitation component, a fellowship-enabled node's enforcement component first checks the resource that it has committed for the previous-hop, and then checks the resource committed for the next-hop, before making the move to forward the packet on behalf of the previous-hop. Once the packet is deemed for transmission, the enforcement component transfers the packet to the restoration component, which then checks for the availability of the wireless channel and the compliance of its packet transmission rate with the next-hop's reception-rate before dispatching the packet to the wireless medium. For instance, node $\mathcal{A}$'s enforcement checks whether its ${}^{\mathcal{A}}\eta_{\mathcal{I}}$ for the previous-hop $\mathcal{I}$ and ${}^{\mathcal{A}}\eta_{\mathcal{J}}$ for the next-hop $\mathcal{J}$ are *positive*, before determining to forward a packet on behalf of the previous-hop $\mathcal{I}$.

At node $\mathcal{A}$, PFR is an important factor that influences the resource value (${}^{\mathcal{A}}\eta_{\mathcal{J}}$) committed for next-hop $\mathcal{J}$. As shown in (3.4), the PFR (${}^{\mathcal{A}}\gamma_{\mathcal{J}}$) of node $\mathcal{J}$ from the perspective of node $\mathcal{A}$ is defined as the ratio of total count of packets forwarded by node $\mathcal{J}$ on the behalf of node $\mathcal{A}$ (${}^{\mathcal{A}}\alpha_{\mathcal{J}}$) to the total count of packets transmitted to node $\mathcal{J}$ by node $\mathcal{A}$ (${}^{\mathcal{A}}\beta_{\mathcal{J}}$).

$$ {}^{\mathcal{A}}\gamma_{\mathcal{J}} = \left(\frac{{}^{\mathcal{A}}\alpha_{\mathcal{J}}}{{}^{\mathcal{A}}\beta_{\mathcal{J}}} \right) \tag{3.4} $$

Assuming ${}^{\mathcal{A}}\eta_{\mathcal{I}}$ and ${}^{\mathcal{A}}\eta_{\mathcal{J}}$ are positive, the enforcement component transfers the packet to the restoration component. Once the restoration component transmits the packet to the next-hop $\mathcal{J}$ (provided the shared wireless channel is free from contention and also its packet transmission is in compliance with node $\mathcal{J}$'s reception rate, ${}^{\mathcal{J}}\tau_{\mathcal{A}}$), it increments ${}^{\mathcal{A}}\beta_{\mathcal{J}}$ to reflect the transmission. As explained below, the enforcement component relies on an IDS-based monitor to verify whether the next-hop $\mathcal{J}$ has forwarded a packet on its behalf, and if the verification turns out to be successful, it then increments ${}^{\mathcal{A}}\alpha_{\mathcal{J}}$ accordingly.

Let us now summarise the attributes of the contribution-list and the parameters of our model. Table 3.1 presents node $\mathcal{A}$'s tuple for node $\mathcal{I}$, which is extracted from its contribution-list, and then Table 3.2 lists the general parameters that are used in fellowship model.

3.3.4.2.1 Packet Dropping or Discarding Next-hop Let us investigate the scenario where node $\mathcal{A}$ reads the next-hop $\mathcal{J}$ as failing to forward packets on its behalf. The root causes for node $\mathcal{A}$ to read node $\mathcal{J}$ as a packet dropper are, (a) *node $\mathcal{A}$ might have experienced a collision at the physical interface that would have prevented it from reading $\mathcal{J}$'s transmission*, (b) *node $\mathcal{J}$ might have experienced channel contention and therefore it would have discarded incoming packets owing to the congested packet transmission queue*, (c) *node $\mathcal{A}$ might have exceeded node $\mathcal{J}$'s ${}^{\mathcal{J}}\tau_{\mathcal{A}}$ and therefore node $\mathcal{J}$ would have discarded those packets that were received from node $\mathcal{A}$ as explained in*

Table 3.1: Instance of Contribution-list's Attributes

Contribution Share	**Reception Threshold**	**Reciprocate Reception**	**Blacklist Timestamp**	**Packets Forwarded on Behalf**	**Transmitted Packets**	**Packet Forwarding Ratio**
${}^{\mathcal{A}}\eta_{\mathcal{I}}$	${}^{\mathcal{A}}\tau_{\mathcal{I}}$	${}^{\mathcal{A}}\tau_{\mathcal{I}}^{\mathbf{recp}}$	${}^{\mathcal{A}}\mathbf{T}_{\mathcal{I}}$	${}^{\mathcal{A}}\alpha_{\mathcal{I}}$	${}^{\mathcal{A}}\beta_{\mathcal{I}}$	${}^{\mathcal{A}}\gamma_{\mathcal{I}}$

Table 3.2: Parameters for Fellowship Model

Parameters	**Description**
Negotiation Threshold (ρ)	Maximum interval for indirectly notifying reception threshold (τ).
Demerit Factor (λ)	Penalty awarded for every flooded or dropped packet.
Merit Factor (σ)	Credit for discarded packet (due to congestion).
Contribution Common (χ)	Sink or source for penalised or awarded resources.
Transmission Threshold (ν)	Cut-off point to mark a neighbour as packet dropper.
Neighbour Count (ϕ)	Total count of neighbours known at any given point.

§*3.3.4.1.1* or, (d) *node $\mathcal{J}$ might have been an adversary that was dropping $\mathcal{A}$'s packets to drain $\mathcal{A}$'s battery resources.* Although node $\mathcal{A}$ can determine that its packet might have been dropped by the next-hop, it adopts a progressive approach to determine the root-cause of the packet drop owing to the above-mentioned ambiguities. Therefore, node $\mathcal{A}$ takes a two-level of approach to confirm whether or not the next-hop $\mathcal{J}$ is not cooperating with its requests, that is, a packet dropper. Similar to the rate-limitation's combined deployment of ${}^{\mathcal{A}}\tau_{\mathcal{I}}$ and ρ to detect whether previous-hop $\mathcal{I}$ is flooding or not, the enforcement component relies on ${}^{\mathcal{A}}\gamma_{\mathcal{J}}$ and the *transmission threshold* (ν) to detect whether or not the next-hop $\mathcal{J}$ is a packet dropper. In other words, $\mathcal{A}$ deems a next-hop $\mathcal{J}$ as a packet dropper whenever its ${}^{\mathcal{A}}\gamma_{\mathcal{J}}$ for $\mathcal{J}$ drops below ν. Although, node $\mathcal{A}$ assumes a uniform ν for every other node in the network, it can be tailored to assign a unique value to node $\mathcal{J}$ (${}^{\mathcal{A}}\nu_{\mathcal{J}}$) and the same can be applied for other nodes. Nevertheless, node $\mathcal{A}$ considers a uniform ν for every other node in this scenario for the ease of explanation.

For all the above-mentioned cases, node $\mathcal{A}$ proceeds to increment ${}^{\mathcal{A}}\beta_{\mathcal{J}}$, but not ${}^{\mathcal{A}}\alpha_{\mathcal{J}}$, regardless of the root-cause, because node $\mathcal{A}$ never deems the next-hop $\mathcal{J}$ as a packet dropper until its ${}^{\mathcal{A}}\gamma_{\mathcal{J}}$ for $\mathcal{J}$ drops below ν. Here ν engulfs the uncertainty factors (collision, contention and congestion, and node $\mathcal{J}$'s negotiation of ${}^{\mathcal{J}}\tau_{\mathcal{A}}$) that account for the discarded packets. For example, if $\nu = x$ such that $0 < x < 1$ then x presents the transmission rate, while $(1 - x)$ accounts for the factors that inhibit transmission.

3.3.4.2.1.1 Cases A and B In the first two cases, although node $\mathcal{A}$ is uncertain of whether or not node $\mathcal{J}$ has forwarded a packet on its behalf at the time of collision, it fails to increment ${}^{\mathcal{A}}\alpha_{\mathcal{J}}$ to discourage adversaries from taking advantage of such a loophole for dropping packets. Similarly, node $\mathcal{A}$ fails to increment ${}^{\mathcal{A}}\alpha_{\mathcal{J}}$ when node $\mathcal{J}$ discards its packets as a result of contention and congestion. In spite of the fact that node $\mathcal{A}$ has not been communicating with node $\mathcal{J}$'s congestion, it infers the channel contention and therefore node $\mathcal{J}$'s congested packet transmission queue. It performs the inference by sampling the total number of packets that were dissipated by node $\mathcal{J}$ per interval. In this situation, the enforcement component advocates the restoration component to reduce the rate of packets that will be forwarded to the next-hop $\mathcal{J}$. Note that node $\mathcal{A}$'s restoration component can be modelled to reduce its transmission rate by imitating the traditional contention window concept. In both of the above cases, node $\mathcal{A}$ purges the copy of the packet that was stored in the packet-replica queue.

A special category for the second case results when the next-hop $\mathcal{J}$ fails to forward packets on the behalf of node $\mathcal{A}$ because its downstream neighbour is a packet dropper. Node $\mathcal{A}$ can infer such a situation by sampling the next-hop $\mathcal{J}$'s transmissions with the help of an IDS-based monitor to check whether $\mathcal{J}$ is dropping all its requests or only discarding requests that pertain to a particular flow. The former remains synonymous with the fourth case that will be discussed below, while the latter points out the next-hop $\mathcal{J}$'s relationship with its downstream neighbour. In this situation, node $\mathcal{A}$'s enforcement component prevents the restoration component from transmitting further packets that belong to the identified flow because those packets will be dropped at the next-hop $\mathcal{J}$. This not only allows node $\mathcal{A}$ to save its battery resources, but also permits it to propagate the information indirectly back to the source; the same approach is adopted at all upstream nodes. Such an approach would also allow the source to switch to an alternate path for communication with the destination. An alternative design to this sub-case is to make the intermediate nodes trigger a route error message back to the source whenever they deem their next-hop is a packet dropper. Although the propagation of a route error would be elegant, it again opens the door for adversaries to launch a further disruption to the packet flow.

3.3.4.2.1.2 Case C The third case points out where node $\mathcal{A}$ overwhelms the next-hop $\mathcal{J}$ with a packet transmission rate that exceeds node $\mathcal{J}$'s packet reception rate ${}^{\mathcal{J}}\tau_{\mathcal{A}}$. Node $\mathcal{A}$ infers the advertised ${}^{\mathcal{J}}\tau_{\mathcal{A}}$ by overhearing node $\mathcal{J}$'s transmissions via an IDS-based monitor. From overheard packets, node $\mathcal{A}$'s enforcement component learns that the next-hop $\mathcal{J}$ has forwarded only a fixed proportion of packets (*i.e.* ${}^{\mathcal{J}}\tau_{\mathcal{A}}$) on its behalf, but has discarded the remaining proportion during every interval. Therefore, the enforcement component instructs the restoration component to equalise its packet transmission rate to ${}^{\mathcal{J}}\tau_{\mathcal{A}}$. Consequently, it also records the value of ${}^{\mathcal{J}}\tau_{\mathcal{A}}$ at ${}^{\mathcal{A}}\tau_{\mathcal{J}}^{recp}$ so that it can reciprocate with the same reception rate to $\mathcal{J}$'s future requests. Similar to the first two cases, the enforcement component restrains ${}^{\mathcal{A}}\alpha_{\mathcal{J}}$ from being incremented for every overwhelmed packet that was discarded by the next-hop $\mathcal{J}$, and also purges the duplicate packet that is stored in its packet-replica queue. However, it increments ${}^{\mathcal{A}}\alpha_{\mathcal{J}}$ for every packet that was forwarded on its behalf by the next-hop $\mathcal{J}$. In other words, the enforcement component does its best to infer the information available from within its disposal to learn whether it has overwhelmed the next-hop $\mathcal{J}$ before deeming $\mathcal{J}$ as a packet dropper.

3.3.4.2.1.3 Case D The final case presents the scenario where the next-hop $\mathcal{J}$ resembles an adversary that either saves its battery resources or drains the battery resources of the previous-hop by dropping the received packets. Analogous to the other three cases, node $\mathcal{A}$'s enforcement component not only investigates the environment's contention for shared wireless but also matches with the next-hop $\mathcal{J}$'s advertised ${}^{\mathcal{J}}\tau_{\mathcal{A}}$. Although it has not been mentioned earlier, the enforcement component employs the following steps in all cases. Whenever the next-hop $\mathcal{J}$ is noted for dropping a packet, the enforcement component not only refrains from incrementing ${}^{\mathcal{A}}\alpha_{\mathcal{J}}$ and purging the copy of the duplicate packet from its packet replica queue, but it also checks whether ${}^{\mathcal{A}}\gamma_{\mathcal{J}}$ is greater than ν. For instance, when node $\mathcal{A}$'s ${}^{\mathcal{A}}\gamma_{\mathcal{J}}$ for the next-hop $\mathcal{J}$ fails to exceed ν, the enforcement component proceeds to reduce ${}^{\mathcal{A}}\eta_{\mathcal{J}}$ that is given in (3.2) by λ and the energy $(E_{\mathcal{A}}(P_k))$ spent in transmitting the packet that is later dropped by $\mathcal{J}$. Consequently, node $\mathcal{A}$ increments χ with the proportion of the resource decremented for the dropped packet. Given that node $\mathcal{A}$ has clarified the possibility of accidental factors for packet loss (such as of packet collision, contention for shared wireless channel, and the transmission of packets at a rate greater than next-hop $\mathcal{J}$'s ${}^{\mathcal{J}}\tau_{\mathcal{A}}$) by using ν and also has observed continuous non-cooperation behaviour from the next-hop $\mathcal{J}$, it then proceeds to prevent further packet loss by reducing the resource committed (${}^{\mathcal{A}}\eta_{\mathcal{J}}$) for $\mathcal{J}$. This not only protects node $\mathcal{A}$ from transmitting further packets to next-hop $\mathcal{J}$ but also restrains it from cooperating to $\mathcal{J}$'s future requests.

3.3.4.2.1.4 Discussion Let us investigate the robustness of the enforcement component against the different types of adversaries. As noted above, the persistent malicious next-hop $\mathcal{J}$ is promptly identified for dropping packets and, accordingly, is isolated from playing the role of next-hop to node $\mathcal{A}$'s future requests or vice versa. However, the same may not hold true if $\mathcal{J}$ is a selfish node because it is motivated to save battery resources and also designed to rely on the cooperation from other nodes to capitalise on the saved battery resources for its communications. Therefore, the enforcement component forces the selfish node $\mathcal{J}$ to cooperate or accept the risk of being isolated. In the case of selective misbehaviours, detection is proportional to the rate of misbehaviour and ν. Although such selective misbehaviour cannot deter communications to the extent of persistent misbehaviour, it can disrupt sensitive communications. The integration of the fellowship with the trust model (which will be discussed in Chapter 6) can overcome such a situation and ensure a reliable path for sensitive communications, provided there exists a reliable path in the network. Recall that ν determines the cut-off when the next-hop $\mathcal{J}$ is suspected as a packer dropper and then a negative ${}^{\mathcal{A}}\eta_{\mathcal{J}}$ marks the isolation of $\mathcal{J}$.

3.3.4.2.2 Requestor – Adversary Let us now examine a scenario that is the opposite to the scenario presented in §3.3.4.2.1. Node $\mathcal{A}$'s enforcement component discards a packet that is received from its previous-hop neighbour $\mathcal{I}$, whenever its ${}^{\mathcal{A}}\eta_{\mathcal{I}}$ for $\mathcal{I}$ reads negative. Let us investigate the factors that would cause ${}^{\mathcal{A}}\eta_{\mathcal{I}}$ to take a negative value, (a) *node $\mathcal{I}$ must have been flooding that would have caused the rate-limitation to reduce ${}^{\mathcal{A}}\eta_{\mathcal{I}}$ in proportion to those flooded packets* and/or, (b) *node $\mathcal{I}$ must have been dropping $\mathcal{A}$'s packets that would have caused the enforcement to reduce ${}^{\mathcal{A}}\eta_{\mathcal{I}}$ in proportion to those dropped packets.*

Node $\mathcal{A}$ justifies its decision to discard those packets that were received from the previous-hop $\mathcal{I}$ because it satisfactorily confirms $\mathcal{I}$'s flooding and/or packet dropping behaviour before deeming $\mathcal{I}$ as a flooder and/or packer dropper. Note that the decision is the represented by the assignment of negative value to ${}^{\mathcal{A}}\eta_{\mathcal{I}}$. Recall that node $\mathcal{A}$ would have obtained the first-level of confirmation for flooding when node $\mathcal{I}$ had failed to comply with ${}^{\mathcal{A}}\tau_{\mathcal{I}}$, which was advertised for ρ intervals. The second-level of confirmation for flooding would have been obtained when $\mathcal{I}$'s packet transmission rate persistently exceeded ${}^{\mathcal{A}}\tau_{\mathcal{I}}$ and, therefore, received a penalty for those flooded packets (those packets that exceeded ${}^{\mathcal{A}}\tau_{\mathcal{I}}$ in every interval following ρ^{th} interval). Similarly, node $\mathcal{A}$ would have obtained a first-level of confirmation for packet-dropping when $\mathcal{I}$ had failed to comply with ν. The second-level of confirmation for packet-dropping would have been obtained when $\mathcal{I}$'s ${}^{\mathcal{A}}\gamma_{\mathcal{I}}$ persistently exceeded ν.

3.3.4.3 Restoration Component

The objective of a mobile node's restoration component is to contribute further resources towards a neighbour's future request whenever the neighbour's current packet is discarded as a result of an unforeseen condition, such as the contention and congestion.

On receiving a packet from the enforcement component, the restoration component is responsible for dispatching the packet to the shared wireless channel. The restoration component first investigates the packet transmission queue to learn about the congestion level and then the *Network Allocation Vector's (NAV)* value [375] to understand the availability of the shared wireless channel. Given that the packet transmission queue is free from congestion and the shared wireless channel is void of contention, the restoration component then transmits the packet to the next-hop $\mathcal{J}$ and sequentially increments ${}^{\mathcal{A}}\beta_{\mathcal{J}}$. In addition, it ensures that the packets are transmitted within the reception rate (${}^{\mathcal{J}}\tau_{\mathcal{A}}$) of the next-hop $\mathcal{J}$ and also stacks a copy of those packets at the packet-replica queue to aid the enforcement component's future investigation. An exception to the stacking process occurs whenever a next-hop evolves as the final destination for a packet. Note that in such situation, the packet will never be propagated further. Optionally, the restoration component is designed to increase the packet transmission rate after every ρ^{th} interval, provided the environment is free from contention and the next-hop $\mathcal{J}$ maintains ${}^{\mathcal{J}}\tau_{\mathcal{A}}$. In other words, the design enables the mobile nodes to request (or offer) additional service from (or for) other mobile nodes whenever the situation is conducive for placing such requests (or offers). Therefore, node $\mathcal{A}$ makes an indirect appeal to the next-hop $\mathcal{J}$ requesting an increase in the reception rate (${}^{\mathcal{J}}\tau_{\mathcal{A}}$) and, henceforth reciprocates with the same reception rate (${}^{\mathcal{A}}\tau_{\mathcal{J}}^{rep}$) to $\mathcal{J}$'s requests in the future. For some reason, if the next-hop $\mathcal{J}$ fails to honour node $\mathcal{A}$'s request, then node $\mathcal{A}$ withholds from incrementing ${}^{\mathcal{A}}\beta_{\mathcal{J}}$ for those packets that had exceeded (${}^{\mathcal{J}}\tau_{\mathcal{A}}$) following ρ^{th} interval. This ensures that node $\mathcal{A}$ precisely captures the next-hop $\mathcal{J}$'s packet forwarding behaviour (${}^{\mathcal{A}}\gamma_{\mathcal{J}}$) without any noise. Another option is to enable the restoration component to provide feedback to the trust model regarding reliable next-hops so that the information is included by the trust model for influencing the routing protocol while selecting the next-hop for the routing packets. We will discuss the design of such an approach in Chapter 6 where we integrate the fellowship model with our trust model, to be presented in the next Chapter 4. Orthogonally, the packet transmitted by restoration component can undergo collision with one of the neighbour's transmissions. Such a condition is beyond the control of the fellowship and, therefore, the design leaves the resolution to the routing protocol.

Contrary to the above scenario, if the packet transmission queue is reported as congested or a high contention exists for the shared wireless channel, then the restoration component discards the packet (P_k) and increases ${}^{\mathcal{A}}\eta_{\mathcal{I}}$ for the previous-hop $\mathcal{I}$ as given in (3.5). The increased resource for the packet owner $\mathcal{I}$ is defined by the *merit factor* (σ) and the energy ($E_{\mathcal{A}}(P_k)$) that would have been spent by node $\mathcal{A}$ in transmitting P_k. Consequently, node $\mathcal{A}$ reduces χ with the proportion of the resource increased for the sake of the discarded packet (P_k). In the fellowship, $\sigma < \lambda$, to ensure that any awarded resource does not mask past misbehaviours.

$$\begin{aligned} {}^{\mathcal{A}}\eta_{\mathcal{I}}(t_{a+1}) &= \Big\{ {}^{\mathcal{A}}\eta_{\mathcal{I}}(t_a) + \big[\sigma \cdot E_{\mathcal{A}}(P_k)\big] \Big\}; \quad \sigma > 1; \\ \chi(t_{a+1}) &= \Big\{ \chi(t_a) - \big[\sigma \cdot E_{\mathcal{A}}(P_k)\big] \Big\}; \quad t_{a+1} > t_a; \end{aligned} \tag{3.5}$$

In sequence, the restoration component notifies the rate-limitation component to reduce the reception rate (${}^{\mathcal{A}}\tau_{\mathcal{I}}$) for the previous-hop $\mathcal{I}$ in proportion to the congestion and contention. Node $\mathcal{A}$ is likely to experience contention whenever heavy traffic flows through it and/or the surrounding environment. Such a scenario prevails when node $\mathcal{A}$ happens to be geographically positioned at the centre of the network. In such case, the rate of the incoming requests exceeds the rate of the outgoing packets owing to the contention and, thereafter, the received packets find themselves congested at the packet transmission queue. Although the incoming packets are discarded because of the congested packet transmission queue, the packets that are already stacked in the transmission queue will be retained for dispatch until one of two events occurs, (a) *the expiry of those packets* or (b) *the channel is free from contention.*

In the above design, node $\mathcal{A}$'s restoration component could have been designed to invoke a control message to the previous-hop $\mathcal{I}$ informing it of the contention and congestion, and therefore requesting a reduced packet transmission rate. However, such an approach does come with issues. First, the dissemination of the addition control packet to each previous-hop neighbour according to each incoming flow may further aggravate the contention and congestion. Second, the dissemination of an additional control packet may also open the door to other types of DoS attacks.

3.3.5 System Operation

In this section, we will summarise the interaction of the fellowship's components using the scenario shown in Figure 3.2. Since we are not interested at the flow level, we focus our discussion only on the node-to-node interactions, in particular, the interactions of the fellowship-enabled node A with its neighbours $\mathcal{X}$, $\mathcal{I}$, $\mathcal{Y}$, $\mathcal{N}$, $\mathcal{J}$, and $\mathcal{M}$. Here,

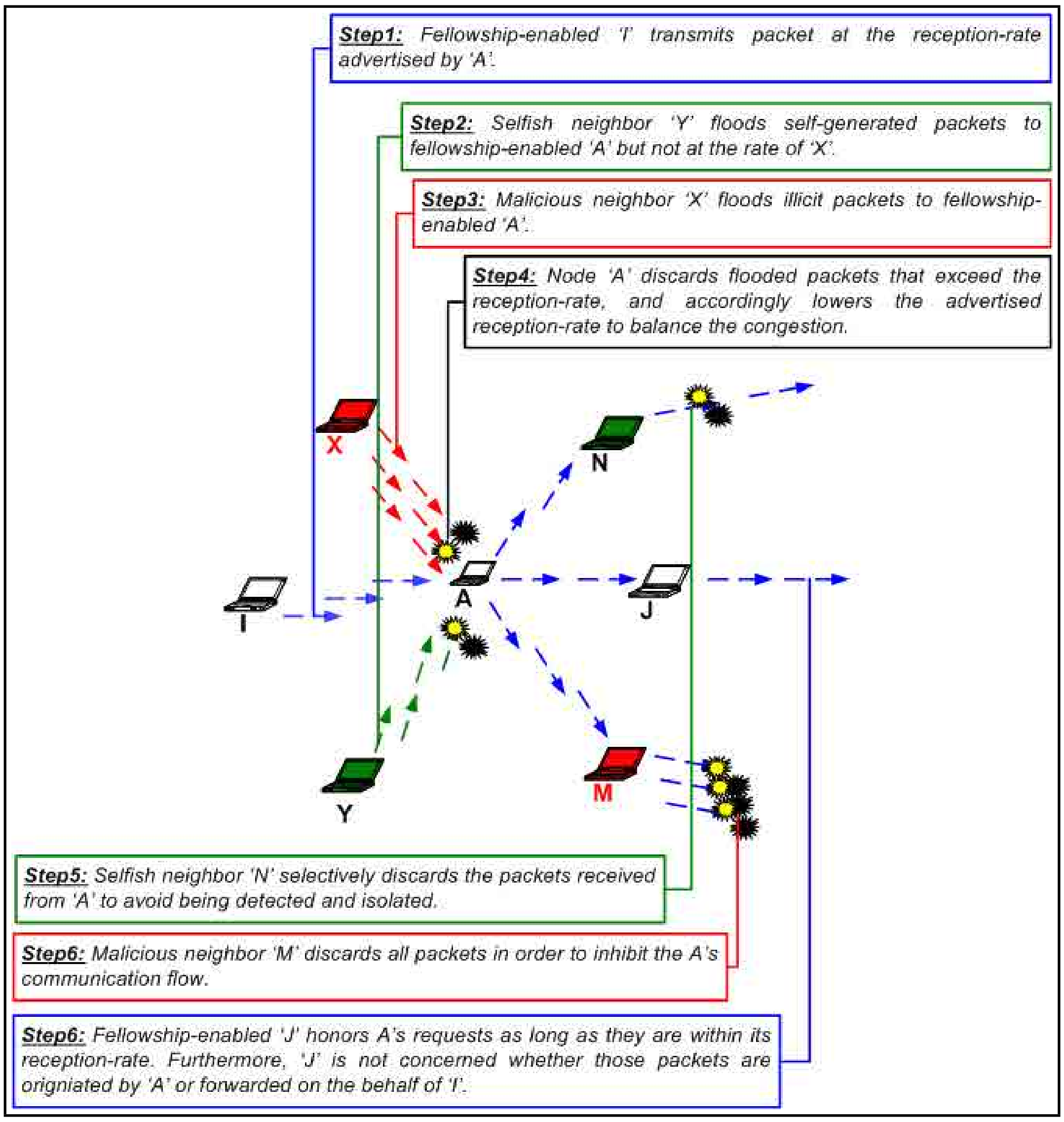

Step1: Fellowship-enabled 'I' transmits packet at the reception-rate advertised by 'A'.
Step2: Selfish neighbor 'Y' floods self-generated packets to fellowship-enabled 'A' but not at the rate of 'X'.
Step3: Malicious neighbor 'X' floods illicit packets to fellowship-enabled 'A'.
Step4: Node 'A' discards flooded packets that exceed the reception-rate, and accordingly lowers the advertised reception-rate to balance the congestion.
X
N
A
I
J
Y
M
Step5: Selfish neighbor 'N' selectively discards the packets received from 'A' to avoid being detected and isolated.
Step6: Malicious neighbor 'M' discards all packets in order to inhibit the A's communication flow.
Step6: Fellowship-enabled 'J' honors A's requests as long as they are within its reception-rate. Furthermore, 'J' is not concerned whether those packets are orignaited by 'A' or forwarded on the behalf of 'I'.

Figure 3.2: Interaction among Fellowship, Malicious and Selfish Nodes.

nodes $\mathcal{X}$, $\mathcal{I}$, and $\mathcal{Y}$ are the previous-hop neighbours to $\mathcal{A}$ and the nodes $\mathcal{N}$, $\mathcal{J}$, $\mathcal{M}$ are the next-hop neighbours to $\mathcal{A}$. Among these neighbours, nodes $\mathcal{I}$ and $\mathcal{J}$ are assumed to have the fellowship-enabled while the remaining neighbours belong to the adversary category. Among those adversaries, $\mathcal{X}$ and $\mathcal{M}$ exhibit malicious behaviour, whereas $\mathcal{Y}$ and $\mathcal{N}$ demonstrate selfish behaviour.

Let us now focus on the operation of node $\mathcal{A}$'s rate-limitation to the incoming requests. As expected, node $\mathcal{A}$ reciprocates with ${}^{\mathcal{A}}\tau_{\mathcal{I}} = {}^{\mathcal{A}}\tau_{\mathcal{I}}^{recp} = {}^{\mathcal{I}}\tau_{\mathcal{A}}$ to honour the reception rate (${}^{\mathcal{I}}\tau_{\mathcal{A}}$) offered by node $\mathcal{I}$ in the past. Moreover, node $\mathcal{I}$ must be aware of the fact that node $\mathcal{A}$ will reciprocate the reception rate. Given that nodes $\mathcal{A}$ and $\mathcal{I}$ have been fellowship-enabled, node $\mathcal{I}$ adheres to the protocol specification and transmits the packets to node $\mathcal{A}$ at the rate, ${}^{\mathcal{I}}\tau_{\mathcal{A}} = {}^{\mathcal{I}}\tau_{\mathcal{A}}^{recp}$. Therefore, node $\mathcal{A}$'s rate-limitation transfers the node $\mathcal{I}$'s packets to its enforcement component, which is again expected to transfer those packets to the restoration component. This is based on the assumption that node $\mathcal{A}$'s ${}^{\mathcal{A}}\eta_{\mathcal{I}}$ for node $\mathcal{I}$ must have been positive, and also making another assumption that node $\mathcal{A}$ has a positive contribution share for a next-hop (or downstream node). The restoration component dispatches those packets to the respective next-hop, provided the channel is available for transmission. Assuming the channel is free from contention and the packets are transmitted to the fellowship-enabled next-hop $\mathcal{J}$, the previous-hop $\mathcal{I}$ can now confirm that node $\mathcal{A}$ has forwarded packets on its behalf, and the same for node $\mathcal{A}$ when node $\mathcal{J}$ later forwards those packets to its next-hop.

Although it is not feasible to discuss all the possible combinations that would prevent the occurrence of the above scenario, most of the common factors that would influence the above scenario are, (a) *nodes $\mathcal{A}$ and $\mathcal{I}$ might not have interacted in the past*, (b) *either node $\mathcal{A}$ or $\mathcal{I}$ experiences a high volume of packets or heavy contention and congestion*, (c) *node $\mathcal{A}$ may encounter a malicious or selfish next-hop for the packets that were received from node $\mathcal{I}$* or (d) *either node $\mathcal{A}$ may increase ${}^{\mathcal{A}}\tau_{\mathcal{I}}$ or node $\mathcal{I}$ may request for an increase in ${}^{\mathcal{A}}\tau_{\mathcal{I}}$*. Let us now briefly consider each of these cases.

3.3.5.1 Case A

The instance where nodes $\mathcal{A}$ and $\mathcal{I}$ have never interacted would, in turn, mean that ${}^{\mathcal{A}}\tau_{\mathcal{I}}$ is set to an initial value that is likely to be the product of node $\mathcal{A}$'s share of the channel bandwidth, and its contention and congestion rate. Since node $\mathcal{I}$ would be reciprocating the same (${}^{\mathcal{I}}\tau_{\mathcal{A}}^{rep} = {}^{\mathcal{A}}\tau_{\mathcal{I}}$) to node $\mathcal{A}$'s future requests, there is no incentive for node $\mathcal{A}$ to manipulate with ${}^{\mathcal{A}}\tau_{\mathcal{I}}$.

3.3.5.2 Case B

In the case of heavy traffic flowing into node $\mathcal{A}$ (which is also possible, if node $\mathcal{A}$ forwards packets for other neighbours such as $\mathcal{X}$ and $\mathcal{Y}$), it is expected to reduce ${}^{\mathcal{A}}\tau_{\mathcal{I}}$ to balance the service that is offered to the requesting neighbours. As pointed out earlier, node $\mathcal{I}$ would be reciprocating the same (${}^{\mathcal{I}}\tau_{\mathcal{A}}^{rep} = {}^{\mathcal{A}}\tau_{\mathcal{I}}$) to node $\mathcal{A}$'s future requests. Remember this model can be overridden by introducing the concept of priority among the packet flows, where node $\mathcal{A}$ can be designed to prioritise node $\mathcal{I}$'s packets over packets received from other neighbours, and therefore keeping ${}^{\mathcal{A}}\tau_{\mathcal{I}}$ free from reduction. Although the investigation for prioritising packet flows is beyond the scope of this thesis, it is important to point out that such QoS-based packet flow models can be complemented using the trust relationships identified by the trust models.

Regardless of whether the QoS-based packet flow is in place or not, node $\mathcal{A}$ is expected to lower ${}^{\mathcal{A}}\tau_{\mathcal{I}}$ for node $\mathcal{I}$ at the time of the contention and congestion. As expected, node $\mathcal{I}$ complies with the advertised ${}^{\mathcal{A}}\tau_{\mathcal{I}}$ and reciprocates the same to node $\mathcal{A}$'s future requests. Node $\mathcal{A}$ is likely to experience a congested state if $\mathcal{X}$ or $\mathcal{Y}$ floods unicast or broadcast packets within its environment. Although the flooding inhibits node $\mathcal{A}$ from assuring the already advertised ${}^{\mathcal{A}}\tau_{\mathcal{I}}$ to node $\mathcal{I}$, node $\mathcal{A}$ ensures that the flooding is restrained within a hop and therefore is not propagated to the rest of network. As explained in the previous §3.3.4, node $\mathcal{A}$ would forward packets on the behalf of $\mathcal{X}$ (or $\mathcal{Y}$) as long as those packets are within ${}^{\mathcal{A}}\tau_{\mathcal{X}}$ (or ${}^{\mathcal{A}}\tau_{\mathcal{Y}}$) and ${}^{\mathcal{A}}\eta_{\mathcal{X}}$ (or ${}^{\mathcal{A}}\eta_{\mathcal{Y}}$) is positive. Regardless of whether $\mathcal{X}$ exhibits a pertinent or selective malicious behaviour, it is likely to expend the battery resource by flooding unicast or broadcast packets within the environment of node $\mathcal{A}$. In the case of the broadcast packets being disseminated by $\mathcal{X}$, node $\mathcal{I}$ participates in restricting those flooded broadcasts from being propagated across the network. Alternatively, node $\mathcal{Y}$ may comply with ${}^{\mathcal{A}}\tau_{\mathcal{Y}}$ to avoid isolation by node $\mathcal{A}$.

3.3.5.3 Case C

Similar to the previous case, node $\mathcal{A}$ may be encountering an adversary as the next-hop for those packets that were received from node $\mathcal{I}$. The adversaries $\mathcal{M}$ and $\mathcal{N}$ are expected to drop those packets that were received from node $\mathcal{A}$. However, the malicious $\mathcal{M}$ drops those packets that were received from node $\mathcal{A}$ more aggressively than the selfish node $\mathcal{N}$ owing to its malignant behaviour. On the other hand, the selfish node $\mathcal{N}$ is likely to comply with node $\mathcal{A}$ to avoid isolation. Node $\mathcal{A}$ reduces ${}^{\mathcal{A}}\eta_{\mathcal{M}}$ for $\mathcal{M}$ as ${}^{\mathcal{A}}\gamma_{\mathcal{M}}$ becomes lower than ν, which would, in turn, lead to a negative ${}^{\mathcal{A}}\eta_{\mathcal{M}}$. This would cause node $\mathcal{A}$ to hold back its transmissions to node $\mathcal{M}$ and, in turn, convey the information indirectly back to node $\mathcal{I}$. As explained in the previous §3.3.4, node $\mathcal{I}$ would be able to sample node $\mathcal{A}$'s transmissions, and learn that $\mathcal{A}$ is able to transmit packets for other mobile nodes and also on its behalf when the next downstream node is other than $\mathcal{M}$.

3.3.5.4 Case D

This case represents a scenario where the interaction between nodes $\mathcal{I}$ and $\mathcal{A}$ are neither affected nor influenced by any adversary or high volume of traffic. In such a situation, it is reasonable for node $\mathcal{A}$ to increase ${}^{\mathcal{A}}\tau_{\mathcal{I}}$ depending on the availability of the channel bandwidth with the expectation that node $\mathcal{I}$ would reciprocate the same in the future. Alternatively, node $\mathcal{I}$ can place an indirect request to node $\mathcal{A}$ following the ρ^{th} interval. It is then left to node $\mathcal{A}$ to honour or dishonour node $\mathcal{I}$'s request. This scenario can be more precisely modelled with the integration of the trust model, where node $\mathcal{I}$'s past responses can be captured to make a better decision.

Next §3.4 introduces the simulator, the tools that were used for analysing the simulation results, scenarios that were used to study the impact of adversaries and then the simulation results that demonstrate the effectiveness of the fellowship model against adversaries.

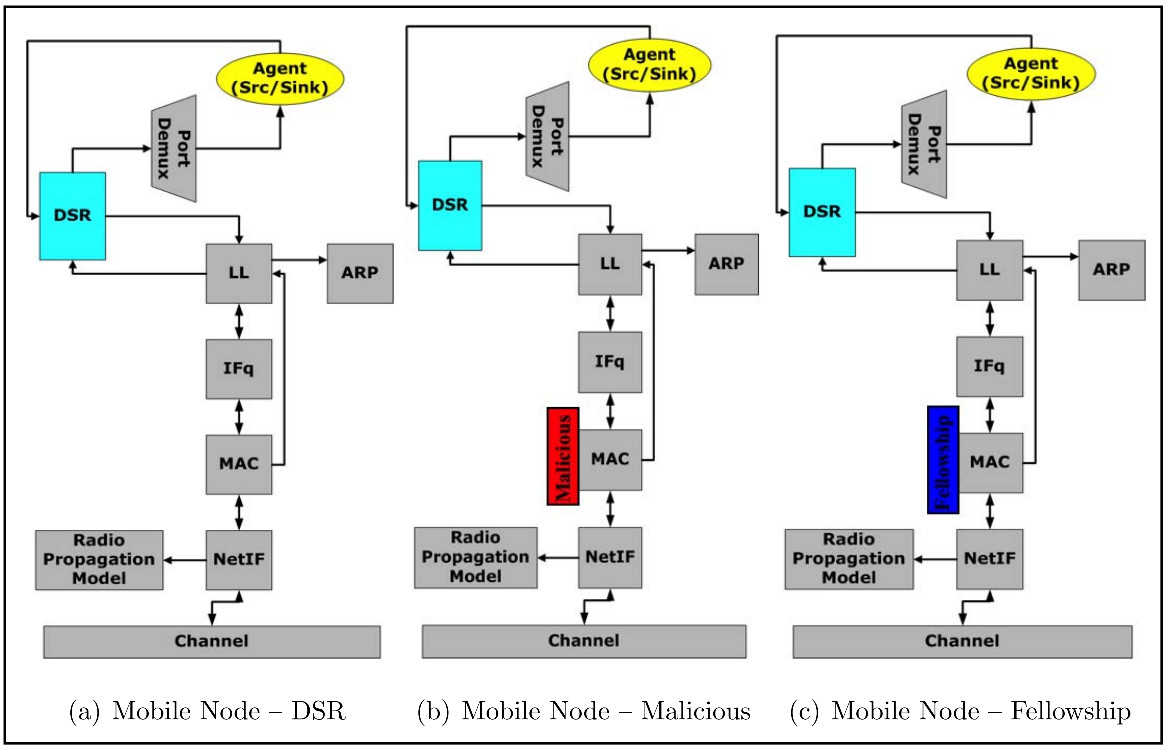


FIGURE 3.3: Constituents and Extensions to a Mobile Node.

3.4 Simulation

The NS2 [316] is an open source tool that supports the simulation of various traffic sources, transport, routing and MAC protocols, and the queue management mechanisms over wired and wireless networks. We have used the NS2 to investigate the performance of the fellowship-enabled mobile nodes against the malicious nodes for the following reasons, (a) *the NS2 contains the implementation of various standard MANET protocols, such as 802.11 MAC, DSR and AODV*, (b) *it has been widely used to study the performance of those MANET protocols* and, (c) *used for the analysis of various security models in the MANET.*

3.4.1 Simulator Overview

The NS2 has been built on two languages, *C++*, and the *OTcl interpreter* (an object oriented extension of Tcl), where the object oriented discrete event simulator is written in C++, and the simulation scripts are executed through the OTcl interpreter.

The simulation scripts and simulator are tightly coupled as a result of the one-to-one correspondence between the compiled class hierarchy of the C++ and the interpreted hierarchy of the OTcl. The compiled class hierarchy contains the detailed definition and operation of various protocols for both wired and wireless extensions. The NS2 requires an extension of an existing protocol or implementation of a new protocol to take place at the compiled class hierarchy. Furthermore, the extension or implementation at the compiled class hierarchy of the C++ reduces the packet and event processing time, and thereby enhances the simulation efficiency. We have implemented the fellowship and adversary models as an extension of the pre-packaged 802.11 MAC at the C++ layer. On the other hand, the OTcl interpreter is used to define, configure and setup the routing protocol, application-based traffic, network topology, simulation scenarios and parameters for the MAC 802.11, fellowship and adversary models. This suits for our implementation well, because the OTcl interpreter allows rapid changes to the simulation setup and scenarios despite its tardiness. Given that the NS2 is a single-threaded discrete event simulator, the results are repeatable, provided the simulation setup and scenarios including the node configuration, traffic generation pattern and the randomisation are reproduced.

In the next §3.4.1.1, we will briefly present the constituents of a mobile node (Figure 3.3(a)) that forms the core of the NS2's wireless extension. A detailed description of the mobile node's constituents and their operation is available at [316]. In our simulations, a mobile node without a fellowship model but with only the DSR as the routing protocol and the 802.11 MAC in between the network and physical layers is referred to as the *DSR-MAC node*. The adversary model is implemented as an inbuilt module between the interfaces of the MAC 802.11 and the routing protocol, where it can be set or unset from the OTcl interpreter using the simulation scripts. A mobile node with the adversary code activated is known as a *malicious node* (Figure 3.3(b)). Fellowship is implemented as a separate set of wrapper C++ classes and called depending on whether it is activated from the OTcl interpreter's scripts, and a fellowship-enabled mobile node is referred to as the *fellowship node* (Figure 3.3(c)).

3.4.1.1 Mobile Node

A mobile node is similar to a node used for wired networks except that it has added functionalities for mobility, and the capability to transmit and receive packets on a wireless channel. The constituents of a mobile node with DSR as the routing protocol are, (a) *agent module*, (b) *port classifier*, (c) *routing protocol*, (d) *link layer*, (e) *Address Resolution Protocol (ARP)*, (f) *interface queue*, (g) *MAC layer*, (h) *network interface*, (i) *radio propagation model* and, (j) *antenna.*

- The agent module is responsible for the generation and reception of the network-layer packets and therefore acts as the endpoints of a communication flow.

- A range of routing protocols including DSR, *Destination Sequence Distance Vector (DSDV)* [406], TORA [198], and AODV is available for the mobile node. There is no address classifier in the case of the DSR and, hence, the DSR becomes the sink for all received packets. The DSR handles the packets for which it is the destination by either passing those packets to the agent module via the port classifier (in the case of data packets) or replying with a route reply (in the case of route request). Given that a mobile node is positioned as an intermediary in the route, the DSR then forwards the received packets along the route mentioned by those packets. Alternatively, the DSR handles the data packets that are received from the co-existing agent module by either forwarding those packets to next the mobile node as per the route or initiating a new route discovery.

- The link layer of the mobile node receives the outgoing packets from the DSR and passes them to the *interface queue (IFQ)* below. In the opposite direction, the link layer collects the incoming packets from the MAC layer and pipes them back to the DSR. The link layer resolves the IP address to the MAC address by employing the ARP module.

- For each outgoing packet, the link layer refers to the ARP to insert the destination's MAC address. For the instance, where the destination's MAC address is unknown, the ARP broadcasts the destination's IP address to discover the MAC address.

- The interface queue prioritises the routing protocol's packets, that is, the DSR's packets over the packets generated by the agent module and also supports a filter to remove packets that contain a specific destination address. The interface queue's priority scheduling is implemented over the *First-In-First-Out (FIFO) scheduling* and the *drop-on-overflow buffer management* that are typical to most of the current Internet routers.

- We have chosen the *IEEE 802.11 Distributed Coordination Function (DCF)* [279] as the MAC layer protocol for the mobile nodes. It sends out the *Request-To-Send (RTS)* and waits for the *Clear-To-Send (CTS)* packets before unicasting the data packets and receiving the corresponding *acknowledgement (ACK)* packets. Note that in the case of broadcasts, it directly sends out the packets. It detects the medium's availability by using both the *physical carrier sense* and the NAV that is also known as the *virtual carrier sense*. Also, the DCF permits the promiscuous tapping of the packets so that they can be passed on to the DSR module for analysis.

- The network interface is the hardware interface to the wireless channel. It marks an outgoing packet with information such as the *transmission power*. It handles packets that are transmitted by other interfaces with the help of the radio propagation model. The incoming packets are regarded as collision-free by the propagation model depending on their transmission power. In the NS2, a mobile node's network interface represents the *Direct Sequence Spread Spectrum (DSSS)* of a Lucent Wave LANs network adapter card.

- The radio propagation model uses the *Friss-space attenuation* at proximity and the *two-ray ground propagation model* to cover a farther distance.

- Finally, the mobile nodes consider the omni-directional antenna with unity gain for receptions and transmissions.

3.4.2 Additional Tools

We have used the Rational Rose Enterprise [171] for the design of the UML class diagrams, and for the code generation of the adversary and fellowship models. Class and sequence diagrams are available at Appendix A and code snippets at Appendix B. We have used the GCC compiler [314] for compilation and the Rational ClearQuest [170] for the submission and tracking of the bugs. We initially used the CVS [243] and then switched to the Subversion [381] for version control. We relied on the GDB [315] and the Valgrind [355] for debugging, and the Python [131] for developing trace analysis scripts. Appendix C contains both the simulation and trace analysis scripts that have been employed in the simulation setup (§3.4.3) and analysis (§3.4.4). Finally, we deployed the Gnuplot [353] for drawing graphs and executed all of these tools on an Ubuntu Server Edition [354].

3.4.3 Simulation Setup

In this section, we present the general simulation parameters, parameters chosen for the 802.11 MAC and the fellowship, and finally the simulation scenarios. The simulation parameters for the NS2, 802.11 MAC and the fellowship are summarised in Table 3.3.

3.4.3.1 General Simulation Parameters

The simulation parameters specified in this section apply to all types of nodes (mobile, fellowship and adversary nodes) that are considered in our simulation until otherwise stated. Since the MANET lacks a standard benchmark for defining the simulation parameters, we have inherited those simulation parameters that have been predominantly adopted in the related security models [159, 164, 302].

We have considered the *random waypoint model* as the mobility model for the mobile nodes. In the random way point model, a mobile node starts from a random point, waits for a duration determined by the pause time, then chooses another random point, and moves to a new point with a velocity uniformly chosen between 0 and the maximum velocity (V_{max}). We have selected 250 m as the radio transmission range for

Table 3.3: Simulation Parameters for NS2, MAC, Fellowship and Adversary

NS2 Parameters	
Mobility	Random waypoint
Radio Transmission Range	250m
Traffic Type	Constant Bit Rate (CBR)
Date Rate	2Mbps
Payload Size	512 bytes
Total Number of CBR Connections	20
Minimum Duration for a CBR Connection	75s
Maximum Duration for a CBR Connection	125s
Total Simulation Time	300s
Average Maximum Velocity V_{max}	20m/s
Average Pause Time	10s
Average Simulation Area	$1200 * 1200m^2$
Total Number of Nodes	100
MAC 802.11 Parameters	
Interface Queue Length	50
Energy Consumed on Receiving a Packet	0.945J
Energy Consumed on Receiving a Packet	1.400J
Energy Consumed when Idle	0.045J
Initial Energy	1000J
Average Reception Time	0.004864s
Fellowship Parameters	
Packet-replica Queue's Packet Expiration Time	1s
Contribution-share ($^{\mathcal{A}}\eta_{\mathcal{I}}$)	5J
Negotiation threshold (ρ)	3 Intervals
Demerit factor (λ)	1.8
Merit factor (σ)	1.2
Transmission threshold (ν)	0.70
Malicious Parameters	
Probability of Malicious Action	100%
Distribution of Malicious Action	Random

a mobile node, and the *Constant Bit Rate (CBR)* as the traffic between the sender and receiver with a data rate of 2 Mbps and a packet size of 512 bytes. The CBR flows have been chosen to study the performance of the SMRTI in the presence of connectionless communication. These flows were dynamically varied between a randomly-chosen set of senders and receivers, instead of being maintained between the fixed set of senders and receivers for the entire simulation period. As demonstrated in the next §3.4.4, we observed this design to discover the multiple routes and to expose the DSR-MAC and fellowship nodes to many malicious nodes. We have allocated 300 s for the simulation duration, and 75 s and 125 s as the minimum and maximum duration for the CBR flows respectively. This realistically allowed a few terminating and commencing flows to overlap, rather than following a batch execution. The V_{max} and pause time are varied between 0 m/s to 50 m/s and 0 s to 40 s respectively, depending on the simulation scenario. Similarly, the simulation area is varied from $500 * 500m^2$ to $5000 * 5000m^2$. Nevertheless, the total number of nodes for all simulation scenarios is fixed at 100.

The NS2's version of the DSR ignores the unidirectional routes, so that mobile nodes can respond with acknowledgements to those received data packets (IEEE 802.11). The NS2 also incorporates the *flow state* extension [156] of the DSR, where an identity is established for a flow to reduce the source routing overhead. We have also enabled the DSR routing protocol to assist the fellowship model to promiscuously monitor the transmissions of the other nodes.

3.4.3.2 Parameters for MAC 802.11 and Fellowship

The interface queue inherits the default value mentioned in the NS2 as its length. The energy consumed for receiving and transmitting a packet are computed based on the packet length, which is given by the CBR traffic parameters and the time taken to transmit the packet. All nodes are provided with an initial energy of 1000 J, of which they trial 5 J of their energy (contribution-share) for every other node in the network to determine the behaviour of those nodes.

The time period for the packet expiration is fixed at one second, based on the value derived from our preliminary simulation tests. On the other hand, the negotiation threshold is set to three intervals to balance between the strict and relaxed requirements for indirect notification. The demerit (λ) and merit (σ) factors are established such that $\sigma > \lambda$; thus mobile nodes are discouraged from exhibiting non-cooperative behaviour. Finally, the transmission threshold (ν) is fixed at 70% to accommodate the uncertainty that steps-in owing to the characteristics of the wireless medium.

3.4.3.3 Performance Metrics

The performance metrics evaluated for all scenarios mentioned below are:

The ***Packet Delivery Ratio (PDR)*** *is the average ratio of total number of the CBR packets received by the destination to the total number of the CBR packets sent by the source.*

However, the performance metrics such as the packet or byte overhead are not evaluated in our simulations because the fellowship nodes do not generate additional packets or headers to communicate control information as in the related models. In the following, we present two sets of simulation results each comprising four different scenarios and demonstrating the effectiveness of the fellowship model against the packet droppers and flooders, respectively.

3.4.4 Simulation Results – Packet Drop Attack

The simulation setup for this category comprises the following two compositions, *fellowship and malicious* or the *DSR-MAC and malicious*. These compositions are independently simulated with identical parameters for each of the scenarios mentioned below. *To be precise, the malicious nodes are modelled to exhibit only the packet dropping behaviour.* The total number of nodes for each of the above-mentioned compositions is fixed at 100, although the proportion between the malicious and the fellowship (or DSR-MAC) nodes may vary depending on the simulation scenario. All these enabled us to compare the performance of the fellowship and the DSR-MAC nodes, that is, to study and compare the performance of the fellowship-enabled nodes with the fellowship-disabled nodes. The performance comparison between fellowship and the

DSR-MAC compositions for each scenario is derived from an average of 20 executions. The simulation scenarios considered for the performance analysis are as follows.

Scenario I. In this scenario, we evaluated the performance of the fellowship and the DSR-MAC nodes against the increasing proportions of the malicious nodes from 0 to 100 in increments of 10, with a pause time of 10 s, a V_{max} of 20 m/s, and a simulation area of $1200 * 1200m^2$. Note that the proportion of the fellowship or the DSR-MAC nodes decreases with an increasing proportion of the malicious nodes to maintain the total count of nodes at 100. The objective of this scenario is to discover the proportion of the malicious nodes after which the performance of the fellowship and the DSR-MAC nodes fall noticeably.

Scenario II. This scenario presents the impact of the mobility on the fellowship's (or the DSR-MAC) performance against the malicious nodes. We have maintained the proportion of the malicious to the fellowship (or the DSR-MAC) nodes to three uniformly-distributed values, (a) 15%, (b) 25% and, (c) 35% of the total nodes in the network. For each of these distributions, we then analysed the performance of the fellowship (or the DSR-MAC) nodes by varying the V_{max} from 0 m/s to 50 m/s in increments of 5 m/s with a pause time of 10 s, and the simulation area of $1200 * 1200m^2$.

Scenario III. Similar to the previous scenario, proportion of the malicious to the fellowship (or DSR-MAC) nodes is maintained at three uniformly-distributed values, (a) 15%, (b) 25% and, (c) 35% of the total nodes in the network. For each one of these distributions, we then analysed the performance of the fellowship (or DSR-MAC) by varying the pause time from 0 s to 40 s in increments of 4 s. This scenario throws an insight into the influence of the pause time on the performance of the fellowship (or DSR-MAC) nodes.

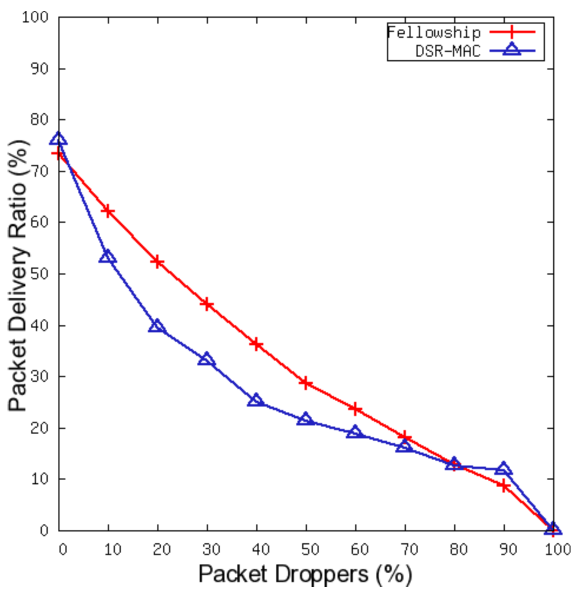
Scenario IV. In contrast to Scenario III, we varied the simulation area from $500 * 500m^2$ to $5000*5000m^2$ in increments of $500*500m^2$ but retained those remaining parameters. This scenario evaluates the impact of the network connectivity and thereafter, the influence of the node density on the performance of the fellowship (or DSR-MAC) nodes in the presence of the malicious nodes.

3.4.4.1 Scenario I

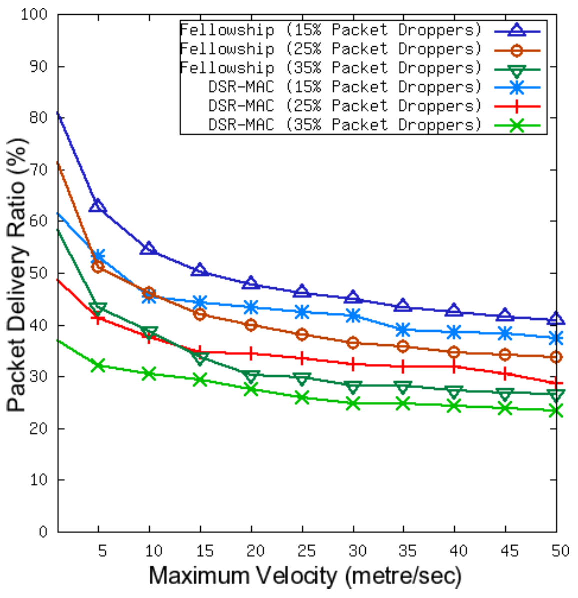
A steep reduction is seen in the DSR-MAC's PDR (Figure 3.4(a)) as the percentage of malicious nodes (or number of adversaries) increases in the network. The root-cause for such a reduction in the PDR results from the DSR-MAC nodes lacking the appropriate mechanism to detect and defend against the packet droppers. Recall that as the proportion of adversary increases, the probability of harbouring an adversary in every route increases. In the case of the connectionless CBR traffic, this situation is further worsened because a source DSR-MAC is not privileged with an acknowledgement from the destination DSR-MAC for every packet that was dispatched.

In comparison with the DSR-MAC nodes, the fellowship nodes sustain a better PDR although they experience the impact of the packet droppers. The fellowship nodes deliver on average a 20% better PDR than the DSR-MAC nodes. In particular, they deliver 30% better PDR when they are the majority in the network (their proportion exceeds the total count of adversaries in network). The strength of the fellowship nodes exists in detecting a packet dropper (next-hop or downstream node) and indirectly notifying the source of the CBR traffic through an upstream or previous-hop. As a result, the fellowship-enabled source node is provided with the information that the route is embedded with an adversary. This, in turn, triggers the source node to use alternative routes to stream the CBR traffic to the destination or to initiate a new route discovery in the absence of alternative routes.

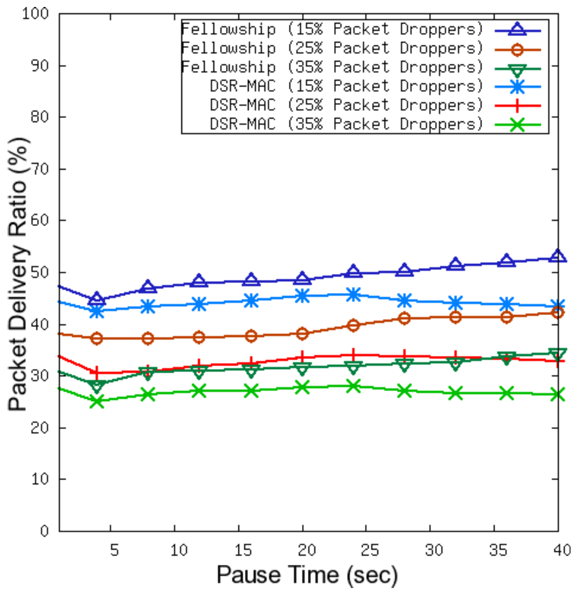
An interesting characteristic can be observed for fellowship nodes whenever the network is dominated by 90% of adversaries. The fellowship nodes exhibit a low PDR in comparison to the PDR exhibited by the DSR-MAC nodes. At such a high concentration of packet droppers, the fellowship nodes disadvantage themselves by attempting to discover new routes for queuing up packets. Since each discovered route is likely to host a packet dropper, the salvaging approach adopted by the fellowship nodes adds



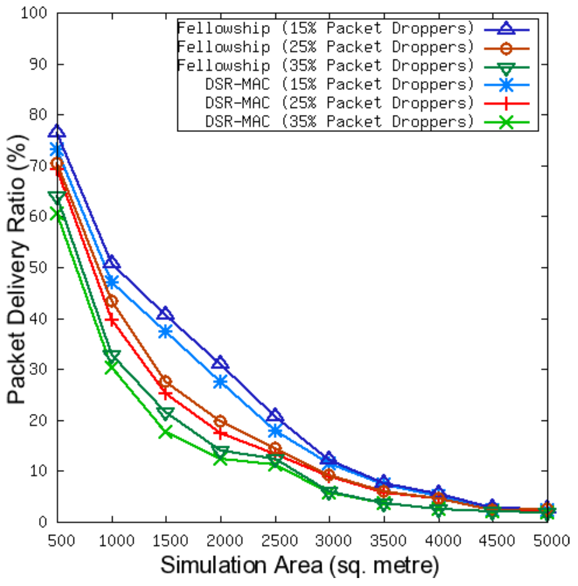
(a) Scenario I – PDR Vs Packet Droppers



(b) Scenario II – PDR Vs V_{max}



(c) Scenario III – PDR Vs Pause Time



(d) Scenario IV – PDR Vs Simulation Time

Figure 3.4: Performance of Fellowship and DSR-MAC Nodes against Packet Droppers.

fuel to the congestion, thereby witnessing the consequence in returns as a low PDR. Given that the DSR-MAC nodes adhere to the original routes, they harvest few stable routes that are provided by remaining 10% DSR-MAC nodes. Therefore, they avoid the self-inflicted congestion such as that inflicted by the fellowship nodes and successfully out-perform the fellowship nodes for once. Also recall that the fellowship nodes outperform the DSR-MAC nodes with a notable margin when the proportion of the adversaries is between 10% and 50% of the network.

3.4.4.2 Scenario II

Analogous to the previous scenario, the current scenario investigates the performance of the DSR-MAC and fellowship nodes against the adversaries with varying mobility, precisely when the proportion of the malicious nodes is between 10% and 50%. Therefore, three proportions of adversary are chosen, 15%, 25% and, 35%, against which the performance of the DSR-MAC and the fellowship nodes are examined. Namely, the DSR-MAC and the fellowship nodes with those proportions of the packet droppers are referred to as the DSR-MAC-15, DSR-MAC-25 and DSR-MAC-35, and the fellowship-15, fellowship-25 and fellowship-35 respectively.

As seen in Figure 3.4(b), the DSR-MAC nodes experience a peak value for the PDR when their mobility is at 0 m/s and a trough value for the PDR at 50 m/s. The root cause for the peak exhibit at 0 m/s in comparison with the trough at 50 m/s is to do with the static environment. On the other hand, the high mobility introduces an unstable environment thereby causing broken links. It also enables the packet droppers to move around the network, which is another factor that poisons stable communications. As expected, the performance descends from the DSR-MAC-15 to the DSR-MAC-35 owing to the increasing proportion of packet droppers.

The fellowship nodes imitate the characteristics exhibited by the DSR-MAC nodes thereby confirming that the change in mobility has a significant impact on the performance, regardless of the approach adopted as a defence mechanism. However, the fellowship nodes differ from the DSR-MAC nodes by capitalising on the static environment provided at 0 m/s. In particular, the peak performance of the fellowship nodes exceeds their trough performance by 38% on average, while it is 18% on average in the case of the DSR-MAC nodes. The key factor relies on the strength of fellowship nodes to detect and defend against packet droppers more effectively in a static environment than in a changing environment. Note that the capability of the fellowship nodes to detect a packet dropper is proportional to the observations made by those fellowship nodes that directly rely on the environment's stability. Although the performance of the fellowship nodes suffers under a high mobility as do with the DSR-MAC nodes owing to the broken links, they again outperform the DSR-MAC nodes at a high mobility because they can identify and isolate known packet droppers. As expected, the PDR descends in the following order, (a) *fellowship-15*, (b) *fellowship-25* and, (c) *fellowship-35*.

3.4.4.3 Scenario III

The varying pause time exhibits an inverse characteristic with respect to the performance obtained for varying mobility, that is, the PDR's slope at the low pause time correlates with the slope obtained for the high mobility or vice versa. As seen in Figure 3.4(c), the DSR-MAC nodes present minor variations in the PDR as the pause time progresses from 0 s to 40 s. In particular, the DSR-MAC nodes exhibit a similar performance at both the low and high ends of the pause time, while there is a marginally better performance between the low and high ends of pause time. At the low end of the pause time, high mobility influences the performance. Alternatively, at the high end of the pause time, over-exposure to adversary-embedded routes influences the performance. As seen earlier, the PDR of the DSR-MAC-15 is better than the PDR exhibited by the DSR-MAC-25. Similarly, the DSR-MAC-25 nodes outperform the DSR-MAC-35 nodes because of the difference in the proportion of packet droppers in the network.

Unlike the DSR-MAC nodes, the fellowship nodes present a smooth increase in the PDR as the pause time increases to the maximum value in Figure 3.4(c). The reason for the PDR to progress with the increasing pause time results because the time required to detect and defend against packet droppers becomes available for the fellowship nodes with the increasing pause times. An exception to the above discussion can be observed for the PDR when the pause time moves from 0 s to 40 s. Although the PDR at 4 s is supposed to exceed the PDR observed at 0 s, especially when the routes are more comparatively stable at 4 s than at 0 s, it does not happen, at least, for the following reason. The route discovery enforced by the intermediate fellowship nodes at the fellowship-enabled source node (after identifying a packet dropper on the route) is likely to fail within 4 s. Such initiation of new route discoveries at 4 s adds overhead to the highly mobile environment and impacts the performance. Similar to the previous scenario, the PDR descends in the following order, (a) *fellowship-15*, (b) *fellowship-25*, and (c) *fellowship-35*.

3.4.4.4 Scenario IV

Figure 3.4(d) presents the performance of the DSR-MAC nodes against varying the simulation area and henceforth the varying node density. The DSR-MAC nodes start with a high PDR when they are densely populated and then the PDR falls as the network area expands. When multi-hopped communications are further stretched as a result of the expanding network area, more broken links are introduced to hamper the communication flow. Furthermore, the expansion of the network area yields a disconnected network, thereby harbouring only one or two-hop communications. Stretching the network area therefore converts the sporadically-connected network into a completely disconnected network that, in turn, yields the minimum PDR for the DSR-MAC nodes. Although the PDR of the DSR-MAC-15 nodes is better than the PDR of the DSR-MAC-35 nodes, the expanding network area and thereby the decreasing nodal density masks the difference between the DSR-MAC and the packet droppers. Otherwise, this scenario signifies the importance of an operational network, even for an adversary to perform its role.

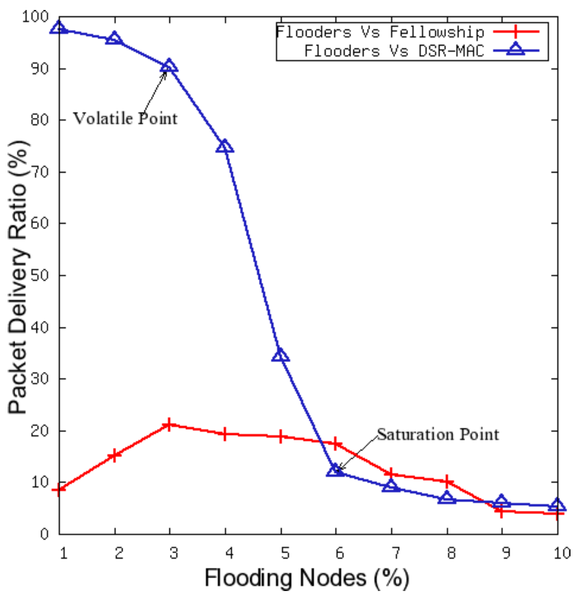
As seen in Figure 3.4(d), the fellowship nodes outperform the DSR-MAC nodes at high nodal densities because of their capability to detect and defend against packet droppers. However, as the simulation area expands, the longer unstable routes fail to complement fellowship's approach to re-discover new routes whenever a packet dropper is encountered. Further expansion in the network area worsen the fellowship's remediation approach of discovering alternative routes, because such alternative routes become unavailable in a partially-disconnected network. The expansion of the network area not only affects the fellowship nodes but also the adversaries because the total count of the active communications is reduced to the total number of available interconnections among nodes. The performance of the fellowship nodes are synonymous with the DSR-MAC nodes beyond a point where the network is totally disconnected and only single-hop communications are possible, that is, the point where the role of an adversary becomes meaningless. As expected, the PDR of the fellowship-15 nodes demonstrates a better performance than the PDR of the fellowship-25 and the fellowship-35 nodes.

3.4.5 Simulation Results – Flooding Attack

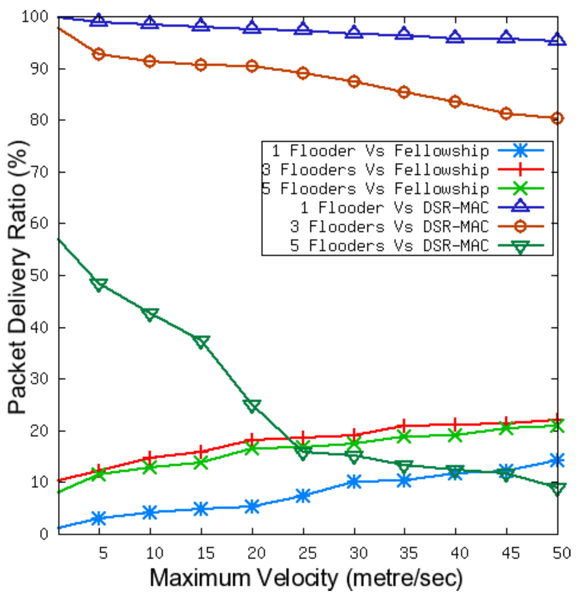
Unlike §3.4.4, this section focuses on understanding the flooding behaviour of the malicious nodes and the effectiveness of the fellowship nodes against them. Similar to the simulation setup described for packet droppers in §3.4.4, two types of nodal compositions are considered here, *fellowship and malicious*, and the *DSR-MAC and malicious*. These compositions are simulated with identical parameters for each of the simulation scenarios mentioned below.

In contrast to the scenarios discussed so far, the flooders take the role of the source node and, therefore, the following scenarios examine the performance or the PDR achieved by the flooders in the networks that are filled with the fellowship and DSR-MAC nodes. Flooding is defined as the transmission of 10 or more packets per second, while normal packet transmission rate is defined as 4 packets/sec. Except for the change in the duration assigned for the communication flow (150 s), the following scenarios inherit all the parameters from Table 3.3 and resemble the scenarios defined for packet droppers,

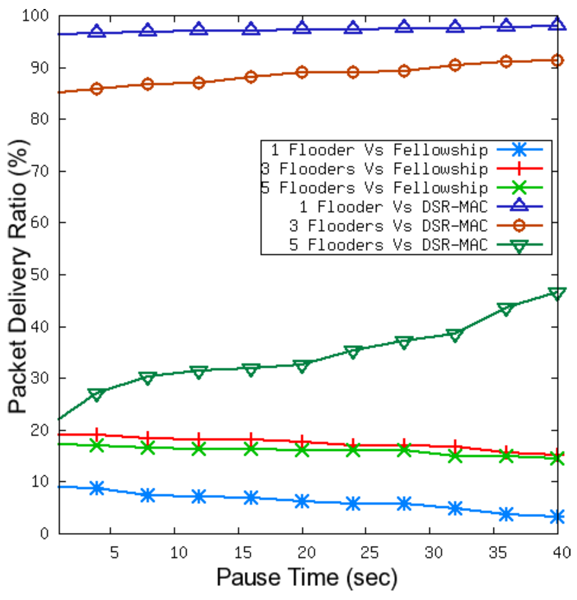
Scenario V. The impact of the flooders against the DSR-MAC and fellowship nodes is evaluated in this scenario. This setup remains synonymous with Scenario I except that the flooders are incremented from 0 to 10. Recall that the objective of this



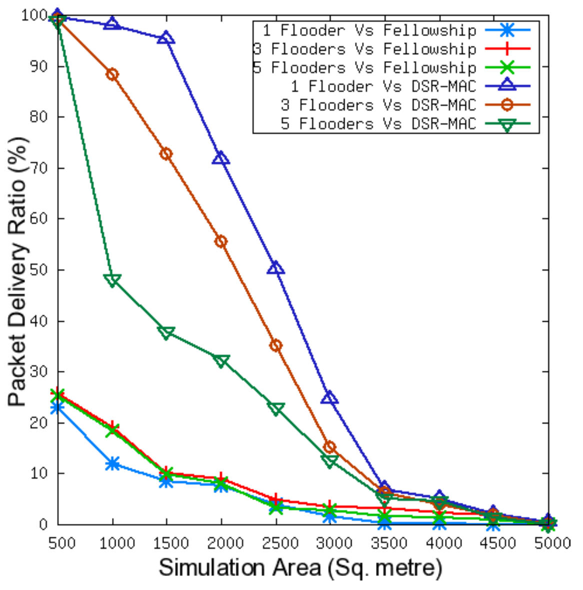
(a) Scenario V – PDR Vs Floodders



(b) Scenario VI – PDR Vs V_{max}



(c) Scenario VII – PDR Vs Pause Time



(d) Scenario VIII – PDR Vs Simulation Area

FIGURE 3.5: Performance of Fellowship and DSR-MAC Nodes against Flooders.

scenario is to discover the impact of the flooders in two independent networks that are filled with fellowship and DSR-MAC nodes respectively.

Scenario VI. This scenario presents the impact of mobility on the performance of the flooders in the presence of the fellowship (or DSR-MAC) nodes. We have maintained the proportion of malicious to the fellowship (or DSR-MAC) nodes to three uniformly-distributed values, (a) 1, (b) 3, and (c) 5 of total nodes in the network. For each of these distributions, we then analysed the performance of the flooders by varying V_{max} from 0 m/s to 50 m/s in increments of 5 m/s with a pause time of 10 s, and simulation area of $1200 * 1200m^2$.

Scenario VII. Similar to the previous scenario, the proportion of the malicious to the fellowship (or DSR-MAC) nodes is maintained at three uniformly-distributed values. For each of these distributions, we then analysed the performance of the flooders by varying the pause time from 0 s to 40 s in increments of 4 s. This scenario throws insight into the influence of the pause time on the performance of the flooders.

Scenario VIII. In contrast to Scenario VII, we varied the simulation area from $500 * 500m^2$ to $5000 * 5000m^2$ in increments of $500 * 500m^2$ but retained the remaining parameters. This scenario evaluates the impact of the network connectivity and the node density on the performance of the flooders in the presence of the fellowship (or DSR-MAC) nodes.

3.4.5.1 Scenario V

As seen in Figure 3.5(a), the flooders perform better against the DSR-MAC nodes than the fellowship nodes. Flooders experience 90% and above the PDR when their count is three or less in the network. In particular, a network filled with the DSR-MAC nodes offers a high PDR of almost 100% when only one flooder is present in the network. This means that entire network bandwidth is available for the flooder to exploit. However, the PDR of the flooders reduces steeply as their count increase from three to six, and beyond that they exhibit marginal variations. We refer to the point beyond which the slope of the PDR falls steeply as the *volatile point*, and the point after which the slope of PDR remains relatively steady as the *saturation point*. After the volatile point, the increase in the count of the flooders imposes a further overhead on the network, which introduces congestion and contention among the DSR-MAC nodes. However, the PDR is observed to show a minimal variation beyond the saturation point as the count of the flooders increases. The reason for the minimal variation results from the fact that the DSR-MAC nodes propagate packets until they are congested and then begin to discard packets thereafter.

The fellowship nodes successfully bring down the PDR of a flooder to 8%, thereby demonstrating the effectiveness of the rate-limitation component. Although the flooders experience a two-fold increase in their PDR when their count is incremented to the volatile point, the resultant PDR is below par and incomparable with the PDR achieved against the DSR-MAC for the same volatile point. Flooders experience an upward PDR up to the volatile point owing to the following factors, (a) *the availability of indisputable network bandwidth* and, (b) *the leverage provided by the mobility to move around the network and to exploit the fellowship nodes*. Note that the exploitation is bounded by the reception rate of the fellowship nodes. Thereafter, the PDR of the flooders undergoes a minimal variation up to the saturation point, and then disintegrates smoothly as the count of fellowship nodes increases beyond the saturation point. The contention and collision induced by the additional flooders, because of their struggle with each other to hijack the shared wireless channel explains why their PDR is below par. In summary, the fellowship nodes mitigate the impact of the flooders and restrict the propagation of the flooded packets; while, on the other hand, the flooders successfully exploit the DSR-MAC nodes to experience a high PDR.

3.4.5.2 Scenario VI

In a continuation with the previous scenario, three proportions of flooders are tested against the DSR-MAC and the fellowship nodes. These proportions are determined, based on the volatile and saturation points marked in Figure 3.5(a), and therefore the following counts are taken for the flooders, (a) 1, (b) 3, and (c) 5. The simulation setup of the DSR-MAC nodes containing these proportions are individually identified as, (a) *DSR-MAC-1*, (b) *DSR-MAC-3* and, (c) *DSR-MAC-5*. Similarly, the simulation setup of the fellowship nodes containing these three proportions are identified as, (a) *fellowship-1*, (b) *fellowship-3* and, (c) *fellowship-5*.

As shown in Figure 3.5(b), the flooders fail to perform better against the increasing mobility amidst the DSR-MAC nodes. It is obvious that the performance of the flooders in the DSR-MAC-1 setup is better than the DSR-MAC-3 and DSR-MAC-5 setups. However, it is not only that these setups differentiate among themselves in terms of performance, but also in terms of the rate at which the performance declines with increasing mobility. Especially, in the case of the DSR-MAC-5, the performance falls steeply against the increasing mobility. The performance of the DSR-MAC-3 setup takes the road of a gradual reduction. The key factor for the reduced performance arises from the mobility-induced broken links. Remember that as velocity increases, the soaring mobility introduces a highly dynamic network that is characterised by unstable short-lived routes that are known for triggering a high rate of route discoveries. The rationale for the steep fall in the performance of the flooders at high velocities results from the fact that their proportion is high. With a high proportion of flooders and mobility, the network is not only stressed with a high rate of broken links but also with traffic that chokes the bandwidth. Therefore, the network ultimately enters into a congested mode with flooded traffic trying to move around the network, while the new route discoveries in the wake of the broken links are attempting to lay a route to propagate that flooded traffic.

Interestingly, the flooders demonstrate an opposite characteristic against the fellowship nodes, except for the following. As expected, the performance of flooders against the fellowship nodes is poorer and lower than the DSR-MAC nodes. Unlike the performance seen against the DSR-MAC nodes, the flooders improve their performance as the mobility increases, provided the remaining nodes in the network are fellowship nodes. Recall that the rate-limitation component of the fellowship model is responsible for the clipping-off of the flooded packets, where the operation is known to excel over a period of time. In other words, as the time period increases, the rate-limitation transforms the knowledge gathered during the preventive clip-off phase into isolating the flooders during the reactive phase. In particular, when the contribution-share (η) for those flooders becomes negative, they fail to accept further packets from those flooders. Therefore, as the duration of the vicinity reduces between the fellowship nodes and the flooders because of the increasing mobility, the fellowship nodes are not provided with an opportunity to react to the behaviour of those flooders. In other words, the mobility allows the flooders to meet the unknown fellowship nodes and to capitalise on the rate-limited bandwidth offered by them. Furthermore, the fellowship nodes are expected to offer a high rate-limited bandwidth as the mobility increases, owing to the illusion created by the mobility that there are few neighbours in the environment. Henceforth, at the extreme end of the mobility, the fellowship nodes tend to assign their share of bandwidth to the flooders.

Another interesting factor is that the flooders exhibit a similar performance in both fellowship-5 and fellowship-3 setups; however, they marginally perform better in fellowship-3 setup than in fellowship-5 setup. Note that an inverse of the above is expected, such as a better performance in the fellowship-5 setup than in the fellowship-1 setup, and a similar performance relation between the fellowship-3 and the fellowship-1 setups. The prime factor that downgrades their performance in the fellowship-5 setup stems from the fact that bandwidth is distributed to match up with the increasing volume of traffic.

In summary, it can be concluded that high mobility impedes the performance of the flooders in a defenceless environment, while it favours them in a defensive environment.

3.4.5.3 Scenario VII

As seen in Figure 3.5(c), the flooders improve their performance when the pause time increases between their mobility in a network filled with DSR-MAC nodes. At the low-end of the pause time, the network is noted for a high count of unstable links. Therefore, the resulting route discoveries add stress to the performance, while at the high-end of the pause time, such stress is not viable owing to the relatively stable links.

As expected, the performance of the flooders in the DSR-MAC-1 setup is better than the performance observed from the DSR-MAC-3 and DSR-MAC-5 setups. The same can be observed in terms of their performance between the DSR-MAC-3 and the DSR-MAC-5 setups. However, the slope of their performance in each of these three simulation setups varies considerably for the following reason. Remember that the added overhead to the performance is proportional to the volume of the flooded packets, which, again, is proportional to the total count of the flooders in the network.

Flooders exhibit an opposite characteristic against the fellowship nodes, that is, they drop their performance as the pause time increases. The root-cause is due to the inverse relationship of the pause time with mobility. As the pause time increases, the time period available for the fellowship nodes to interact and react to the behaviour of flooders increases, which is the similar to the observation obtained for the low-end of the mobility in the previous scenario. Similarly, flooders perform marginally better in the fellowship-3 setup than in the fellowship-5 setup, which again contradicts the expectations. The causal factor for such observations stems from the fact that the bandwidth sharing increases with the increasing number of communication flows.

In summary, the high-end of the pause time that acts as the platform for the flooders to exploit the bandwidth of the DSR-MAC nodes in turn serves as the tool for the fellowship nodes to rate-limit and isolate those flooders.

3.4.5.4 Scenario VIII

The performance of flooders against both the DSR-MAC and the fellowship nodes reduces with an increase in the simulation area in Figure 3.5(d). In a tightly-packed network where the node density is high, an elevated performance is experienced by the flooders. Although they experience an ideal PDR against the DSR-MAC nodes, the same is not the case with the rate-limiting fellowship nodes.

As the simulation area expands, the traffic flows are characterised with a significant latency owing to the multi-hop routes. Further expansion introduces either longer routes with unstable links or smaller disconnected clusters. As a result, a few connections fail to complete the circuit, thereby causing a reduction in performance. Interestingly, the expanding simulation area transforms sparingly-connected network into a completely-disconnected network, in which, if at all, only single-hop connections are possible due to the node proximity. In such a situation, flooders perform invariably against both the DSR-MAC and the fellowship nodes. This further throws insight into the importance of a functional network for any operation to be executed, regardless of the objective.

3.5 Limitations

In this section, we analyse the limited extension of the fellowship to the pure MANET. We leave other limitations, such as constraints of promiscuous monitoring-based IDS to next Chapter 4, because such limitations are common to both the fellowship and SMRTI.

3.5.1 Pure MANET

Although the fellowship well suits the pure MANET, it is not fool-proof against adversaries. The fellowship's efficiency is restricted to single-hop, because the pure MANET lacks the centralised or distributed on-line or off-line authorities of the managed MANET. For this reason, the fellowship holds the previous-hop mobile nodes as responsible for transmitting packets irrespective of who is the originator of those packets. Given that the fellowship-enabled mobile nodes cannot authenticate their upstream neighbours prior to their previous-hop neighbour, it transfers the responsibility for categorising and prioritising the packets to the previous-hop before receiving those packets.

The above discussion does not guarantee that the fellowship-enabled mobile nodes are bullet-proof, especially against the one-hop neighbours. Although the security of the link layer is beyond the scope of this thesis, it has been shown that the MAC addresses are subject to spoofing attacks, similar to the IP addresses of the network layer. Therefore, the adversaries can feasibly spoof the MAC addresses to defeat the fellowship's operation. As explained in §3.3.4.1.4, the fellowship mitigates such attacks in terms of the policies that restrict the service offered to any unknown neighbour. Note that these limitations are specific to the type of network in which the fellowship operates and are not intrinsic to the fellowship. In other words, the fellowship operates better than the related models whilst considering the environment of the pure MANET, where the related models are predominantly tailored to handle the adversaries only in the managed MANET. Hence, the fellowship model is expected to excel in the presence of the managed MANET, where the network is characterised with pre-established secret associations among the mobile nodes. We discuss such a setup in Chapter 7. As noted earlier, Chapter 6 presents the integration of the fellowship and the SMRTI. The performance of the SMRTI integrated fellowship with the AODV as the routing protocol and the subjective-logic [183–188] as an extension will be discussed in Chapter 9.

3.5.2 Preventive Vs Reactive Approach

Although the fellowship is characterised as a prevention model, it is embedded with a reactive approach. The rate-limitation component of the fellowship proactively prevents a flooding attack, while the enforcement component of the fellowship resorts to a reactive approach to mitigate the packet drop attacks. As a result, the enforcement component is in a catch-up situation in comparison with the rate-limitation component. Hence, the fellowship nodes are compelled to being exposed to a notable behaviour of the packet droppers before excluding those packet droppers from the communication flows.

3.5.3 Formal Basis

Another limitation of the fellowship is the lack of a formal basis for its parameters and their thresholds and relationships among them. Note that the current approach defines the threshold values based on the simulation results that are the estimate of the averages. Alternatively, a formal basis can allow a precise definition for the threshold values by understanding the relationships between the various parameters and their thresholds. In other words, a formal study of thresholds would enable us to determine the optimal thresholds. Nevertheless, such a formal study warrants another thesis because of the complexity involved as a result of the linear and non-linear relationships between the numerous parameters in the fellowship model.

3.5.4 Resource as Metrics

Recall that battery energy is used as the resource that a mobile node can offer to other mobile nodes to estimate their nature, that is, whether they are benign or malicious. In our model, the contribution common (χ) is used as a source or sink for the offered or penalised resources respectively. In reality, there is no need for such a variable, provided a formal relationship is established among the involved parameters. In Chapter 6, we show how trust metrics can be used as metrics instead of the battery resource.

3.6 Conclusion

Although the fellowship is subject to the above limitations, most of those limitations are uniform for the related models. Furthermore, the fellowship outperforms those related models by resolving their shortcomings and operating within the inherent feature of the MANETs. In particular, the fellowship defends against both types of adversaries, *the malicious*, and *the selfish nodes*, and therefore, the possible packet drop and flooding attacks.

In summary, we have described how the fellowship operates as a decentralised and independent module at every node. In each mobile node, the fellowship is characterised by three components, the *rate-limitation*, *enforcement* and the *restoration*. The rate-limitation component is shown to be effective against flooders by restricting the rate of the received packets. Also, the rate-limitation has shown its impact against flooders by restricting those flooded packets from being propagated beyond a single hop. Furthermore, the rate-limitation has also proved its reactiveness to flooders by allowing them to expend their energy and not by including them in the future communication flow.

The enforcement component of the fellowship has been shown to mitigate packet drop attacks by identifying the packet droppers and excluding those packet droppers from their communication. In addition, the enforcement component assists the fellowship-enabled mobile nodes to discard the requests that are received from the packet droppers. Recall that the rate-limitation has been shown to exhibit a similar behaviour towards the flooders. The fellowship-enabled mobile nodes have shown the usage of the restoration component whenever they accept packets for forwarding, but fail to do so because of unforeseen conditions, such as congestion.

Unlike the incentive-based models that use tamper-proof tokens to motivate cooperation among mobile nodes, fellowship-enabled nodes use their internal battery resources to achieve the same objective. Although the notion of incentives appear to be similar between fellowship and related models, fellowship's design is free from incentive-related issues, such as, (a) *the need for a tamper-proof token*, (b) *the necessity to manage the token pricing*, (c) *the demand to distribute tokens* and (d) *the requisite to prevent nodes from being deprived of tokens.* For these reasons, fellowship-enabled nodes manifest their battery resources as incentives, such that those resources are contributed towards benign nodes to reciprocate their cooperation. Alternatively, they withdraw their resources to isolate packet droppers (or flooders).

This chapter further presents the distinguishing feature of the fellowship, where additional control packets or extra headers are not used to communicate the existence of the packet droppers to the upstream nodes. The information is communicated by the fellowship-enabled nodes to their previous-hop via superimposing with the approach adopted for the reception-rate negotiations. The downside of such an approach is the delay involved in communicating the presence of packet droppers through indirect negotiation. Furthermore, the policies adopted by the fellowship nodes with respect to bandwidth sharing and unknown neighbours have shown better performance results.

Finally, we have demonstrated the performance of the fellowship model using simulation results. Various simulation setups have been constructed against the flooders and packet droppers by taking into account the self-organised characteristics of the dynamically changing MANET. Those simulation results confirm that flooders struggle to thrive in the presence of the fellowship nodes. On the other hand, it has been shown that the fellowship nodes excel even in the presence of the packet droppers. Furthermore, those results confirm that there is a need for an operational network for any functionality to be realised. All performance metrics seen in the simulation results indicate that the fellowship would be a better defence against flooding and packet drop attacks (*i.e.* against adversaries such as malicious and selfish nodes).

Although we have not demonstrated colluding attacks in our simulations, fellowship's design inherently defends against colluding packet droppers (and flooders). Given that fellowship-enabled nodes evaluate their experiences and not their neighbour's experiences, they formulate subjective opinions. Furthermore, they neither request for feedbacks from other fellowship-enabled nodes nor from other colluding nodes. Therefore, their decisions are not influenced by other nodes. In other words, regardless of whether or not a packet dropper (or a flooder) colludes with another packet dropper (or flooder), the colluder is only rewarded with resources that are proportional to its behaviours towards a fellowship-enabled node. For this reason, a colluder exhibits the behaviour of a selective packet dropper (or a selective flooder) from the perspective of a fellowship-enabled node. As demonstrated in this chapter, fellowship-enabled nodes can successfully mitigate selective packet droppers (and flooders) and, therefore, they can effectively defend against the colluding packet droppers (and flooders).

4 SMRTI: Secure MANET Routing with Trust Intrigue

4.1 Introduction

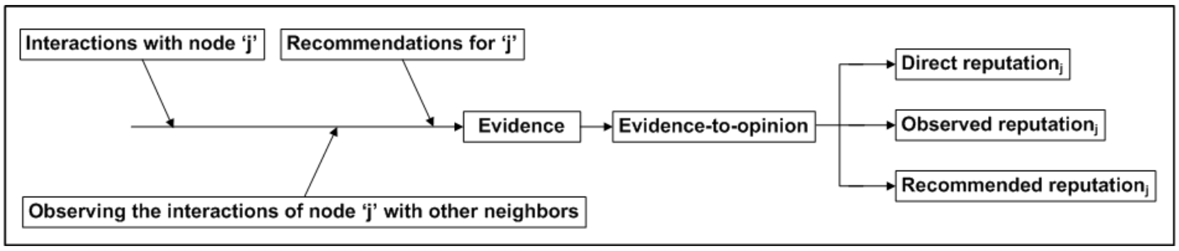
As discussed in Chapter 2, the shortcoming of the key management, secure routing, incentive-based payment and the IDS to measure the trustworthiness of other mobile nodes has eventually led to the growth of trust management systems [3, 38, 39, 64–71, 82, 83, 107, 112, 113, 116, 125, 134, 137, 140, 169, 178, 190, 204, 212, 214, 220–222, 233–235, 239, 241, 242, 252, 261, 270–273, 292, 294, 301–305, 308–311, 327, 328, 335, 347, 372, 373, 379, 380, 401, 404, 412, 424, 425, 461, 468] for the MANET. These systems are designed to proactively detect and reactively isolate malicious nodes despite the lack of consensus on the definition of *trust* at a more fundamental level. Although trust management systems have introduced a defence-in-depth strategy to enhance the security of the MANET communications, they are still incomplete owing to a few shortcomings discussed in Chapter 2.

In this chapter, we propose a reputation-based trust management system known as the ***Secure MANET Routing with Trust Intrigue (SMRTI)*** to resolve the shortcomings of the related trust management based models and, thence, to enhance the security of the MANET communications. An exception to the list of shortcomings covered by the SMRTI in this chapter involves the authentication of the collected evidence, which will be covered in detail in Chapter 7 together with the integration of the SMRTI and the key management and secure routing systems as a part of the ***Trust Enhanced security Architecture for MANET (TEAM)***.

The rest of this chapter is organised as follows. In the next §4.2, we will outline the approach adopted for the SMRTI. In §4.3, we will present the design of the SMRTI's detection-reaction based architecture. Section 4.4 details the DSR's adaptation to complement and tightly couple with the decision-making operation of the SMRTI for various event-specific contexts. In §4.5 and §4.6, we will explain in detail the *reaction* and *detection components* of the SMRTI, while the operation of *reputation-update module* that is common to both these components is then discussed in detail in §4.7. Finally, §4.8 explores the limitations of the SMRTI and §4.9 then gives some concluding remarks and summarises this chapter.

4.2 Proposed Approach

In this section, we will outline the approach adopted for the SMRTI, in which we will briefly discuss the collection and formulation of the evidence into opinions, and then the transformation of those opinions into trust-based decisions.



Interactions with node 'j'
Recommendations for 'j'
Evidence
Evidence-to-opinion
Direct reputation$_j$
Observed reputation$_j$
Recommended reputation$_j$
Observing the interactions of node 'j' with other neighbors

Figure 4.1: Evidence Collection and Formulation.

4.2.1 Evidence of Trustworthiness

A SMRTI-enabled mobile node collects the evidence of trustworthiness for other mobile nodes to formulate its opinion and establish a relationship with them. As seen in Figure 4.1, evidence is collected in the following manner, (a) *using a simple promiscuous monitoring-based IDS module* [247] and, (b) *through recommendations*. Note that the resolution to collect evidence from the IDS adds flexibility to the SMRTI, because the SMRTI can evolve against emerging attacks by readily advancing the IDS module without changing its internal operations. Similar to the related models, the behavioural evidence collected by passively monitoring the transmissions of neighbours after forwarding packets to those neighbours is known as *direct evidence*. This enables a node to precisely classify its neighbours as either benign or malicious. In addition, the SMRTI collects the behavioural evidence of the neighbours by passively observing their interactions; the collected evidence is known as *observed evidence*. The reason for observing those interactions between the neighbours is to shortlist those neighbours that are likely to misbehave in a future interaction and thus to defend against them, even before interacting with them. Analogous to the observed evidence that shortlists a malicious node even before an interaction, the *recommended evidence* is derived from the recommendations commune as a list of benign nodes even before interacting with them. In comparison with the related models, the SMRTI deploys a novel approach to communicate recommendations and thereby to resolve those issues that are associated with the recommendations. Notably, the approach deployed for the SMRTI neither disseminates extra packets nor includes additional headers to communicate the recommendations. Therefore, it eliminates the free-riding problem and operates within the

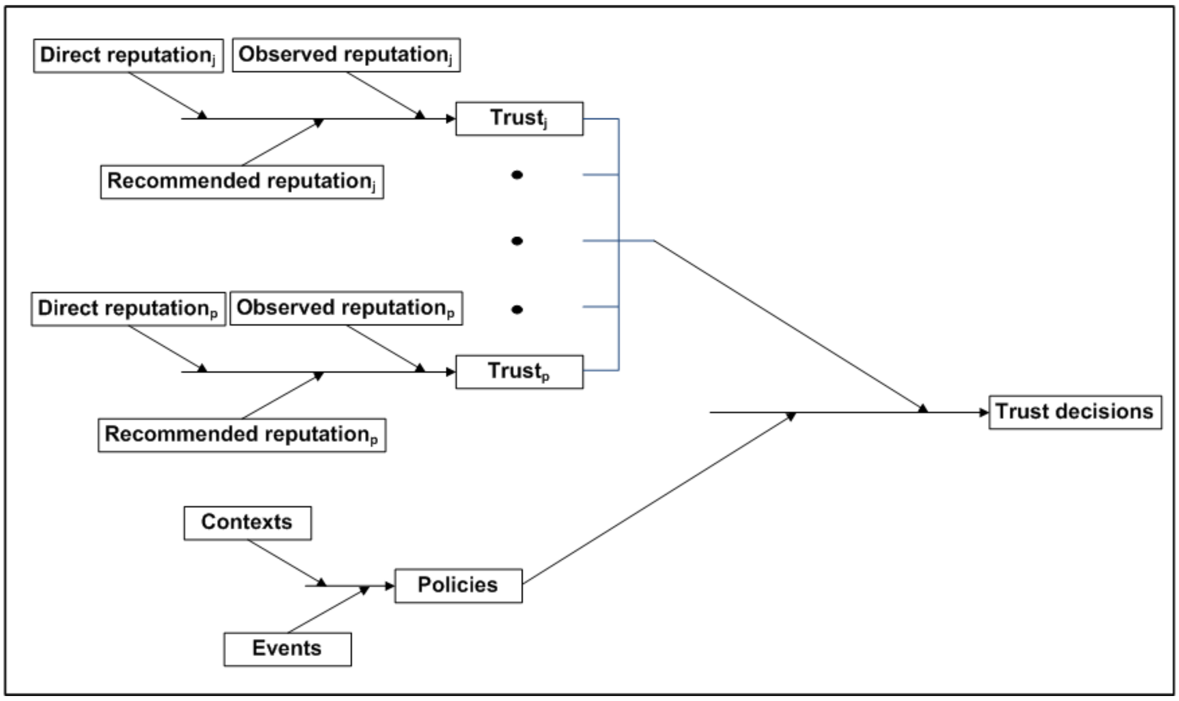


Figure 4.2: Process of Making Trust based Decisions.

specifications of the basic routing protocols. Furthermore, the SMRTI is able to defend against the recommender's bias and honest-elicitation problem, because the opinions are never disseminated as recommendations.

4.2.2 Evidence-to-Opinion Mapping

The evidence-to-opinion mapping function in the SMRTI quantifies the collected evidence to represent the resulting opinion as *reputation*. Therefore, in Figure 4.1, the node $\mathcal{I}$'s reputation for node $\mathcal{J}$ symbolises its relationship with $\mathcal{J}$. Given that $\mathcal{I}$ collects three types of evidence (direct, observed and recommended) for $\mathcal{J}$, it establishes three types of relationships with $\mathcal{J}$, namely, (a) *direct*, (b) *observed* and, (c) *recommended* reputations, respectively. It then uses those reputations to predict the future behaviour or *trustworthiness* of $\mathcal{J}$. The SMRTI resolves the issues pertaining to the evidence-to-opinion mapping function by, (a) *representing an opinion in the continuous reputation values between* -1 *and* $+1$ *as in [373]*, (b) *measuring the relative rate of change in the behaviour of the other mobile nodes, and classifying selectively-behaving*

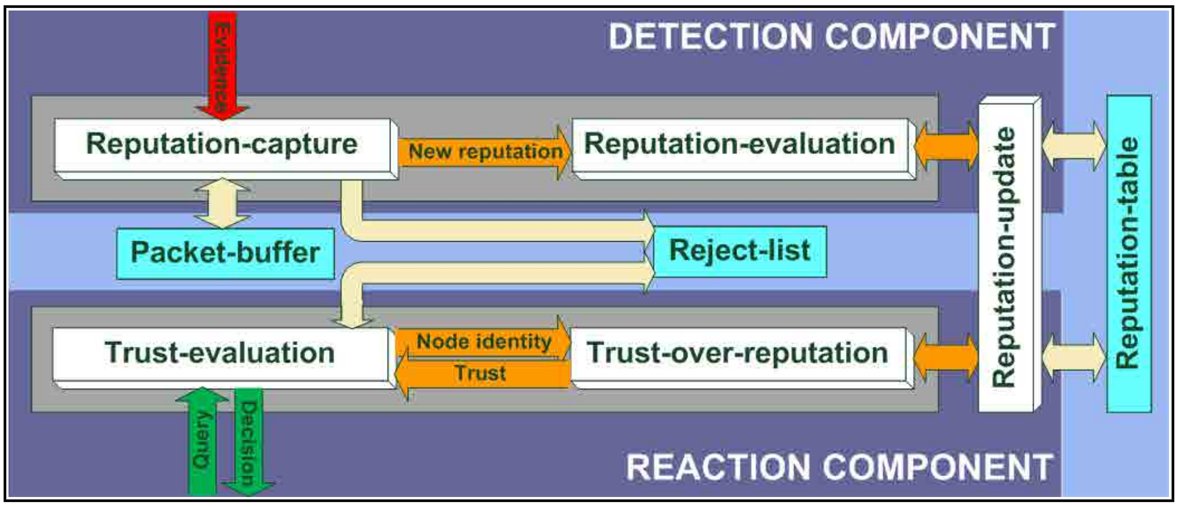


Figure 4.3: Architecture of SMRTI.

benign and malicious nodes accordingly into separate categories, (c) *accounting the uncertainty introduced into a relationship with another mobile node owing to the mobility or absence of evidence* and, (d) *giving more weightage to recent evidence, but ensuring that continuous benign or malicious behaviours do not conceal future malicious or benign behaviours.*

4.2.3 Trust Decisions

The SMRTI assists a node to make effective decisions for various event-oriented contexts by defining efficient policies for each of those contexts, and therefore measuring the trustworthiness of the nodes identified by those context-oriented policies (Figure 4.2). Note that the SMRTI's capability to define policies for future contexts that are relevant to emerging attacks potentially moulds it as an efficient trust-based decision-making system. The SMRTI is independent of the centralised and distributed online trusted authorities to make those trust-based decisions, and does not rely on any tamper-proof hardware for its operation. In this chapter, we will explore the detection-reaction based architectural design of the SMRTI using the DSR protocol. We will describe the interlock between the SMRTI and the AODV in Chapter 8, where the SMRTI is extended to incorporate the subjective-logic to measure the uncertainties in the trust relationships that exist between the newly-joining and existing nodes. We will use the terms, (a) *opinion*, (b) *reputation* and, (c) *relationship* interchangeably; they will also mean the same throughout this thesis. The instance of the SMRTI is deployed at every node and its operation at each node is distributed, decoupled and independent of the operation at the other mobile nodes.

4.3 Architectural Design

The SMRTI's architectural is composed of two main components, (a) *detection* and (b) *reaction*. The detection component constitutes the evidence collection and evidence-to-opinion mapping operations, while the reaction component constitutes the decision-making operation.

4.3.1 Detection Layer

The SMRTI *asynchronously* captures evidence for the benign and malicious behaviours of other mobile nodes using a promiscuous monitoring-based IDS module and through recommendations. It then expresses its opinion for those mobile nodes by quantifying the captured evidence as reputation depending on their behaviour and the context of the event. Hence, we define *'reputation' as an 'opinion' held for another mobile node, where the opinion is based on the evidence captured from that node's behaviour and it is subjective.*

As seen in Figure 4.3, the SMRTI's detection component is responsible for capturing and quantifying the evidence as reputation. It accomplishes the required operations through its modules, (a) *reputation-capture*, (b) *reputation-evaluation* and, (c) *reputation-update*, and the data structures, (a) *packet-buffer*, (b) *reputation-table* and, (c) *reject-list*. The reputation-capture module incorporates a promiscuous monitoring-based IDS module to collect evidence for an examined node's benign and malicious behaviours. It then quantifies and represents the collected evidence as the reputation, while the reputation-evaluation module aggregates a recent reputation with a previously held reputation to revise the opinion that is held for the examined node. Prior to the aggregation, the reputation-evaluation module calls the reputation-update module to account for the duration over which there has been no evidence since the last revision. The packet-buffer is used to store packets so that they can be used for comparison with the packets that are overheard by the reputation-capture through the IDS module. The reputation-table is used by the reputation-evaluation and reputation-update modules to retrieve the reputation of an examined node. A node formulates three types of reputation towards an examined node, (a) *direct*, (b) *observed* and, (c) *recommended*, which depend on the perspective by which the evidence was captured for the examined node, *i.e.* whether the evidence was collected from direct interaction, observation or recommendation. As detailed in §4.7, the reject-list contains the identity of the nodes that have misbehaved during one of the stages of the ongoing communication flow and therefore excludes them until the completion of the corresponding communication flow regardless of their high reputation.

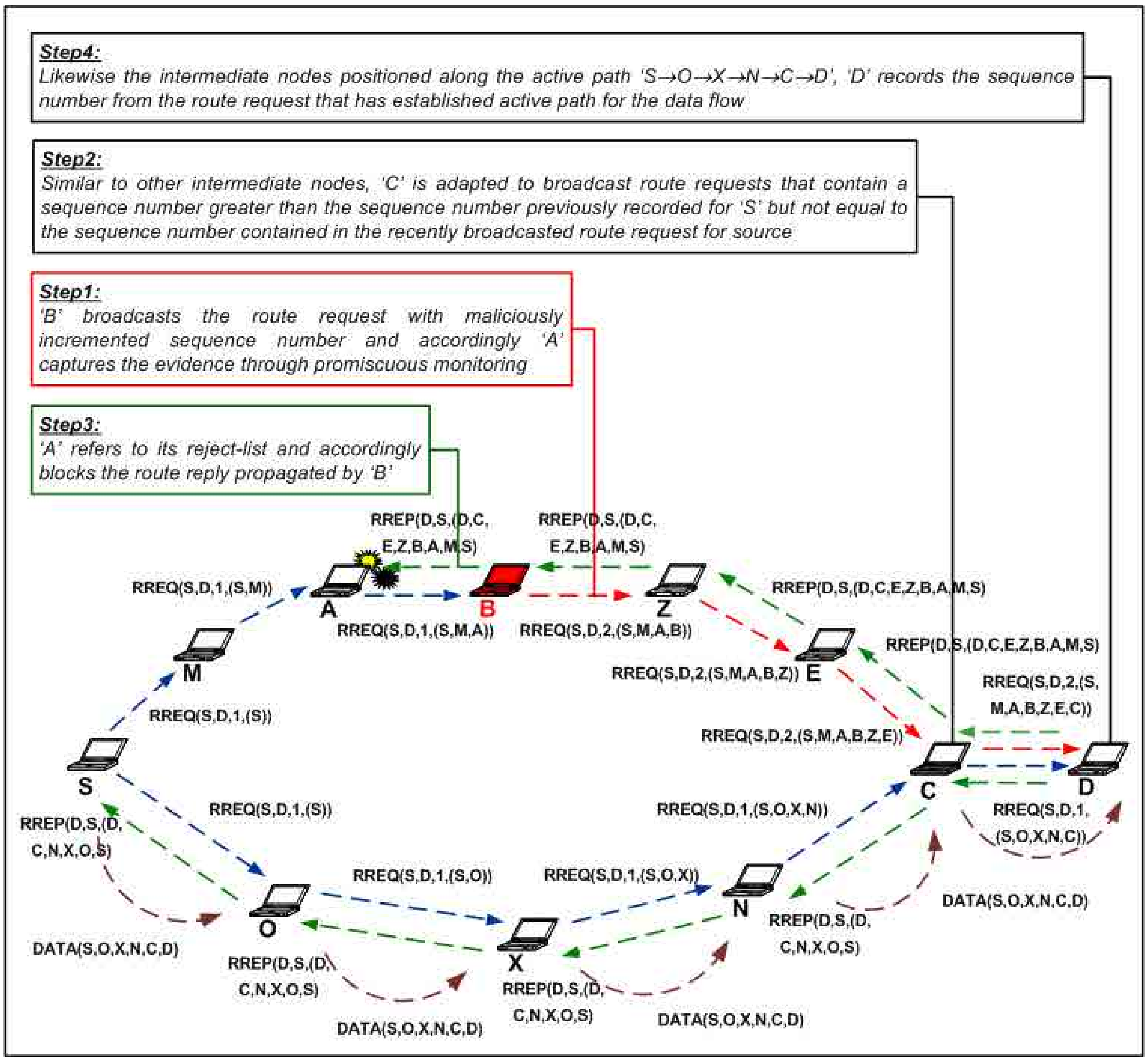
4.3.2 Reaction Layer

The SMRTI *synchronously* assists the DSR protocol to make the following decisions, such as, whether to, (a) *accept or reject a newly-discovered route from route discovery*, (b) *record or discard a route from a forwarded packet*, (c) *send a packet to or forward a packet on behalf of other nodes*, (d) *participate or refrain from a communication flow initiated by other mobile nodes*, (e) *include or exclude selectively-behaving nodes in a communication flow*, (f) *consider or ignore evidence*, (g) *update the latest sequence number contained in a route request or maintain the last seen sequence number for a source* and, (h) *which route to select for a self-initiated communication.* The decision for each of these event-specific contexts depends on the policy and the trustworthiness held for the shortlisted nodes by that context-oriented policy. Hence, we define *'trust' as the 'expectation' that a node will behave as predicted, with the factor then influencing the expectation being the reputation held for that node.* Since trust is subjective and depends on the opinion held for other mobile nodes and also applies only to a given context and time, it is expressed as a function of reputation. Hence, the trustworthiness of a mobile node is determined from the reputations (direct, observed and recommended) held for that node.

As seen in Figure 4.3, the SMRTI uses the reaction component to respond back with a decision to queries that are received from the DSR protocol. The reaction component performs the required operations through its modules, (a) *trust-evaluation*, (b) *trust-over-reputation* and, (c) *reputation-update*, and the data structures, (a) *reputation-table* and (b) *reject-list*. For each event-specific context that requires a decision, the trust-evaluation module extracts the specific nodes from the route depending on the policy detailed for that context. It then executes the decision for the context, based on the trustworthiness of those extracted nodes. The trust-over-reputation module computes each of the extracted node's trustworthiness using the reputations (direct, observed and recommended) held for them. The module relies on the reputation-table for its computation and reputation-updates to account for the period for which there has been no change in those reputations. The reject-list is used by the trust-evaluation module to exclude the nodes that have misbehaved during one of the stages of corresponding communication flow. Note that discrete values can only provide a small set of possible values, while the reputation and trust evolves continuously. Therefore, both reputation and trust values are represented by continuous real values $[-1, +1]$ as detailed in [373].

4.4 System Operation

This section presents the adaptations that have been incorporated into the DSR protocol to complement and tightly couple with the decision-making operation of the SMRTI for various event-specific contexts. For all contexts including, (a) *sequence number update for a source*, (b) *route recording, selection and pruning for a data flow*, (c) *packet propagation*, (d) *handling sensitive communications* and, (e) *route maintenance packets*, the SMRTI first confirms that the mobile nodes that are specified by the context-oriented policies are not enlisted in the reject-list. This approach enables the SMRTI to prevent the inclusion of mobile nodes that have misbehaved during one of the stages of the ongoing communication flow.



Step4:
Likewise the intermediate nodes positioned along the active path 'S→O→X→N→C→D', 'D' records the sequence number from the route request that has established active path for the data flow

Step2:
Similar to other intermediate nodes, 'C' is adapted to broadcast route requests that contain a sequence number greater than the sequence number previously recorded for 'S' but not equal to the sequence number contained in the recently broadcasted route request for source

Step1:
'B' broadcasts the route request with maliciously incremented sequence number and accordingly 'A' captures the evidence through promiscuous monitoring

Step3:
'A' refers to its reject-list and accordingly blocks the route reply propagated by 'B'

Figure 4.4: Adaptation of DSR to Record Valid Sequence Number.

4.4.1 Sequence Number Update

Let us consider the operation of *updating the sequence number* for a source at the SMRTI-enabled intermediate and destination nodes. The SMRTI instructs the intermediate and destination nodes to update the sequence number for a source from a route request only if the route request establishes an active route for the data flow. This not only prevents the intermediate nodes from recording a maliciously-incremented sequence number, but also allows the broadcast of other valid route requests that belong to the same route discovery cycle. Given that one of the SMRTI-enabled intermediate nodes can capture evidence for sequence number modification and accordingly enlist the adversary to its reject-list, thereafter the propagated malicious route request is prevented from establishing an active route between the source and the destination. Note that the policy to record the sequence number from a route request that has established an active route that, in turn, prevents the intermediate nodes from ignoring the duplicate route requests and hence counteracting the purpose of the sequence number. For this reason, the adaptation is extended to make the intermediate nodes broadcast a route request only if the route request's sequence number is greater than the previously recorded sequence number, but not equal to the sequence number that is contained in a recently-broadcasted route request.

Let us consider the scenario presented in Figure 4.4, where all nodes except the adversary $\mathcal{B}$ are enabled with the SMRTI. The intermediate nodes that are positioned along the paths, (a) $\mathcal{S} \mapsto \mathcal{M} \mapsto \mathcal{A} \mapsto \mathcal{B} \mapsto \mathcal{Z} \mapsto \mathcal{E} \mapsto \mathcal{C} \mapsto \mathcal{D}$ and (b) $\mathcal{S} \mapsto \mathcal{O} \mapsto \mathcal{X} \mapsto \mathcal{N} \mapsto \mathcal{C} \mapsto \mathcal{D}$ propagate a route request to destination $\mathcal{D}$ without updating the sequence number for source $\mathcal{S}$. In this scenario, the route request that is propagated along the path $\mathcal{S} \mapsto \mathcal{O} \mapsto \mathcal{X} \mapsto \mathcal{N} \mapsto \mathcal{C} \mapsto \mathcal{D}$ contains a valid sequence number, whilst the other one that is propagated along the path $\mathcal{S} \mapsto \mathcal{M} \mapsto \mathcal{A} \mapsto \mathcal{B} \mapsto \mathcal{Z} \mapsto \mathcal{E} \mapsto \mathcal{C} \mapsto \mathcal{D}$ holds a sequence number incremented by $\mathcal{B}$. Given that node $\mathcal{A}$ is positioned as a previous-hop to $\mathcal{B}$, it would collect evidence for $\mathcal{B}$'s malicious behaviour and enlists $\mathcal{B}$ into its reject-list. Although $\mathcal{D}$ responds back with a route reply to both malicious and valid route requests, node $\mathcal{A}$ effectively blocks the route reply received from $\mathcal{B}$. Therefore, the route reply that relates to the valid route request establishes the path $\mathcal{S} \mapsto \mathcal{O} \mapsto \mathcal{X} \mapsto \mathcal{N} \mapsto \mathcal{C} \mapsto \mathcal{D}$ for the data flow and the corresponding sequence number is then recorded at $\mathcal{O}$, $\mathcal{X}$, $\mathcal{N}$, $\mathcal{C}$, and $\mathcal{D}$. Alternatively, if $\mathcal{B}$ is benign, node $\mathcal{C}$ then propagates the first-seen route request along the path $\mathcal{S} \mapsto \mathcal{O} \mapsto \mathcal{X} \mapsto \mathcal{N} \mapsto \mathcal{C} \mapsto \mathcal{D}$ and effectively blocks the duplicate route request received along the path $\mathcal{S} \mapsto \mathcal{M} \mapsto \mathcal{A} \mapsto \mathcal{B} \mapsto \mathcal{Z} \mapsto \mathcal{E} \mapsto \mathcal{C} \mapsto \mathcal{D}$. Thus, the *SMRTI-directed DSR* behaves similar to the *pure DSR* in the absence of an adversary and effectively defends against the update of a maliciously-incremented sequence number in the presence of an adversary. For instance, in Figure 4.4, if the SMRTI is disabled at all nodes, then the DSR would allow the intermediate nodes ($\mathcal{Z}$, $\mathcal{E}$ and $\mathcal{C}$) to record the sequence number from the route request propagated by adversary $\mathcal{B}$. In addition, it would also compel $\mathcal{C}$ to discard the valid route request that was received along the path $\mathcal{S} \mapsto \mathcal{O} \mapsto \mathcal{X} \mapsto \mathcal{N} \mapsto \mathcal{C} \mapsto \mathcal{D}$.

4.4.2 Route Recording

Let us consider the operation of *recording a route* at the SMRTI-enabled source, destination and intermediate nodes. The source and destination nodes discover new routes through the route discovery process, while the intermediate nodes capture routes from the forwarded packets. Although the source records a newly-discovered route at the end of the route discovery process, the destination is deferred from recording the route until it witnesses the data flow through that route, and the same holds true for the intermediate nodes. This policy ensures that the destination and intermediate nodes consider only the operational routes for the route cache, because those are the only routes that are piggybacked with data packets. Note that in the case of the source, the route reply piggybacks the route request to deliver an operational route. In all the above cases, the SMRTI adheres to the policy of evaluating the trustworthiness of all the nodes that are listed in the route (except the mobile node at which it conducts the evaluation) before registering the route into the route cache. The SMRTI adopts this strategy to prevent the entry of routes that may otherwise contain nodes that have been misbehaving, except in the current communication flow.

As shown in Figure 4.4, the SMRTI-enabled destination $\mathcal{D}$ and the intermediate nodes, $\mathcal{O}$, $\mathcal{X}$, $\mathcal{N}$ and $\mathcal{C}$ wait for the data flow to record the piggybacked operational route $\mathcal{S} \mapsto \mathcal{O} \mapsto \mathcal{X} \mapsto \mathcal{N} \mapsto \mathcal{C} \mapsto \mathcal{D}$ from a data packet, provided that the route is trustworthy. Alternatively, the SMRTI-enabled source $\mathcal{S}$ records the piggybacked operational route $\mathcal{S} \mapsto \mathcal{O} \mapsto \mathcal{X} \mapsto \mathcal{N} \mapsto \mathcal{C} \mapsto \mathcal{D}$ directly from the route reply at the end of the route discovery process, provided that the route is trustworthy.

4.4.3 Route Selection

Let us now consider the operation of the *route selection* at a SMRTI-enabled source. To begin with, the source node initially checks its route cache for a route to the destination, whenever it wishes to send data packets to the destination. On finding a route, the source then evaluates the route's trustworthiness using the SMRTI. Recall that a route's trustworthiness is determined by evaluating the trustworthiness of all nodes that are positioned in the route, except for the node that is conducting the evaluation (the source node). Although a route's trustworthiness is evaluated whenever the route is recorded into the route cache, the SMRTI adheres to the policy of evaluating the route's trustworthiness prior to every deployment. The reason for the re-evaluation is that trust is *non-monotonic*, and hence the trustworthiness of a route is likely to change at anytime between the point of entry and deployment. The route with the highest trustworthiness is then chosen for the communication, given that there are multiple trustworthy routes to the destination. If the trustworthiness held for two routes is at the same level, then the route with the shortest path is selected for the communication. One of the routes can be arbitrarily chosen for communication if the path lengths are identical. Alternatively, the less-congested path can be selected to reduce the communicational delay, because the shortest paths are often prone to congestion. It is noted that the selection of less-congested path warrants the presence of a load balancing mechanism [157]. A new route discovery is initiated to the destination, only if untrustworthy routes exist or routes are unavailable to the destination. In the case of the former, the SMRTI instigates the route cache to purge the untrustworthy routes.

4.4.4 Route Pruning

Let us consider the operation of *pruning a route* at the source by one of the SMRTI-enabled intermediate nodes. In the case of the data flow, if an intermediate node's next-hop modifies the data packet's route header and, accordingly, the intermediate node captures the evidence for modification, then the SMRTI instructs the intermediate node

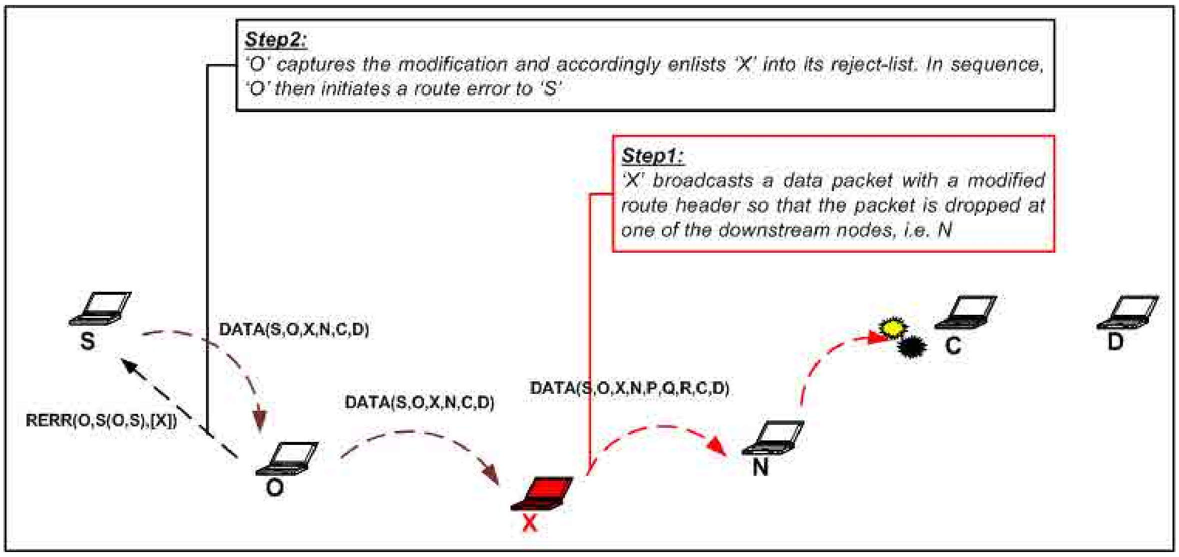

Step2:
'O' captures the modification and accordingly enlists 'X' into its reject-list. In sequence, 'O' then initiates a route error to 'S'
Step1:
'X' broadcasts a data packet with a modified route header so that the packet is dropped at one of the downstream nodes, i.e. N
S
DATA(S,O,X,N,C,D)
RERR(O,S(O,S),[X])
O
DATA(S,O,X,N,C,D)
X
DATA(S,O,X,N,P,Q,R,C,D)
N
C
D

Figure 4.5: Adaptation of DSR to Handle Route Modification Attacks.

to enlist the malicious next-hop into its reject-list. In sequence, the SMRTI triggers the intermediate node to initiate a route error to the source so that the latter is notified on the status of data flow. In addition, the route error enables the source node to prune the downstream segment of the route that starts from the position of the reported malicious node. Finally, it also induces the source node to choose an alternate route for the data flow. We will discuss in the next §4.4.5, that only the route error generated by the benign intermediate node would reach the source, so that it can influence the source to switch to an alternate route. Note that the DSR has provision to extend the route error to accommodate additional notifications, other than for broken links.

Let us consider Figure 4.5, where adversary $\mathcal{X}$ is assumed to modify the routing header of a data packet before forwarding the packet to its next-hop $\mathcal{N}$. In such an instance, the previous-hop $\mathcal{O}$ would capture $\mathcal{X}$'s malicious behaviour in the form of direct evidence. Note that $\mathcal{O}$ can capture direct evidence, even for a modified payload, which may be either plain or end-to-end encrypted. However, the hop-by-hop encrypted payload is beyond the reach of $\mathcal{O}$'s promiscuous monitoring-based IDS. The collected direct evidence then influences $\mathcal{O}$ to enter $\mathcal{X}$ into its reject-list and eventually prevents it from forwarding subsequent data packets to $\mathcal{X}$. In addition, $\mathcal{O}$ also generates and propagates a route error that reports $\mathcal{X}$ as unacceptable to $\mathcal{S}$, so that $\mathcal{S}$ can choose an alternate route for the data flow.

4.4.5 Packet Propagation

Now, consider in sequence the operation of *receiving and forwarding packets* at the SMRTI-enabled intermediate, source, and destination nodes. At the intermediate nodes, the SMRTI details a set of policies to decide whether to forward a packet on behalf of (or to send a packet to) another node. To accept a packet, it evaluates the trustworthiness of the previous-hop from which the packet was received. Likewise, the SMRTI also evaluates the trustworthiness of the next-hop to which the packet must be sent. Since the previous-hop's (or next-hop's) trustworthiness is determined by using its reputations and hence from the collected evidence, the SMRTI can thus identify the malicious or misbehaving previous-hop (or next-hop). In such an instance, the SMRTI excludes the malicious previous-hop (or next-hop) from the communication flow. Therefore, the SMRTI directed trust evaluations at the intermediate nodes allow them to effectively defend against the reception (or propagation) of the route discovery packets from the malicious nodes and hence prevent the establishment of routes that are embedded with malicious nodes. On the assumption that the DSR discovers multiple routes between the source and the destination nodes, the data flow is then likely to be undeterred by another trustworthy route. Finally, the SMRTI assists the intermediate nodes to make a decision whether to participate or refrain from a communication flow by evaluating the trustworthiness of the source and the destination. The reasoning is that the intermediate nodes forward the packets only to bridge the communication between the source and the destination.

As shown in Figure 4.6(a), the SMRTI-enabled intermediate node $\mathcal{N}$ initially evaluates the trustworthiness of its previous-hop $\mathcal{X}$. This is in accordance with the expectation that the received upstream packet is likely to be free from modification only if $\mathcal{X}$ is trustworthy. Similarly, $\mathcal{N}$ then evaluates the trustworthiness of its next-hop $\mathcal{C}$ that is in accordance with the expectation that the transmitted downstream packet is likely to reach the destination $\mathcal{D}$ without modification, only if $\mathcal{C}$ is trustworthy. Finally, $\mathcal{N}$

decides to participate in the communication flow between the source $\mathcal{S}$ and the destination $\mathcal{D}$, depending on the trustworthiness held for them. As explained previously in §4.4.4, it is the non-monotonic characteristic of the trust that drives $\mathcal{N}$ to perform a sequence of trust evaluations prior to the propagation of every packet.

Similarly, in Figure 4.6(b), the SMRTI-enabled source $\mathcal{S}$ adheres to the policy of evaluating the trustworthiness of the next-hop $\mathcal{O}$ and the destination $\mathcal{D}$ before propagating a packet. Alternatively, in Figure 4.6(c), the SMRTI-enabled destination $\mathcal{D}$ adheres to the policy of evaluating the trustworthiness of the previous-hop $\mathcal{C}$ and the source $\mathcal{S}$ before accepting a packet.

This set of evaluations for the packet propagation holds true for all events except for the route request event. The source and intermediate nodes vary the evaluations during the route request event, because the next-hop is unknown owing to the broadcast nature of the route request. In such a situation, the intermediate nodes only evaluate the trustworthiness of the previous-hop and communication flow (for the source and destination). Similarly, the source only evaluates the trustworthiness of the destination.

4.4.6 Sensitive Communications

In the case of a sensitive communication, the SMRTI instructs the source and the intermediate node to refrain from routes that contain selectively-behaving benign or malicious nodes. Nonetheless, the evidence for a selectively-misbehaving node's benign behaviour (or selectively well-behaving node's malicious behaviour) is evaluated as discussed in §4.6. Later, a route containing a selectively-misbehaving node is recorded or packets are forwarded on the behalf of (or sent to) a selectively-misbehaving node, only if the selectively-misbehaving node increases its trustworthiness at the evaluating source and the intermediate nodes by exhibiting a persistent benign behaviour. Otherwise, selectively-misbehaving nodes are given an opportunity by the source node only when no alternative routes exist for an insensitive communication flow.

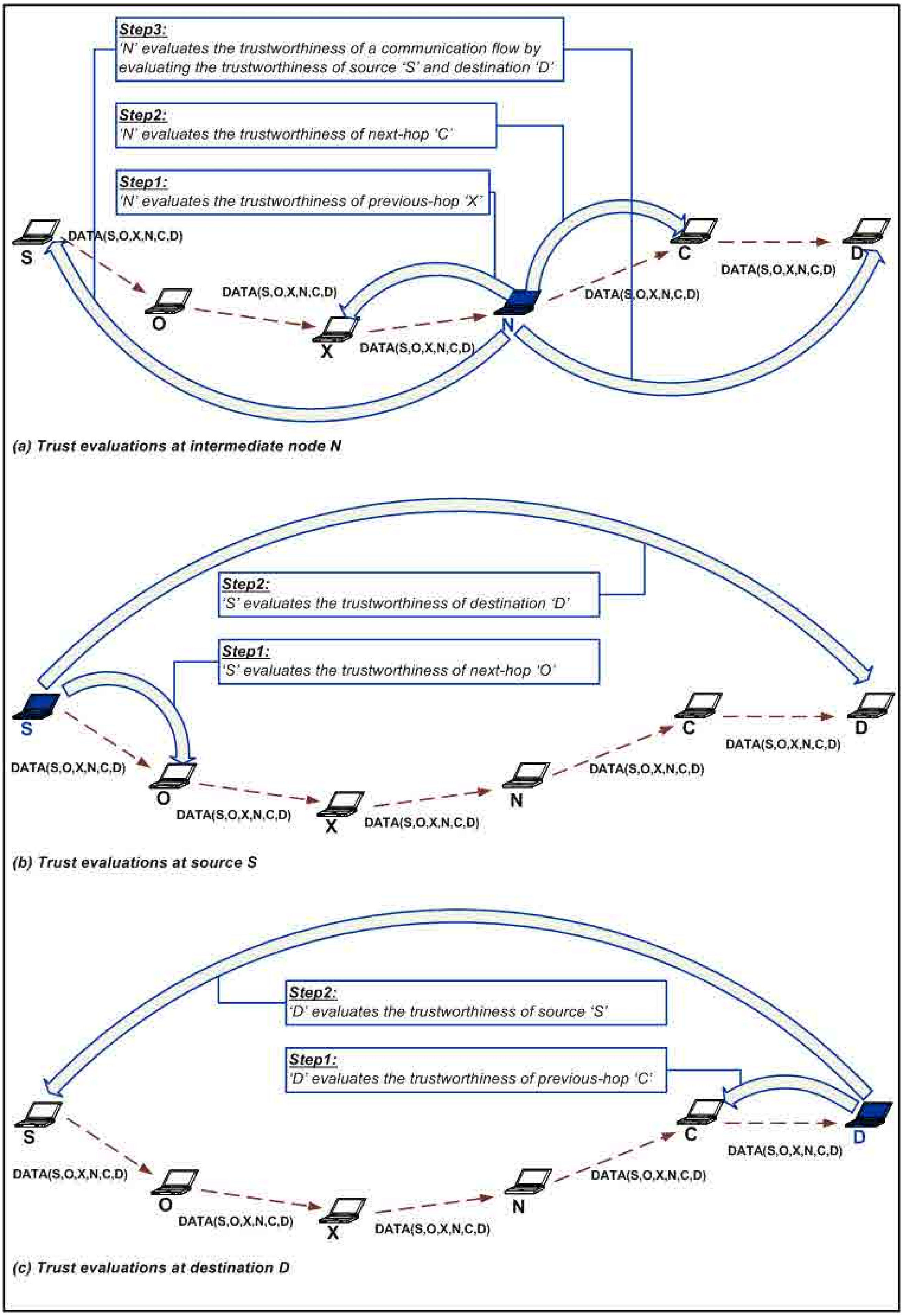

Step3:
'N' evaluates the trustworthiness of a communication flow by evaluating the trustworthiness of source 'S' and destination 'D'
Step2:
'N' evaluates the trustworthiness of next-hop 'C'
Step1:
'N' evaluates the trustworthiness of previous-hop 'X'
S
O
X
N
C
D
DATA(S,O,X,N,C,D)
(a) Trust evaluations at intermediate node N
Step2:
'S' evaluates the trustworthiness of destination 'D'
Step1:
'S' evaluates the trustworthiness of next-hop 'O'
(b) Trust evaluations at source S
Step2:
'D' evaluates the trustworthiness of source 'S'
Step1:
'D' evaluates the trustworthiness of previous-hop 'C'
(c) Trust evaluations at destination D

Figure 4.6: Adaptation of DSR to Propagate Trustworthy Packets.

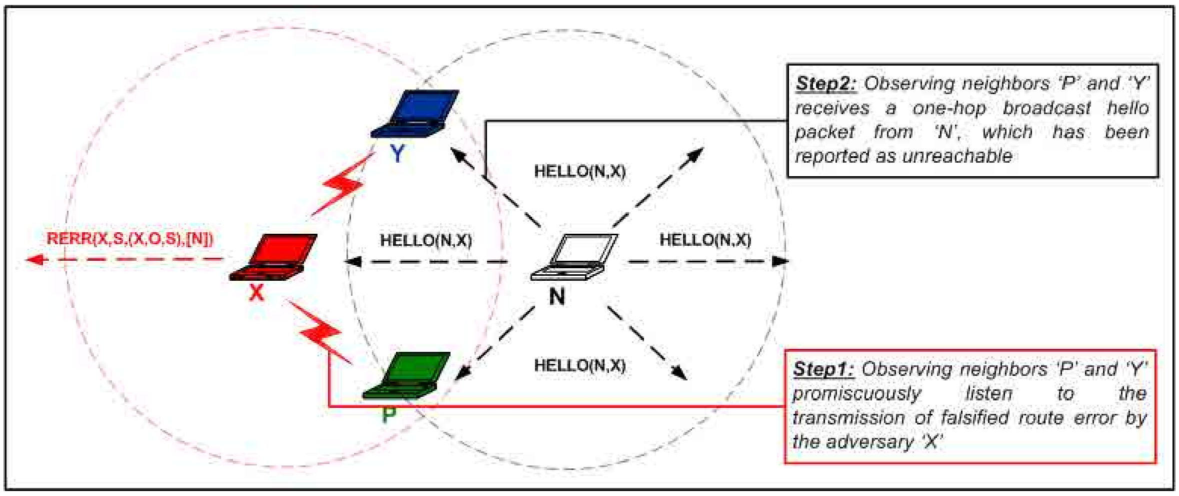

Step2: Observing neighbors 'P' and 'Y' receives a one-hop broadcast hello packet from 'N', which has been reported as unreachable
Y
HELLO(N,X)
RERR(X,S,(X,O,S),[N])
HELLO(N,X)
HELLO(N,X)
X
N
HELLO(N,X)
P
Step1: Observing neighbors 'P' and 'Y' promiscuously listen to the transmission of falsified route error by the adversary 'X'

Figure 4.7: Adaptation of DSR to Defend Against Falsified Route Error.

4.4.7 Route Maintenance

Recall that the SMRTI assists the DSR in making efficient trust-based decisions by defining effective policies for all event-specific contexts. These policies are responsible for validating the credibility of a given context by adapting the operation of the DSR and evaluating the trustworthiness of the nodes that are relevant to the given context. It is the sequence of the multi-stage route discovery followed by the data flow that facilitates the SMRTI in validating the credibility of any given context. In the same way, the SMRTI also validates the evidence collected from direct interactions, observations and recommendations before considering them for direct, observed and recommended reputations, respectively. This will be discussed in detail in §4.6.

However, in comparison with all contexts that have been discussed so far, the route maintenance context stands separate, because it is difficult to validate the credibility of a broken link that is reported in a route error packet. The difficulty results from the single-staged design of the route error event and, therefore, the lack of additional information to validate the genuineness of a broken link. For this reason, most of the secure routing systems [21, 75, 76, 105, 120, 136, 150, 154, 155, 159, 161–164, 179, 180, 215, 250, 260, 282, 284, 337, 367, 397, 399, 439, 440, 445, 446, 455] struggle to cope with route maintenance-related attacks.

Regardless of the difficulty associated with the validation of a route maintenance context, the SMRTI mitigates the impact of the falsified route error by instructing the intermediate and destination nodes to conduct the sequence of trust evaluations mentioned in §4.4.4. Therefore, the SMRTI-enabled intermediate nodes propagate a route error depending on the trustworthiness held for the following nodes, (a) *source of route error*, (b) *previous-hop from which the route error was received*, (c) *next-hop to which the route error must be sent* and, (d) *the destination of route error*. Similarly, the SMRTI-enabled destination accepts a route error only if its previous-hop and the source of the route error are trustworthy. In the case of a falsified route error, the SMRTI directs the node that has been falsely reported as unreachable by the route error to broadcast a one-hop hello packet. Given that the SMRTI-enabled neighbours can observe the route error propagated by an adversary and the subsequent hello packet broadcasted by the reported node (that has been notified as unreachable by the route error), the SMRTI-enabled neighbours effectively collect evidence for the adversary's malicious behaviour. For instance, in Figure 4.7, the likelihood of a falsified route error being dropped at one of the intermediate nodes (or the target) and the certainty that the adversary's ($\mathcal{X}$) observed reputation would be decreased at all observing neighbours, ($\mathcal{P}$ and $\mathcal{Y}$) is believed to discourage the adversary ($\mathcal{X}$) from propagating the falsified route error.

4.5 Reaction Component

As we have mentioned earlier, the reaction component synchronously responds with trust-based decisions to the queries received from the DSR protocol.

4.5.1 Trust-evaluation Module

The trust-evaluation module makes trust-based decisions for the various event-specific contexts depending on the policies defined for those contexts, which were detailed in §4.4. It declares a context as trustworthy only if the trustworthiness of the context is at least the predefined threshold value, *threshold-limit* (Δ). Prior to each evaluation, the module also confirms that the nodes identified by the context-specific policy are not enlisted in the reject-list. Although Δ can be varied depending on the context and the type of event, a uniform Δ is assumed for the ease of explanation.

Let us consider the evaluation of a communication flow's trustworthiness. The trust-evaluation module at an intermediate node $\mathcal{I}$ computes the trustworthiness of a communication flow *f*, $(T_{\mathcal{I}}Flow_f(t_{a+1}))$, according to (4.1). Parameter α shifts the priority between a packet's source *Src* and destination *Dest* depending on the type of event. For instance, the source *Src* is given a higher priority during the route request event, while the destination *Dest* is given a higher priority during the route reply event. The trust-evaluation module at node $\mathcal{I}$ computes the trustworthiness of the source *Src*, $(T_{\mathcal{I}}Node_{Src}(t_a))$, and the destination *Dest*, $(T_{\mathcal{I}}Node_{Dest}(t_a))$, by calling the trust-over-reputation module.

$$T_{\mathcal{I}}Flow_f(t_{a+1}) = \Big\{\big[\alpha \cdot T_{\mathcal{I}}Node_{Src}(t_a)\big] + \big[(1-\alpha) \cdot T_{\mathcal{I}}Node_{Dest}(t_a)\big]\Big\}; \qquad 0 < \alpha < 1; \quad t_{a+1} > t_a; \tag{4.1}$$

Similar to (4.1), the trust-evaluation module at node $\mathcal{I}$ computes the trustworthiness of a route R, $(T_{\mathcal{I}}Route_R(t_{a+1}))$, which is given in (4.2). Recall that the trust-over-reputation module is used to compute the trustworthiness of each node $\mathcal{J}$, $(T_{\mathcal{I}}Node_{\mathcal{J}}(t_a))$, that is positioned on the route. The parameter $\beta_{\mathcal{J}}$ signifies the priority assigned to the trustworthiness of each of those nodes that are positioned in the route. However for ease of explanation, the trustworthiness of all nodes that are positioned in the route are assumed to have the same priority.

$$\begin{aligned} T_{\mathcal{I}}Route_R(t_{a+1}) &= \sum_{(\mathcal{J}\in R)\wedge(\mathcal{J}\neq\mathcal{I})} \big[\beta_{\mathcal{J}} \cdot T_{\mathcal{I}}Node_{\mathcal{J}}(t_a)\big] \\ 1 &= \sum_{(\mathcal{J}\in R)\wedge(\mathcal{J}\neq\mathcal{I})} (\beta_{\mathcal{J}}) \end{aligned} \tag{4.2}$$

4.5.2 Trust-over-reputation Module

A node initialises its reputations (direct, observed and recommended) for other mobile nodes to at least Δ during the initial stages of the deployment. The trust-over-reputation module at node $\mathcal{I}$ retrieves the reputations (direct, observed and recommended) held for node $\mathcal{J}$ from the reputation-table via the reputation-update module, and then computes the trustworthiness of $\mathcal{J}$, $(T_{\mathcal{I}}Node_{\mathcal{J}}(t_{a+1}))$. As explained in detail in §4.7, the reputation-update module revises those reputations in proportion to the duration for which there has been no evidence owing to either mobility or the lack of communication. In (4.3), $\omega^{RR}_{\mathcal{I}\mathcal{J}}(t_a)$ refers to the reputation of type RR (direct, observed, or recommended) held for $\mathcal{J}$ by $\mathcal{I}$. Parameter $\gamma^{RR}_{\mathcal{I}\mathcal{J}}$ signifies the priority given to each type of reputation during the computation of a node's trustworthiness. As explained in the §4.6, the reputations are prioritised in the following order, (a) *direct*, (b) *observed* and, (c) *then recommended.* This is based on the belief that personal experiences (direct interactions and observations) take a higher precedence over the recommendations received from others, and direct interactions precede over those interactions observed between neighbours.

$$\begin{aligned} T_{\mathcal{I}}Node_{\mathcal{J}}(t_{a+1}) &= \sum \left[\gamma^{RR}_{\mathcal{I}\mathcal{J}} \cdot \omega^{RR}_{\mathcal{I}\mathcal{J}}(t_a)\right] \\ 1 &= \sum \left[\gamma^{RR}_{\mathcal{I}\mathcal{J}}\right] \end{aligned} \tag{4.3}$$

4.6 Detection Component

The detection component collects, quantifies, and represents the evidence as the reputation for each of the following types, (a) *direct*, (b) *observed* and, (c) *recommended.* In the following, we will explain the role of the reputation-capture and the reputation-evaluation modules for each type of reputations.

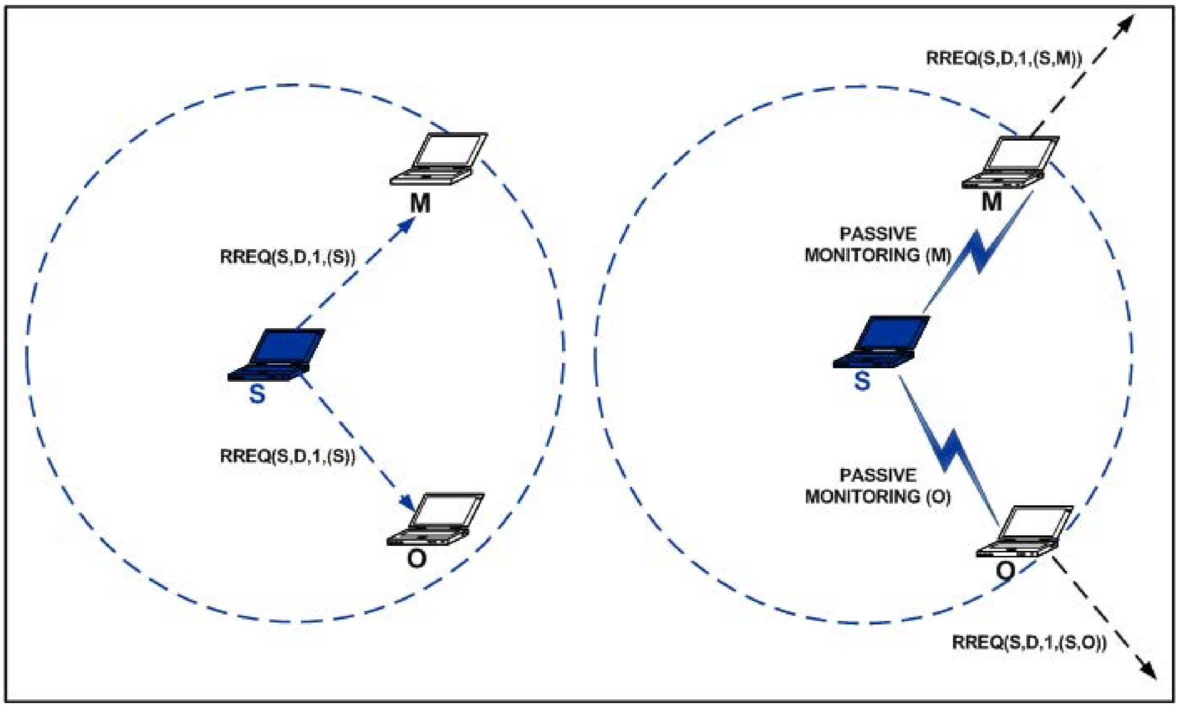


Figure 4.8: Reputation-capture: Evidence Collection for Direct Reputation.

4.6.1 Direct Reputation

Direct reputation is defined as the opinion held by a node for another node depending on the summary of evidence captured and quantified from their one-to-one interactions. As seen in Figure 4.8, the reputation-capture module collects evidence for the direct reputation through the passive monitoring-based IDS, where the overheard packet is verified against the duplicate packet stored in the packet-buffer. In the absence of link layer encryption that can be expensive, the payload can also be verified. Nevertheless, the plainness of the routing headers (regardless of encryption) for the routing purpose, in turn convenes the verification. The collected evidence is then quantified into a positive or negative value depending on whether it accounts for a benign or malicious behaviour, respectively. A positive (or negative) value, which accounts for the reputation of the latest evidence, is then passed on to the reputation-evaluation module for aggregation with the *accrued direct reputation.* As we will explain in detail in §4.7, prior to the aggregation, the reputation-update module revises the accrued direct reputation of a monitored node in proportion to the duration for which there has been no evidence.

The evidence is quantified into a positive value, *pos(event)*, only if the overheard packet (that relates to the downstream packet that was sent earlier to the next-hop) has been transmitted by the next-hop without any modification. The magnitude of the positive value depends on the type of event. Alternatively, a negative value, *neg(event, action)*, is assigned if the evidence confirms as a malicious behaviour. Unlike the positive value, the negative value not only depends on the type of event but also on the type of malicious action.

In case of misbehaviour, the malicious node responsible for the misbehaviour is inserted into the reject-list so that it can be excluded from the communication flow. As mentioned earlier in §4.4, the entry of malicious node into the reject-list during the route discovery cycle prevents the modified route from becoming the active route for the data flow. Similarly, enlisting the malicious node into the reject-list during the data flow triggers the propagation of the route error to the source of the communication flow. In both cases, the source is either triggered to choose an alternate trustworthy route (if there is one) or enforced to initiate a new route discovery.

As given in (4.4), the reputation-evaluation module at node $\mathcal{I}$ aggregates the reputation value for a latest evidence (*pos(event)* or *neg(event, action)*) with the *accrued direct reputation* ($\omega_{\mathcal{IJ}}^{Direct}(t_a)$) held for node $\mathcal{J}$. The result $\omega_{\mathcal{IJ}}^{Direct}(t_{a+1})$ becomes the accrued direct reputation for future computations. Prior to the computation shown in (4.4), $\mathcal{I}$ revises the accrued direct reputation of $\mathcal{J}$ in proportion to the duration for which there has been no direct interactions between them. This is achieved through the reputation-update module, which will be explained in detail in §4.7. Note that the quantification of the new evidence by $\mathcal{I}$ is independent of the opinion held for $\mathcal{J}$. This approach in combination with the limits -1 and $+1$, prevents the continual benign behaviours from concealing a future misbehaviour (or continual malicious behaviours from concealing a future benign behaviour). For example, the maximum direct reputation that $\mathcal{I}$ could set for the consistent benign behaviours of $\mathcal{J}$ is limited to $+1$, after which the continual benign behaviours of $\mathcal{J}$ fail to contribute. The saturated state denotes the absolute trustworthiness that $\mathcal{I}$ has towards $\mathcal{J}$ with respect to one-to-one

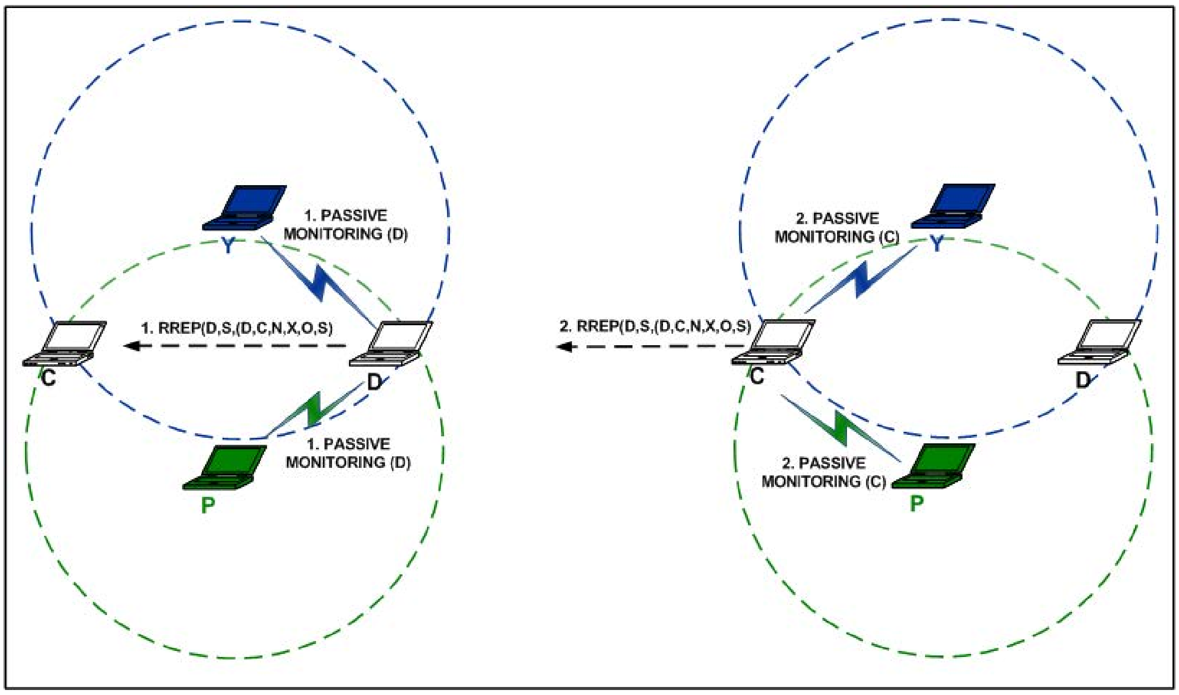

1. PASSIVE MONITORING (D)
Y
1. RREP(D,S,(D,C,N,X,O,S)
C
D
1. PASSIVE MONITORING (D)
P
2. PASSIVE MONITORING (C)
Y
2. RREP(D,S,(D,C,N,X,O,S)
C
D
2. PASSIVE MONITORING (C)
P

Figure 4.9: Reputation-capture: Evidence Collection for Observed Reputation.

interactions. However, an instance of malicious behaviour from $\mathcal{J}$ in the near future is sufficient to disintegrate such absolute trustworthiness. This design enables a node to detect compromised and selectively misbehaving nodes so that they can be excluded from sensitive communications.

$$\begin{aligned}
&\{\mathcal{J} \text{ exhibits misbehaviour}\} \equiv \\
&\qquad \left\{\omega_{\mathcal{IJ}}^{Direct}(t_{a+1}) = \max\left\{[-1],\ \left[\omega_{\mathcal{IJ}}^{Direct}(t_a) + neg(event,\, action)\right]\right\}\right\} \\
&\{\mathcal{J} \text{ exhibits benign behaviour}\} \equiv \\
&\qquad \left\{\omega_{\mathcal{IJ}}^{Direct}(t_{a+1}) = \min\left\{[+1],\ \left[\omega_{\mathcal{IJ}}^{Direct}(t_a) + pos(event)\right]\right\}\right\}
\end{aligned} \tag{4.4}$$

4.6.2 Observed Reputation

Observed reputation is defined as the opinion held by a node towards an observed node depending on the evidence captured and quantified from the observed node's behaviour towards a common neighbour[1]. The motivation for the observed reputation is derived from social psychology, where an individual's behaviour is observed whenever the individual deviates from normal behaviour. In turn, this demonstrates the psychology of the observers, who are primarily interested in remembering the individuals known for misbehaviours. The objective of the observers is to take advantage of their observations, so that they can be cautious when they interact with the individuals who are known for misbehaving. The definition of a normal behaviour may be subjective from the perspective of an observing individual even though a generic definition exists in terms of social laws. Observers fail to consider the individual's normal behaviour unless it is of direct benefit to them. Alternatively, observers do take into account the individual's extra-ordinary behaviours that are not expected as a part of normal behaviours. The objective of such observation is to identify the individuals that are likely to be trustworthy in a social context.

Consider the scenario from Figure 4.9, where nodes $\mathcal{P}$ and $\mathcal{Y}$ update the observed reputation of $\mathcal{C}$ based on the interactions observed between their neighbours $\mathcal{D}$ and $\mathcal{C}$. To begin with, $\mathcal{P}$ and $\mathcal{Y}$ overhear the packet forwarded by $\mathcal{D}$ to $\mathcal{C}$, and then the packet forwarded by $\mathcal{C}$ on behalf of $\mathcal{D}$. Nodes $\mathcal{P}$ and $\mathcal{Y}$ discard the observed evidence, if $\mathcal{C}$ has forwarded the packet without modification. From the perspective of $\mathcal{P}$ and $\mathcal{Y}$, $\mathcal{C}$ forwarding $\mathcal{D}$'s packet is not only an instance of normal behaviour, but also relatively insignificant. Furthermore, the decision to discard the evidence that was observed for normal behaviour assists in counteracting colluding attacks. Otherwise, $\mathcal{D}$ and $\mathcal{C}$ may exchange dummy packets between them to increase their observed reputation at $\mathcal{P}$ and $\mathcal{Y}$. An exception to this rule applies if $\mathcal{C}$ generates a route error whenever $\mathcal{D}$ becomes unreachable. Note that the creation of a route error can be considered as an extra-ordinary behaviour in a resource-constrained MANET. On the other hand, $\mathcal{P}$ and $\mathcal{Y}$ assign a negative value for $\mathcal{C}$, if $\mathcal{C}$ has performed a modification attack. The negative value is proportional to both the type of event and the attack. Node $\mathcal{C}$ is appended to their reject-list for exclusion until the completion of the corresponding communication flow. Node $\mathcal{C}$ not only loses direct reputation at its previous-hop $\mathcal{D}$

[1]Node positioned within the radio transmission range of both observing and observed nodes.

for each of its misbehaviours, but also the observed reputation at all the observing neighbours including $\mathcal{Y}$ and $\mathcal{P}$. In summary, penalising a malicious node at multiple nodes (interacting previous-hop $\mathcal{D}$ and multiple observing neighbours $\mathcal{Y}$ and $\mathcal{P}$) because of its misbehaviour discourages it from deviating from normal behaviour.

If node $\mathcal{J}$ has misbehaved, then the reputation-evaluation module at node $\mathcal{I}$ aggregates the reputation value of the latest evidence (*neg(event, action)*) with the *accrued observed reputation* ($\omega_{\mathcal{IJ}}^{Observed}(t_a)$) held for node $\mathcal{J}$, which is given in (4.5). However, if $\mathcal{J}$ exhibits a normal behaviour, then the accrued observed reputation of $\mathcal{J}$ remains unaltered. An exception applies to this rule if $\mathcal{J}$ generates a valid route error. Finally, the result $\omega_{\mathcal{IJ}}^{Observed}(t_{a+1})$ becomes the accrued observed reputation. Prior to the computation shown in (4.5), $\mathcal{I}$ revises its accrued observed reputation for $\mathcal{J}$ using the reputation-update module in proportion to the duration for which there has been no observed evidence.

$$\begin{aligned}
&\{\mathcal{J} \text{ exhibits misbehaviour}\} \equiv \\
&\quad \left\{\omega_{\mathcal{IJ}}^{Observed}(t_{a+1}) = \max\left\{[-1],\ \left[\omega_{\mathcal{IJ}}^{Observed}(t_a) + neg(event,\ action)\right]\right\}\right\} \\
&\{\mathcal{J} \text{ exhibits benign behaviour}\} \equiv \\
&\quad \left\{\omega_{\mathcal{IJ}}^{Observed}(t_{a+1}) = \min\left\{[+1], \left[\omega_{\mathcal{IJ}}^{Observed}(t_a) + pos(event, route\ error)\right]\right\}\right\}
\end{aligned} \tag{4.5}$$

4.6.3 Recommended Reputation

Recommended reputation is defined as the indirect opinion held by a mobile node towards another mobile node based on the derived recommendations. A mobile node that provides recommendations is referred to as a *recommender*. Likewise, a recommended node is referred as a *recommendee*, and a mobile node that receives recommendations is known as the *requesting node*. In general, most models [3, 39, 64, 66, 68–71, 82, 83, 107, 112, 116, 125, 134, 137, 140, 169, 178, 190, 204, 214, 220–222, 233–235, 239, 241, 242, 252, 261, 270–273, 292, 294, 301–305, 308–311, 327, 335, 347, 372, 373, 379, 380, 401, 404, 424, 425, 461, 468] communicate recommendations by disseminating additional packets or extra headers. However, these models may corrupt their decisions by using those recommendations received from other mobile nodes. First, they lack the ability to determine the bias of a recommender. Second, they are short of well-developed approaches for investigating the credibility of a recommender, such as whether the recommender exhibits honest-elicitation and (or) free-riding. In addition, the dissemination of the recommendations increases the overhead and degrades the network performance.

4.6.3.1 Processing Conventional Recommendations

Unlike approaches [38, 328] that eliminate the concept of recommendations, we believe that understanding the steps involved in disseminating the recommendations is vital to solve the associated issues. The disseminated recommendation can be defined as an opinion held by a recommender towards a recommendee, which is then forwarded by the recommender to a requesting node. The disseminated recommendation reflects the opinion of a recommender and it may summarise the evidence collected by the recommender from the direct interactions with the recommended node. Otherwise, it may also include the summary of the recommendations received by the recommender from other mobile nodes in favour of the recommended node. Since a disseminated recommendation presents the snapshot of the recommender's relationship with the recommended node, it is therefore reasonable to *deduce* from the disseminated recommendation whether the recommender will forward a packet for the recommended node in the near future, given that their relationship has not changed. This *deduction* applies only to identical contexts in which the disseminated recommendation applies. For instance, in the case of the routing, the context refers to the *forwarding of the packets without any modifications.*

The disseminated recommendation is then evaluated by the requesting node to revise the indirect opinion that is held for the recommended node. The reason for the evaluation is that the requesting node might not have witnessed those events from which the disseminated recommendation was created. Also, the disseminated recommendation reflects only the recommender's evaluation of those events. The requesting node accepts or rejects the disseminated recommendation based on the trustworthiness held

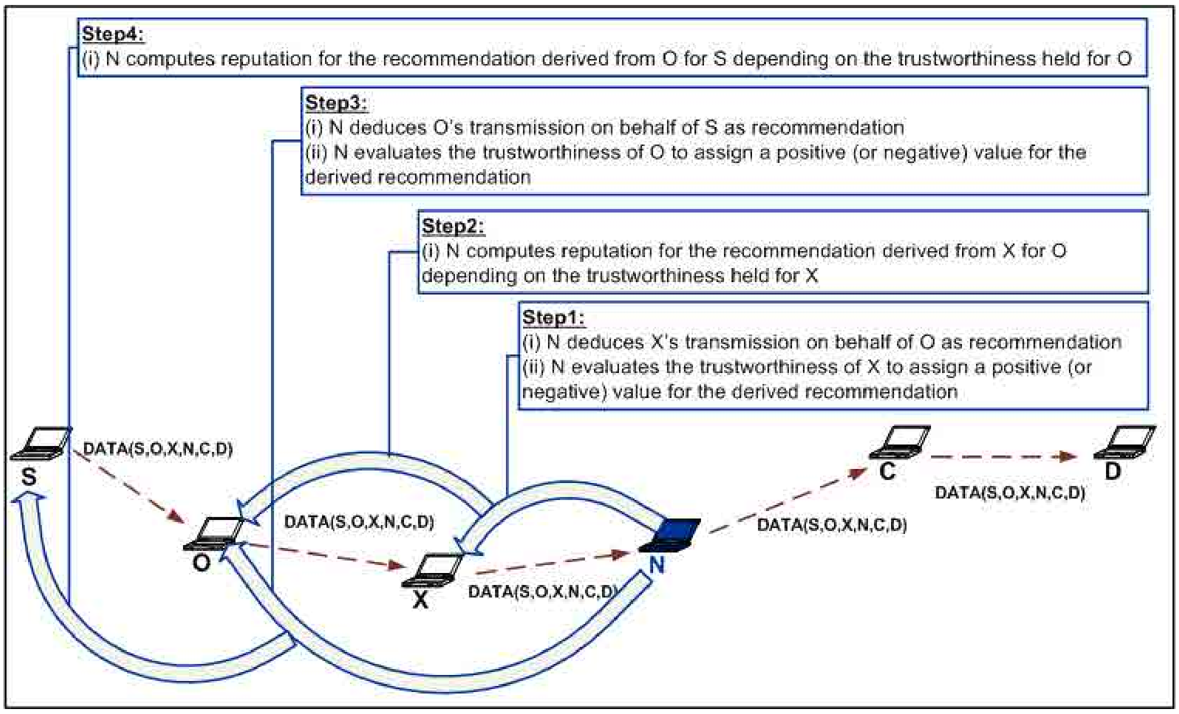

Step4:
(i) N computes reputation for the recommendation derived from O for S depending on the trustworthiness held for O
Step3:
(i) N deduces O's transmission on behalf of S as recommendation
(ii) N evaluates the trustworthiness of O to assign a positive (or negative) value for the derived recommendation
Step2:
(i) N computes reputation for the recommendation derived from X for O depending on the trustworthiness held for X
Step1:
(i) N deduces X's transmission on behalf of O as recommendation
(ii) N evaluates the trustworthiness of X to assign a positive (or negative) value for the derived recommendation
S
DATA(S,O,X,N,C,D)
O
DATA(S,O,X,N,C,D)
X
DATA(S,O,X,N,C,D)
N
DATA(S,O,X,N,C,D)
C
DATA(S,O,X,N,C,D)
D

Figure 4.10: Reputation-capture: Evidence Collection for Recommended Reputation.

for the recommender. However, if it accepts the disseminated recommendation, it then scales the disseminated recommendation in proportion to the level of trustworthiness held for the recommender.

4.6.3.2 Proposed Approach for Deriving Recommendations

As established above, it is reasonable to deduce from a disseminated recommendation whether the recommender will forward packets on behalf of a recommended node in the near future. In our approach, we inverse the deduction process to communicate the recommendations, such that those communicated recommendations are free from related issues.

Consider the scenario in Figure 4.10, where $\mathcal{X}$ unicasts a packet to $\mathcal{N}$, containing the route $\mathcal{S} \mapsto \mathcal{O} \mapsto \mathcal{X} \mapsto \mathcal{N} \mapsto \mathcal{C} \mapsto \mathcal{D}$. In this scenario, $\mathcal{N}$ (requesting node) *derives* an implicit recommendation for $\mathcal{O}$ (recommended node) from its previous-hop $\mathcal{X}$ (recommender), because $\mathcal{X}$ has forwarded the packet that was received from its previous-hop $\mathcal{O}$. Similarly, $\mathcal{N}$ derives the recommendation for $\mathcal{S}$ from $\mathcal{O}$, given that

$\mathcal{O}$ has forwarded the packet received from its previous-hop, $\mathcal{S}$. Node $\mathcal{N}$ validates the recommendation that is derived for $\mathcal{O}$ from $\mathcal{X}$ based on the fact that $\mathcal{X}$ has forwarded the packet because it has trusted its previous-hop ($\mathcal{O}$), next-hop ($\mathcal{N}$) and the communication flow ($\mathcal{S}$ and $\mathcal{D}$) (§4.4.4). Similarly, $\mathcal{N}$ validates those recommendations derived from other upstream nodes that can be traversed backwards along the route to the source of the packet. Since recommendations are derived from the route contained in a received packet, $\mathcal{N}$ evaluates the trustworthiness of its previous-hop ($\mathcal{X}$), the communication flow ($\mathcal{S}$ and $\mathcal{D}$) and the route (as detailed in §4.4) before deriving the recommendations from the route. Node $\mathcal{N}$ then computes its indirect opinion (the recommended reputation) for the recommended node from derived recommendation, depending on the trustworthiness held for the recommender.

Let us now consider the revision of the recommended reputation for the recommended node based on the derived recommendation. After deriving the recommendation for $\mathcal{O}$ from $\mathcal{X}$, node $\mathcal{N}$ then computes the trustworthiness of $\mathcal{X}$ using (4.3). Node $\mathcal{N}$ assigns a *positive* or *negative* value to the derived recommendation to demonstrate its viewpoint, depending on whether the trustworthiness of $\mathcal{X}$ is at least Δ. Note that $\mathcal{N}$'s positive (or negative) value for the recommendation derived for $\mathcal{O}$ from $\mathcal{X}$ is identical with the positive (or negative) value assigned to the rest of recommendations that were derived for $\mathcal{O}$ from other mobile nodes. Hence, $\mathcal{N}$ scales the positive (or negative) value that represents the recommendation derived for $\mathcal{O}$ from $\mathcal{X}$ using the trustworthiness held for $\mathcal{X}$. The resultant is the reputation value for the recent recommendation that is derived for the recommended node ($\mathcal{O}$), and the scaling affirms that the assigned reputation is proportional to the trustworthiness held for recommender ($\mathcal{X}$). Finally, $\mathcal{N}$ revises the *accrued recommended reputation* held for $\mathcal{O}$ by aggregating it with the reputation of the latest recommendation. A similar operation is then carried out for the recommendation derived from $\mathcal{O}$ for $\mathcal{S}$.

Equation (4.6) summarises the above-mentioned operations at time t_{a+1}, in which node $\mathcal{I}$ aggregates the accrued recommended reputation ($\omega^{Rec}_{\mathcal{I}\mathcal{J}}(t_a)$) with the reputation computed from the latest recommendation $\left[T_{\mathcal{I}}Node_{\mathcal{H}}(t_a) * (packet)\right]$ or $\left[T_{\mathcal{I}}Node_{\mathcal{H}}(t_a) * (packet)\right]$ to arrive at the result $\omega^{Rec}_{\mathcal{I}\mathcal{J}}(t_{a+1})$, where the latest recommendation is derived for node $\mathcal{J}$ from node $\mathcal{H}$. Later, the result becomes the accrued recommended reputation for future computations. Prior to the computation shown in (4.6), $\mathcal{I}$ revises the accrued recommended reputation of $\mathcal{J}$ using the reputation-update module in proportion to the duration for which there has been no recommendation.

$$\begin{aligned}
&\big[T_{\mathcal{I}}Node_{\mathcal{H}}(t_a) \geqslant \Delta\big] \equiv \\
&\qquad \Big\{\omega^{Rec}_{\mathcal{I}\mathcal{J}}(t_{a+1}) = min\Big\{[+1],\ \big[\omega^{Rec}_{\mathcal{I}\mathcal{J}}(t_a) + T_{\mathcal{I}}Node_{\mathcal{H}}(t_a) \cdot pos(event)\big]\Big\}\Big\} \\
&\big[T_{\mathcal{I}}Node_{\mathcal{H}}(t_a) < \Delta\big] \equiv \\
&\qquad \Big\{\omega^{Rec}_{\mathcal{I}\mathcal{J}}(t_{a+1}) = min\Big\{[-1],\ \big[\omega^{Rec}_{\mathcal{I}\mathcal{J}}(t_a) - T_{\mathcal{I}}Node_{\mathcal{H}}(t_a) \cdot neg(event)\big]\Big\}\Big\}
\end{aligned} \tag{4.6}$$

In our approach, recommendations are primarily derived from the route, provided the previous-hop, route and packet are trustworthy. However, our analysis reveals that the proposed approach is not foolproof, because any one of the upstream nodes (prior to the node that derives recommendations) can maliciously modify the route between its position and the source node. In such a situation, the downstream nodes positioned after the malicious node in the route derive recommendations from the modified route and eventually corrupt their accrued recommended reputation. Notably, the malicious node's previous-hop discards the corresponding route reply received from the malicious node, because the previous-hop would have enlisted the malicious node in its reject-list based on the captured directed evidence. Similarly, the malicious node's neighbours would have enlisted the malicious node in their reject-list after capturing the observed evidence. Thus, the modified route is prevented from becoming an active route for the corresponding communication flow. Therefore, it is apparent that the reputation-capture module's IDS validates the captured direct and observed evidence by referring the overheard packet with the buffered packet. In comparison with the direct and observed evidence, it is noticeable that the reputation-capture module lacks the reference to validate the evidence captured for recommended reputation. To achieve this, the recommendations are not derived until the route is valid. Therefore, the recommendations are derived only once for a communication flow, that is, either during the start or at the end of data flow, for which the route has to be error-free and hence trustworthy, as explained in §4.4.3.

In summary, the proposed approach prevents a node's recommended reputation from being corrupted by a recommender's recommendation, and this, in turn, encourages the node to believe only in its decisions. Hence, the node better resolves the issues concerned with the recommender's bias. Given that the recommenders do not disseminate their opinions as recommendations, they are eventually prevented from exhibiting both honest-elicitation and free-riding behaviours.

4.7 Reputation-update

In the MANET, the mobility-induced dynamically changing topology and the absence of a communication flow has an impact on the trust relationships established between the mobile nodes. This is because the nodes can change their behaviour during the period of separation or in the absence of a communication flow. For example, when a node moves away from a neighbour (owing to mobility), it is unclear whether it must consider the neighbour with the same level of opinion during the next interaction (when it returns). It may be that the neighbour could retain its current behaviour that could be either benign or malicious. Assuming that the neighbour is benign and given that the MANET is prone to inherent issues, then there is a chance for the neighbour to be compromised prior to the next interaction. Otherwise, if the neighbour is malicious, then it may be repenting and expecting an interaction to improve its relationship. Another possibility is that the neighbour may be malicious as a result of a past compromise, and it may be redeemed prior to next interaction.

The reputation-update module adopts a heuristic approach to resolve the uncertainty in the trust relationships that may result from the absence of behavioural evidence between the mobile nodes. The reputation range $[-1, +1]$ is divided into eight equal segments $\{S_1, S_2, \cdots, S_7, S_8\}$ of length, $\theta = 0.25$, such that $\{+1 \geqslant S_1 > 0.75, \cdots, 0.25 \geqslant S_4 > 0, 0 > S_5 \geqslant -0.25, \cdots, -0.75 > S_8 \geqslant -1\}$. Mobile nodes then

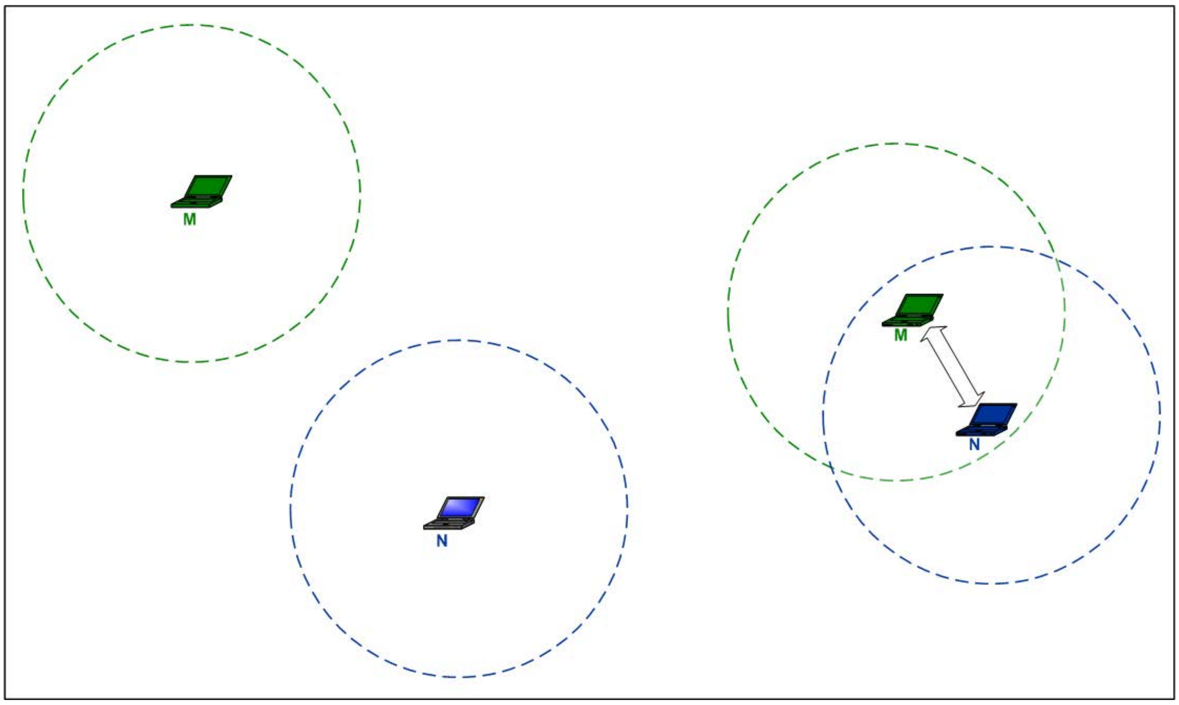


Figure 4.11: Reputation-update: Effect of Mobility on Reputations.

pessimistically decrement or optimistically increment the accrued reputation (direct, observed or recommended) for other mobile nodes for which there has been no evidence. However, the increment or decrement is contained in the segment to which the accrued reputation belongs.

Consider the scenario from the first half of Figure 4.11, in which the nodes $\mathcal{N}$ and $\mathcal{M}$ are positioned beyond each other's radio transmission range (lack of evidence for direct and observed reputations) and have no common communication flow (lack of evidence for recommended reputation). From the last half of Figure 4.11, let us assume that $\mathcal{N}$ and $\mathcal{M}$ have moved into each other's transmission range or a communication flow now exists through them. Now, $\mathcal{N}$ has to revise one of the reputations (direct, observed and recommended) held for $\mathcal{M}$, based on the newly-collected behavioural evidence or to evaluate the trustworthiness of $\mathcal{M}$ to send a packet to or accept a packet from $\mathcal{M}$. In such conditions, the reputation-update module of $\mathcal{N}$ first evaluates the accrued reputation of $\mathcal{M}$ (direct, observed or recommended) against Δ.

Node $\mathcal{M}$ is considered trustworthy by $\mathcal{N}$ if its accrued reputation at $\mathcal{N}$ is at least Δ.

This leads $\mathcal{N}$ to *decrement* its accrued reputation for $\mathcal{M}$ in proportion to the duration for which there has been no behavioural evidence. However, this decrement is limited to the *floor* of the segment S_i to which the accrued reputation belongs. For example, if the accrued reputation of $\mathcal{M}$ is 0.73, then the decrement is limited by the floor of the segment S_2 (> 0.50) to which it belongs. Assuming that $\mathcal{M}$ has been compromised at the time of separation, then the adopted decrement approach will unequivocally push down the trustworthiness of $\mathcal{M}$ to the next lower segment S_{i+1}, even for a single misbehaviour. However, if $\mathcal{M}$ has retained its benign behaviour, then the limit applied to the decrement approach ensures that the trustworthiness of $\mathcal{M}$ is still within the same segment S_i, and allows $\mathcal{M}$ to increase the trustworthiness at $\mathcal{N}$ by continual benign behaviours.

Similarly, if $\mathcal{M}$ is considered untrustworthy because the accrued reputation of $\mathcal{M}$ is below Δ, $\mathcal{N}$ then *increments* its accrued reputation for $\mathcal{M}$ in proportion to the duration for which there has been no behavioural evidence. However, this increment is limited to the *ceiling* of the segment S_i to which the accrued reputation belongs. For example, if the accrued reputation of $\mathcal{M}$ is -0.53, then the increment is limited to the ceiling of the segment S_7 (< 0.50) to which it belongs. Assuming that $\mathcal{M}$ has been redeemed at the time of separation, then the adopted increment approach will unambiguously push up the trustworthiness of $\mathcal{M}$ because of the subsequent continual benign behaviours. However, if $\mathcal{M}$ has retained its malicious behaviour, then the limit applied to the increment approach ensures that trustworthiness of $\mathcal{M}$ is still within the segment S_i and will lead the trustworthiness of $\mathcal{M}$ further down owing to persistent misbehaviours.

Given that $\mathcal{M}$ is unaware of the accrued reputations held by $\mathcal{N}$ and all the above-mentioned operations are carried out at $\mathcal{N}$, the node $\mathcal{M}$ is restricted from taking advantage of the above-mentioned set of operations. Those segments that are above and below Δ are the segregators for selectively-behaving benign and malicious nodes, respectively. For instance, if Δ is set to 0.50 then the segments, $0.75 \geqslant S_2 > 0.50$ and $0.50 \geqslant S_3 > 0.25$, can be used to segregate selectively-behaving benign and malicious nodes. The segment S_1 then contains the highly trustworthy nodes, while segments

$\{S_4, \cdots, S_8\}$ contain nodes with an increasing level of untrustworthiness.

Equation (4.7) presents the above-mentioned operations at time t_{a+1}, where $\omega^{RR}_{\mathcal{IJ}}(t_a)$ is the accrued reputation of type RR (direct, observed and recommended) held by node $\mathcal{I}$ for node $\mathcal{J}$. To arrive at the result $\omega^{RR}_{\mathcal{IJ}}(t_{a+1})$, (4.7) decrements, increments or maintains the accrued reputation depending on the evaluation against Δ and the duration for which $\mathcal{I}$ lacks evidence for the behaviours of $\mathcal{J}$. It defines a decrement and increment function based on the exponential function, where δ is known as the *reputation growth/decay factor*.

$$\begin{aligned}
&\Big\{\big[\omega^{RR}_{\mathcal{IJ}}(t_a)\, mod\, (\theta)\big] = 0\Big\} \equiv \\
&\qquad\qquad \big[\omega^{RR}_{\mathcal{IJ}}(t_{a+1}) = \omega^{RR}_{\mathcal{IJ}}(t_a)\big]; \quad 0 < \delta < 0.01; \quad \theta = 0.25; \\
&\qquad \big[\omega^{RR}_{\mathcal{IJ}}(t_a) \geqslant \Delta\big] \equiv \\
&\qquad\qquad \Big\{\omega^{RR}_{\mathcal{IJ}}(t_{a+1}) = min\Big\{\big[\omega^{RR}_{\mathcal{IJ}}(t_a)\cdot e^{-t\delta}\big],\ \big[\theta \cdot \lfloor \omega^{RR}_{\mathcal{IJ}}(t_a)/\theta \rfloor + \delta\big]\Big\}\Big\} \\
&\qquad \big[0 < \omega^{RR}_{\mathcal{IJ}}(t_a) < \Delta\big] \equiv \\
&\qquad\qquad \Big\{\omega^{RR}_{\mathcal{IJ}}(t_{a+1}) = max\Big\{\big[\omega^{RR}_{\mathcal{IJ}}(t_a)\cdot e^{t\delta}\big],\ \big[\theta \cdot \lceil \omega^{RR}_{\mathcal{IJ}}(t_a)/\theta \rceil\big]\Big\}\Big\} \\
&\qquad \big[\omega^{RR}_{\mathcal{IJ}}(t_a) < 0\big] \equiv \\
&\qquad\qquad \Big\{\omega^{RR}_{\mathcal{IJ}}(t_{a+1}) = max\Big\{\big[\omega^{RR}_{\mathcal{IJ}}(t_a)\cdot e^{-t\delta}\big],\ \big[\theta \cdot \lceil \omega^{RR}_{\mathcal{IJ}}(t_a)/\theta \rceil - \delta\big]\Big\}\Big\}
\end{aligned} \tag{4.7}$$

4.8 Limitations

In this section, we analyse the limitations of SMRTI including its, (a) *limited extension to the pure MANET*, (b) *constrained promiscuous monitoring-based IDS module*, (c) *restricted defence against route reply modification*, (d) *limited availability of the observed and recommended evidence* and, (e) *the ambiguity in choosing the initial reputation value for the other mobile nodes.*

4.8.1 Extension to the Pure MANET

In a managed MANET, centralised or distributed online or offline authorities are used to establish secret associations between the mobile nodes. Such online or offline authorities are unrealistic in a peer-to-peer civilian or a commercial pure MANET. As a result, the pre-existing secret associations are not feasible and hence it is difficult to initiate the security relationships after the deployment of self-organised mobile nodes in a pure MANET. The SMRTI well suits such a pure MANET because of its capability to establish relationships between the nodes depending on their behavioural evidence and then to make decisions based on the established relationships. Nonetheless, the decisions are vulnerable to imprecision for the reason that a pure MANET lacks the secret associations to authenticate the collected behavioural evidence. Therefore, the distinguished features of the SMRTI (§4.6) become infinitesimal in the presence of an unauthenticated behavioural evidence. This places the SMRTI on par with the trust management based models [72, 356, 433] that lack a well-defined approach to collect the evidence and formulate the trust relationship between the mobile nodes. Nevertheless, the SMRTI's features can still be effective for enhancing the security of a pure MANET, if an offline CA-based key management mechanism [167, 398] is deployed to complement the self-organised characteristic of the mobile nodes. Note that the SMRTI is also tailored for a managed MANET by integrating it with the key management and secure routing protocol. This will be presented in detail in Chapter 7.

4.8.2 Limitations on Promiscuous Monitoring

In [247], Marti *et al.* point out that the evidence collected through promiscuous monitoring-based IDS may be restricted by the, (a) *receiver collisions*, (b) *ambiguous collisions* and, (c) *variations in transmission power*.

4.8.2.1 Receiver and Ambiguous Collisions

We believe that the *virtual carrier-sensing mechanism* specified by IEEE 802.11 using the *Request-To-Send (RTS)* and *Clear-To-Send (CTS)* frames effectively resolves

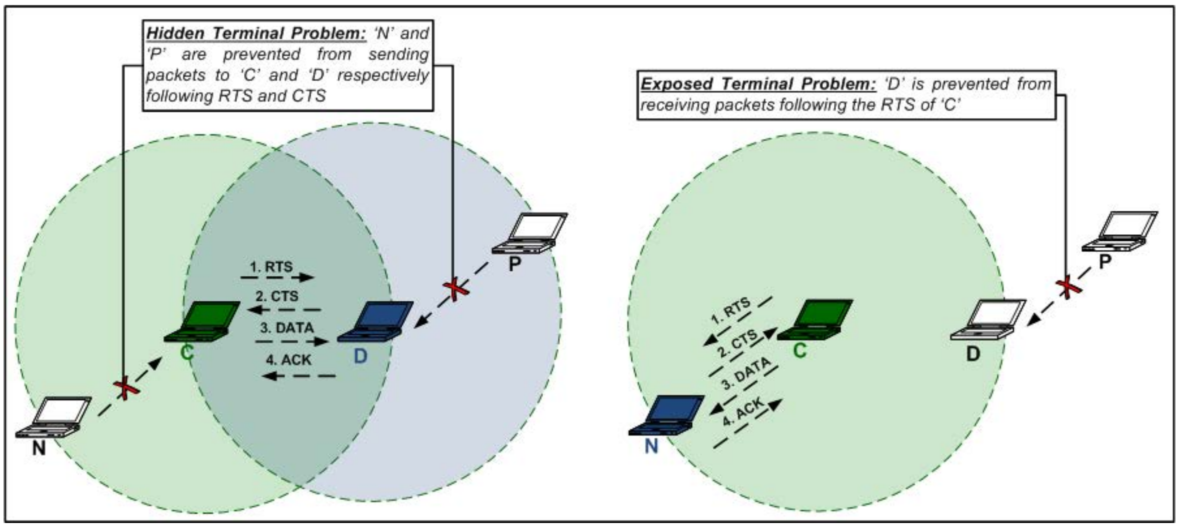

Hidden Terminal Problem: 'N' and 'P' are prevented from sending packets to 'C' and 'D' respectively following RTS and CTS
Exposed Terminal Problem: 'D' is prevented from receiving packets following the RTS of 'C'
1. RTS
2. CTS
3. DATA
4. ACK
N
C
D
P

Figure 4.12: RTS/CTS Defence against Receiver and Ambiguous Collisions.

both the receiver and the ambiguous collisions. For example, on the left hand-side of Figure 4.12, the RTS/CTS handshake reduces the probability of a collision in the receiver $\mathcal{D}$'s radio range. The CTS transmitted by $\mathcal{D}$ prevents a node $\mathcal{P}$ positioned in its range, but *hidden* from the transmitter $\mathcal{C}$, from colliding with its transmissions. This is because $\mathcal{P}$ reserves the transmission medium as busy until the end of communication, using the information extracted from the CTS frame. Since RTS/CTS handshake effectively eliminates the collision at receiver $\mathcal{D}$, the node $\mathcal{N}$ (previous-hop to $\mathcal{C}$) can assure that the promiscuously-overheard packets of $\mathcal{C}$ are also received by $\mathcal{D}$. Note that the initial RTS frame reduces the probability of a collision in the transmitter $\mathcal{C}$'s radio range. It prevents the nodes (*e.g.* $\mathcal{N}$) that are positioned in $\mathcal{C}$'s radio range from receiving packets from another node that is beyond $\mathcal{C}$'s radio range; the effect is known as an *exposed terminal problem.* Interestingly, the exposed terminal problem facilitates promiscuously-monitoring nodes from defending against ambiguous collisions. For instance, in the right hand-side of Figure 4.12, node $\mathcal{D}$, which is positioned in the transmitter $\mathcal{C}$'s radio range, is prevented from receiving a packet from another node $\mathcal{P}$ that is beyond the radio range of both the transmitter $\mathcal{C}$ and the receiver $\mathcal{D}$. Nevertheless, the RTS/CTS handshake holds only for unicasts and not for broadcasts such as route requests.

4.8.2.2 Varied Transmission Power, Directional Antenna and Mobility

Recently, several power-aware routing protocols [95, 126, 139, 197, 281, 402] have been proposed, based on the transmission power control (TPC) technique [205]. Since these protocols dynamically vary the transmission power of the nodes, some neighbouring nodes may not be able to promiscuously-overhear the transmissions of a node, even though they are positioned within the node's transmission power range. Another limitation of the promiscuous monitoring-based IDS is its non-compliance with the directional antenna to aid packet-overhearing. As described in Chapter 3, the mobility can also interrupt its operation when a receiving node moves away from the transmitter following packet reception and, for this reason, its operation is discouraged in the VANET.

4.8.2.3 Security of MAC and Physical Layers

In [84, 145, 207, 208, 319, 331], the authors present a range of DoS attacks that are launched at the MAC and the physical layers of the MANET. For completeness, we confine the situation to DoS attacks that are specific to the promiscuous monitoring-based IDS. For instance, the DoS attacks at the physical layer jam the wireless transmissions [84], while they violate the RTS/CTS handshake at the MAC layer to induce collisions. In addition, adversaries can vary the transmission power at their physical layer, even in the absence of a power-aware routing protocol to disrupt the promiscuous operation. For example, in the right hand-side of Figure 4.12, let us assume that $\mathcal{C}$ modifies the upstream packet received from $\mathcal{D}$ before forwarding the packet to the next-hop $\mathcal{N}$. In such a condition, $\mathcal{C}$ can deceive the promiscuously-monitoring $\mathcal{D}$ and most of the observing neighbours by taking advantage of the mobility and varying its transmission power to reach only $\mathcal{N}$. Finally, the mobility can also aid the adversaries to defeat the promiscuous monitoring but with a *non-cooperation tag*, as mentioned in Chapter 3.

The focus of our thesis is on the network layer security and that the security of the MAC and physical layers is beyond the scope of our thesis. However, we rely on the security solutions [84, 135, 319] that have been proposed for the MAC and the physical layers to date, despite the fact they are incomplete against the above-mentioned lower layer DoS attacks.

4.8.2.4 Discussion

In summary, the promiscuous monitoring-based IDS is confined to, (a) *unicasted data packets*, (b) *uniform transmission power*, (c) *omni-directional antenna*, (d) *less mobile nodes* and, (e) *secured MAC and physical layers.* Likewise, in the related trust management models [64–72, 301–305, 308–311], the SMRTI is also constrained by the limitations of the promiscuous monitoring-based IDS for the evidence collection. In other words, the promiscuous monitoring-based IDS confines the SMRTI to the behavioural evidence that is only detectable, discernible and classifiable. Nonetheless, the SMRTI differs from the related models in, (a) *establishing an effective trust relationship between the nodes* and, (b) *making efficient trust-based decisions for every context, but within the boundaries of the MANET's inherent issues (Chapter 2).* In addition, it is also the effect of several promiscuous monitoring-based trust models that have propelled and motivated us to build a trust model that can not only meet their shortcomings but also measure the maximum value proportion of such promiscuous operations. The next Chapter 5 will present the empirical study of the SMRTI that deploys the promiscuous monitoring-based IDS for evidence collection. However, we anticipate that the SMRTI's flexible design to upgrade the IDS module and also the ability to collect evidence from other security systems would lead to the improvement of a more advanced lightweight IDS. This we leave it for our future work. A snapshot of the SMRTI's capability to collect from other security systems is presented in Chapter 7, where the SMRTI is integrated with the key management and secure routing protocols.

4.8.3 Constrained Defence against Route Reply Modification

Although the SMRTI effectively discerns and classifies the detectable modification attacks, it is restrained from efficiently defending against the modification of the route

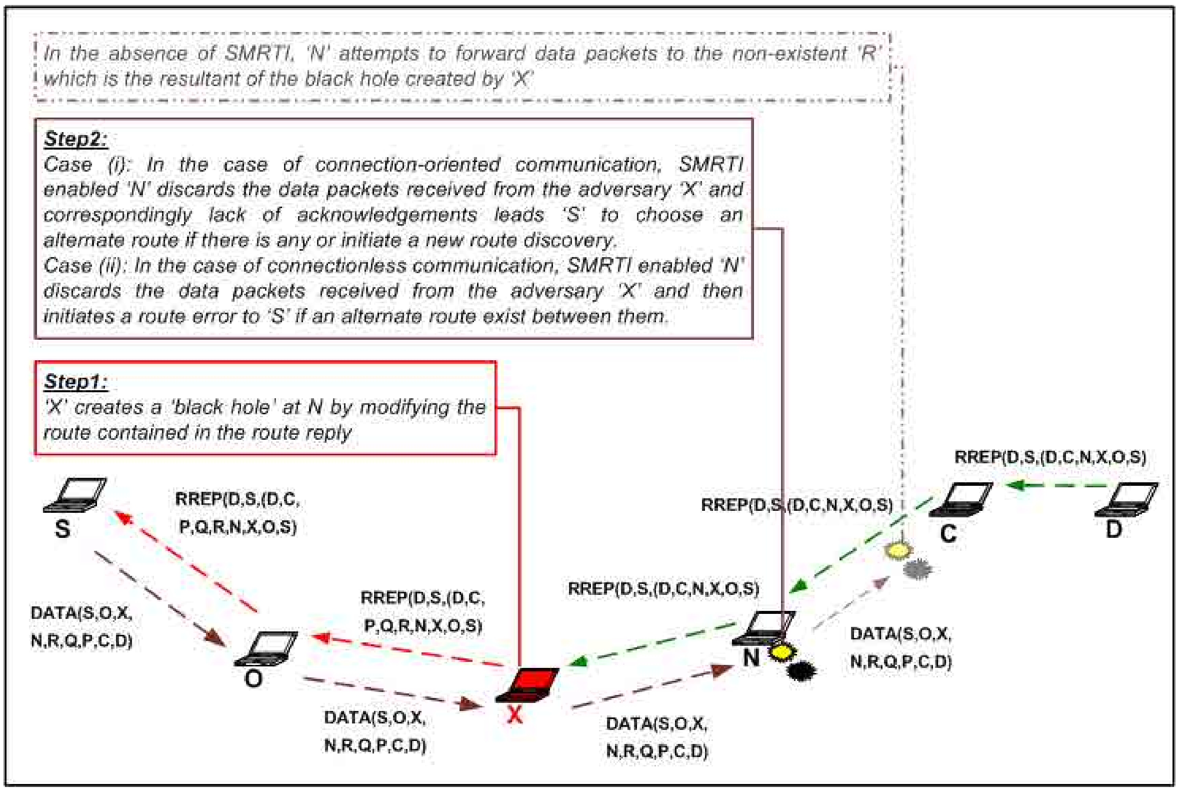

In the absence of SMRTI, 'N' attempts to forward data packets to the non-existent 'R' which is the resultant of the black hole created by 'X'
Step2:
Case (i): In the case of connection-oriented communication, SMRTI enabled 'N' discards the data packets received from the adversary 'X' and correspondingly lack of acknowledgements leads 'S' to choose an alternate route if there is any or initiate a new route discovery.
Case (ii): In the case of connectionless communication, SMRTI enabled 'N' discards the data packets received from the adversary 'X' and then initiates a route error to 'S' if an alternate route exist between them.
Step1:
'X' creates a 'black hole' at N by modifying the route contained in the route reply
S
RREP(D,S,(D,C, P,Q,R,N,X,O,S)
DATA(S,O,X, N,R,Q,P,C,D)
O
RREP(D,S,(D,C, P,Q,R,N,X,O,S)
DATA(S,O,X, N,R,Q,P,C,D)
X
RREP(D,S,(D,C,N,X,O,S)
DATA(S,O,X, N,R,Q,P,C,D)
N
RREP(D,S,(D,C,N,X,O,S)
DATA(S,O,X, N,R,Q,P,C,D)
C
RREP(D,S,(D,C,N,X,O,S)
D

Figure 4.13: Constrained Defence against Route Reply Modification Attack.

reply header. As discussed earlier in §4.6.2, an intermediate node enabled with the SMRTI and positioned previous to the adversary during a route reply propagation, would then be able to capture the route modification performed by the adversary. Hence, this would cause the intermediate node to enlist the adversary into its reject-list. However, the intermediate node is restrained from notifying the route reply's target (the source of the communication flow) with a route error, because the target is positioned after the adversary on the route. In such a situation, the defence carried out by the SMRTI-enabled intermediate node depends on the type of transport protocol deployed for the communication flow.

In the case of the connection-oriented communication, the SMRTI prevents the intermediate node from receiving any data packets from the adversary during the subsequent data flow. For this reason, the source of the communication flow (the route reply's target) fails to receive the acknowledgements for the disseminated data packets. The source node then propagates data packets along an alternate route (if any exist) or initiates a new route discovery for the communication flow.

Alternatively, in the case of a connectionless communication, the lack of acknowledgements prevents the intermediate node from informing the source regarding the corrupted route and hence the deterred data flow. The SMRTI performs its best to notify the source by instructing the intermediate node to initiate a route error to the source, provided that an alternate route exists between them. In the absence of an alternate route, the SMRTI allows the intermediate node to discard those data packets received through the adversary. Such a decision, at the least, prevents the downstream nodes from saving the battery resource; otherwise they would be propagating data packets that would be dropped at the black hole created by the modified route.

Let us consider the scenario presented in Figure 4.13, in which the adversary $\mathcal{X}$ modifies the route that is contained in the route reply. As expected, the SMRTI-enabled $\mathcal{N}$ captures the evidence for the route modification and accordingly enlists $\mathcal{X}$ into its reject-list. Although $\mathcal{S}$ propagates data packets along the modified route $\mathcal{S} \mapsto \mathcal{O} \mapsto \mathcal{X} \mapsto \mathcal{N} \mapsto \mathcal{R} \mapsto \mathcal{Q} \mapsto \mathcal{P} \mapsto \mathcal{C} \mapsto \mathcal{D}$, the node $\mathcal{N}$ fails to accept the data packets from $\mathcal{X}$, based on the entry in its reject-list. Assuming that the deployed transport protocol is connection-oriented, $\mathcal{S}$ then chooses an alternate route (if there is one) or initiates a new route discovery, based on the absence of the acknowledgements for the propagated data packets. Alternatively, if the connectionless transport protocol is deployed, then $\mathcal{S}$ implicitly fails to sense the route modification. However, $\mathcal{N}$ informs $\mathcal{S}$ through a route error if an alternate route to $\mathcal{S}$ exists. Given that $\mathcal{N}$ is connected to $\mathcal{S}$ only through the modified route $\mathcal{S} \mapsto \mathcal{O} \mapsto \mathcal{X} \mapsto \mathcal{N} \mapsto \mathcal{R} \mapsto \mathcal{Q} \mapsto \mathcal{P} \mapsto \mathcal{C} \mapsto \mathcal{D}$, $\mathcal{N}$ then continues to discard the data packets received from $\mathcal{X}$, rather than attempting to propagate those packets to a non-existent $\mathcal{R}$.

4.8.4 Limited Availability of Observed and Recommended Evidence

In comparison with the direct evidence (§4.6.1), a relatively low count of observed and recommended evidence is captured for the following reasons. In the case of the observed evidence, an observing node is required to be positioned in the common radio range of the interacting neighbours. On the other hand, a node is required to participate in many communication flows to derive the recommendations. Irrespective of these limitations, the collected observed or recommended evidence is precisely attributed to the observed behaviour or recommended relationship, respectively. This is because each of the observed or the recommended evidence is evaluated independently without allowing past evaluations or reputations to impact the evaluation. However, merging the recent evaluation with past evaluations enables the node to draft a portrait of the observed or recommended node. Note that the precision of the portrait depends on the proportion of the evidence collected so far the observed or recommended node.

In addition, the observed and recommended evidence complement each other, as discussed below. Although the observing nodes can capture the evidence for both malicious and benign behaviours, they consider only the malicious behaviours as a precautionary measure against the malicious nodes and disregard the benign behaviours to defend against the colluding attacks. Similarly, the recommendations complement the observations, because the recommended evidence is derived only from the trustworthy nodes. However, the derived recommendations are attributed to either positive or both positive and negative values, depending on the threshold-limit set for the packet propagation and the derived recommendations. Let us assume Δ_c is the threshold-limit for evaluating the trustworthiness of the various contexts (§4.4) and Δ_r is the threshold-limit for evaluating derived recommendations. In this particular instance, the intermediate nodes propagate through an active route only when the trustworthiness of the contexts evaluate to Δ_c. Since the intermediate nodes derive recommendations only from a trustworthy route during the data flow (§4.6.3.2), the condition $\Delta_r \leqslant \Delta_c$ is refined to $\Delta_r = \Delta_c$. This is because Δ_c is the minimum trustworthiness defined by the intermediate nodes to propagate those data packets received from the upstream nodes, *i.e.* recommenders. Henceforth, the derived recommendations always meet Δ_r and evaluate to a positive value. Alternatively, the condition $\Delta_r > \Delta_c$ assigns either a positive or negative value to the derived recommendations, depending on whether or not the trustworthiness held for the recommenders is at least Δ_r, respectively. Furthermore, this scenario highlights the capability of the SMRTI to assign a specific threshold-limit to each of the contexts, and the direction for the threshold-limit assignment depends on how the order of importance and sensitivity exist among those contexts.

4.8.5 Initial Arbitrary Reputations

Let us consider the instance of a new node joining the network, where the existing nodes in the network may not have a record of past evidence to trust or distrust the newly-joining node. In such a situation, assigning an arbitrary level of trust for the new node poses several issues. In general, trust models resolve this issue by either pessimistically assigning a neutral or low level of trust or optimistically assigning a high level of trust to the new node. The purpose of the pessimistic approach is to compel the new node to exhibit a consistent benign behaviour from the point it enters the network. However, in some of these models, it is not always clear how the less trusted new node is selected for communications when the nodes with high trust values exist in the network. If a new node is not preferred for communications because of its low level of trust, then it lacks the opportunity to gain trust with the existing nodes. Alternatively, with an optimistic approach, the intention is to promptly identify whether the new node exhibits malicious behaviour from the point it enters into the network. We favour the optimistic approach when prompt identification is feasible, because the high level of trust is assigned to a newly-joining node decreases rapidly as the malicious behaviour increases. However, the optimistic approach fails to discriminate a new node from the existing nodes, whose dynamically changing selective behaviour has warranted the same level of trust. In synopsis, this issue arises when the trust models explicitly fail to represent an existing node's ignorance about a newly-joining node's behaviour. Note that the issue also extends to nodes that have already established a trust relationship with one another. For example, when a node moves away from a neighbour, it is unclear whether to consider the neighbour with the same level of trust or distrust during the next interaction (when it returns). Recollect that we solve the issues resulting from mobility or absence of evidence by either increasing or decreasing the trustworthiness of the nodes (§4.7). Nonetheless, the seriousness of this approach is that the trustworthiness of other mobile nodes only accounts for the probability of their benign or malicious behaviour and barely represents ignorance. Hence, failing to explicitly represent the notion of ignorance and the associated uncertainty has a fundamental impact on the trust model. All these have motivated us to propose a *subjective-logic based SMRTI model* in Chapter 9, so that a node can explicitly represent and manage the uncertainty in its relationship with the other nodes.

4.9 Conclusion

Although the SMRTI is subject to the above limitations, these limitations are uniform for all trust models. Furthermore, the SMRTI outperforms those related models by resolving their shortcomings and operating within the inherent features of the MANET. The SMRTI is adaptable for all types of routing protocols with minor alterations and suits both the open and managed networks. For any protocol to feed the information and trust-related queries into the SMRTI and to receive trust-based decisions in return from the SMRTI, the protocol must need interfaces. For this reason, some minor adaptations were made in the DSR protocol.

The SMRTI operates as a decentralised and independent module at every node, where it collects three types of evidence, (a) *direct*, (b) *observed* and, (c) *recommended*, to draft the portrait of other mobile nodes. Direct evidence enables the SMRTI-enabled nodes to identify the malicious and benign nodes, based on direct interactions, while the observed evidence facilitates the SMRTI-enabled nodes to shortlist the malicious nodes even before interacting with them. Alternatively, the recommended evidence allows the SMRTI-enabled nodes to deduce trustworthy relationships that exist between the other nodes. The unique approach adopted by the SMRTI to deduce recommendations assists the SMRTI in eliminating, (a) *honest-elicitation*, (b) *free-riding*, (c) *recommender's bias* and, (d) *additional message dissemination.* The SMRTI collects direct and observed evidence through the promiscuous monitoring-based IDS and evaluates the evidence from each before accepting it. The de-coupled design of the evidence collection and the formulation facilitates the SMRTI-enabled nodes to upgrade their

IDS to a comprehensive lightweight IDS because of the availability or whenever there is a necessity to collect evidence for emergent attacks.

The SMRTI represents its opinion for other nodes by formulating each type of evidence collected for those nodes as a reputation rating. In sequence, it combines a recent reputation with the past reputation held for those nodes. The limits $[-1, +1]$ applied to the reputation rating ensure that SMRTI handles the *reputation-saturation problem* by preventing the evidence captured for persistent benign or malicious behaviour from concealing the future malicious or benign behaviours, respectively. The SMRTI also provides more weightage to a recent evidence captured for those nodes to prevent them from taking advantage of their past benign behaviours. It also prevents the evidence collected for a recent benign behaviour from over-riding the past record of the malicious behaviours. For this reason, the SMRTI classifies erratically-behaving nodes into a separate category, known as selectively-behaving benign or malicious nodes. In synopsis, the SMRTI uses the reputations (direct, observed and recommended) to arrive at the portrait of those nodes and to establish a relationship with them. Furthermore, it also resolves the uncertainty introduced into a trust relationship owing to mobility or the absence of evidence. The corollary of this approach leads to the accommodation of repenting malicious nodes or also allowing the redemption of the compromised nodes.

The SMRTI uses the policies for the various contexts and trust relationships that have been already established with other mobile nodes to make efficient decisions for those contexts. In addition, the SMRTI defines a policy to reactively isolate a misbehaved node from a communication flow by using the reject-list, regardless of the trustworthiness held for that misbehaved node. This approach enables the SMRTI to accommodate new contexts and accordingly to define efficient policies for counteracting emerging attacks. Since SMRTI-enabled nodes do not share their reputation ratings with other nodes, their decisions are never corrupted by the reputation ratings of the other nodes.

In summary, we have shown how the SMRTI is built on realistic assumptions and does not rely on a centralised authority or tamper-proof hardware to enhance the security of the MANET.

5 Performance Analysis of SMRTI

5.1 Introduction

In this chapter, we analyse the performance of the SMRTI using large-scale NS2-based simulations. These simulation results confirm that the SMRTI renders a promising solution for enhancing the security decisions in the MANET. This chapter is organised as follows. In the following, we describe the adversary model against which the performance of the DSR and the SMRTI nodes are tested. We then briefly discuss the NS2 simulator and present a detailed description of the simulation setup. Finally, we discuss and compare the performance of the SMRTI and the DSR nodes against the adversary model.

5.2 Adversary Model

As mentioned in Chapter 2, the objective of an adversary is to either disrupt the route establishment or the data flow. Here, we present an adversary model in which the adversary modifies the route headers to disrupt the route discovery and the data flow. Those modifications include, (a) *the increment of a route request's sequence number*, (b) *the deletion of the intermediate nodes from route* and, (c) *the insertion of the identity of the invalid nodes to the route.* In Chapter 7, we will consider the propagation of the spoofed packets by the adversaries. Recollect that the adversary model pertaining to the interruption attacks was detailed in Chapter 3. In summary, the adversary achieves its objective by either performing one of the above modifications or a combination of the modifications, such as the addition or deletion of the route and the increment of the route request's sequence number.

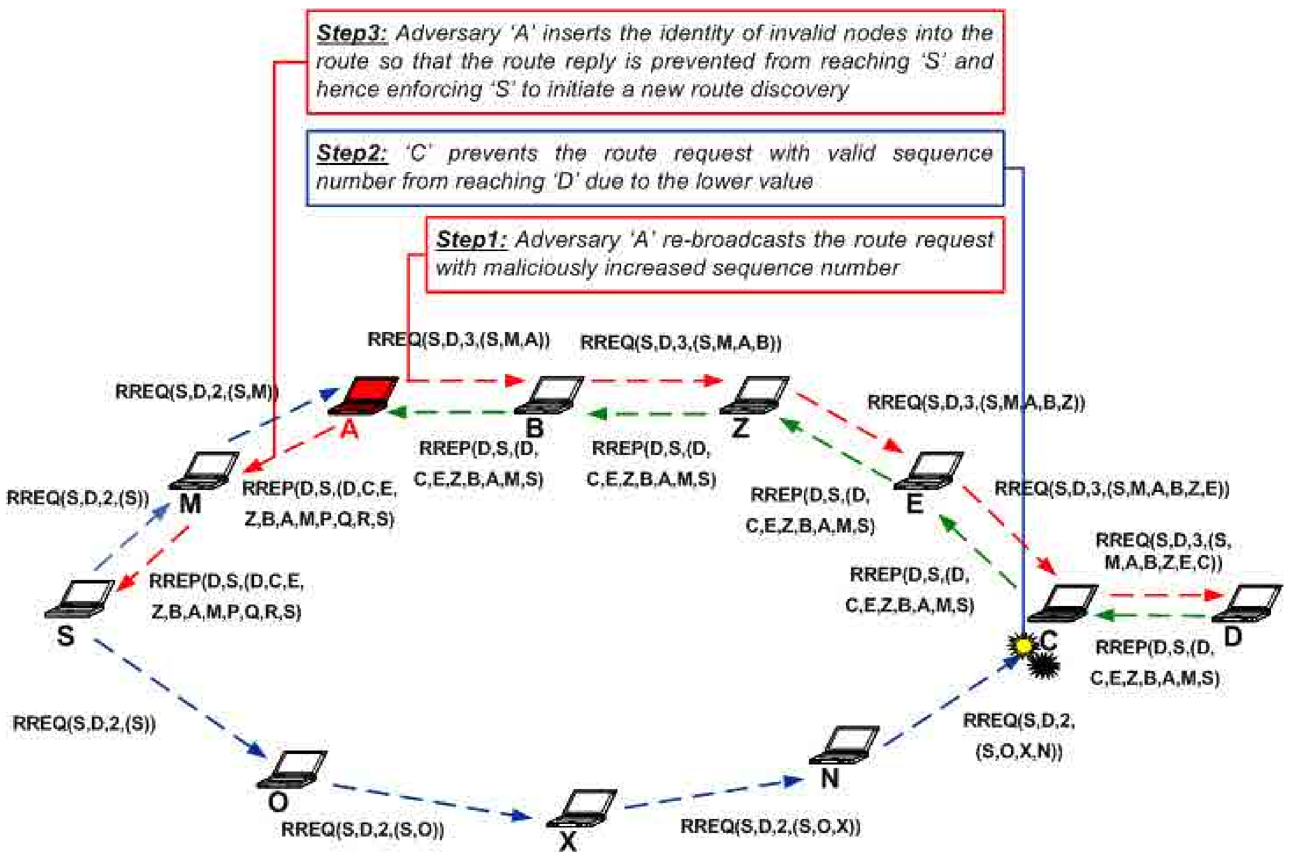
5.2.1 Route Discovery Disruption

The adversary disrupts a route discovery as follows. First, it broadcasts the route request received from a source with a maliciously-incremented sequence number and then responds back to the source with a route reply that contains the maliciously-inserted identity of the invalid nodes. Here, the broadcast of the malicious route request suppresses the propagation of the valid route requests that take the overlapping paths and belong to the same route discovery phase. In addition, the malicious route request prevents the destination from replying to the few valid route requests that might have travelled through the disjointed paths. However, the suppression of the valid route requests does not come easily, because the adversary is required to propagate the malicious route request either through a congestion-free path that has the least delay or a shorter path. The requirement for faster propagation is to establish the modified sequence number as the highest sequence number at all downstream nodes, including the destination. Henceforth, the downstream nodes are induced to discard the route requests that contain the valid sequence number as *stale route requests.* Remember that it is the design specification of the DSR protocol to record the route request with the highest sequence number as the fresh route request from a source.

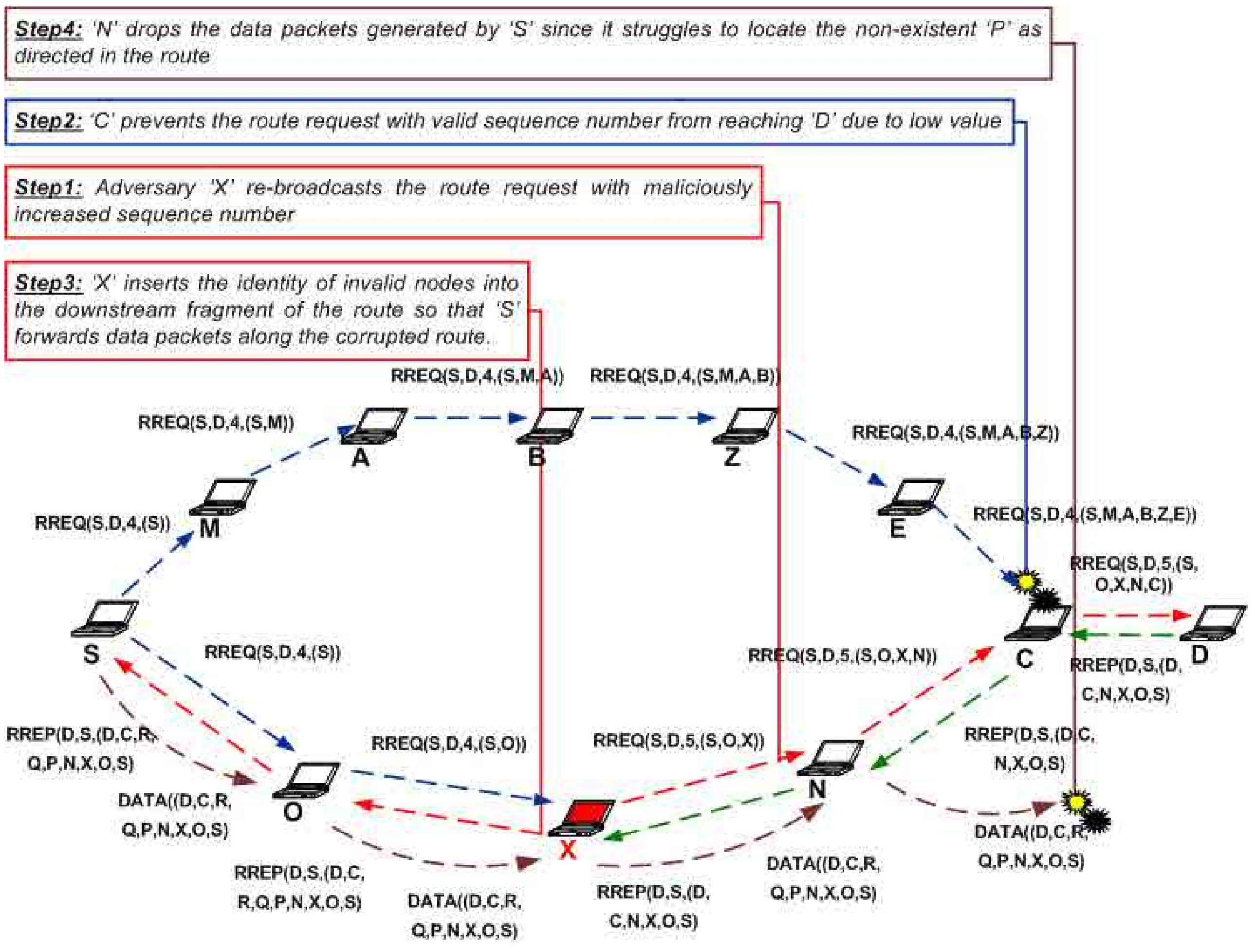
Later, the adversary either inserts the identity of the invalid nodes into or deletes the identity of the intermediate nodes from the downstream fragment of the route reply (the fragment of the route that is in between the adversary and the route request originator). Note that the adversary responds back with a tampered route reply for the propagated malicious route request. The tampered route then eventually prevents the downstream intermediate nodes from delivering the route reply to the route request's originator and, hence, triggers the route request's originator to initiate a new route discovery.

The increment to a route request's sequence number not only suppresses the propagation of the valid route requests that belong to the same route discovery cycle, but also suppresses the propagation of the valid route requests that belong to the future route discovery cycles. This impact persists until the source generates a route request with a sequence number that is higher than the malicious route request's sequence number. In such an instance, a vigilant adversary can repeat the above process to persistently prevent the source from discovering a route. As discussed in Chapter 2, a variation of the above attack can be perceived as the *detour attack* in which an adversary modifies the route headers to ensure that a specific set of nodes are never reached.

An instance of a route discovery disruption is shown in Figure 5.1(a), where the adversary $\mathcal{A}$ increments a route request's sequence number and propagates the route request to the destination $\mathcal{D}$ through a congestion-free path that has the least delay. Another route request that belongs to the same route discovery cycle and is propagated through the congested path $\mathcal{S} \mapsto \mathcal{O} \mapsto \mathcal{X} \mapsto \mathcal{N} \mapsto \mathcal{C}$ is dropped at $\mathcal{C}$, because $\mathcal{C}$ has already seen the malicious route request. Node $\mathcal{D}$ then replies to $\mathcal{S}$ through $\mathcal{A}$ with a route reply that contains the route $\mathcal{D} \mapsto \mathcal{C} \mapsto \mathcal{E} \mapsto \mathcal{Z} \mapsto \mathcal{B} \mapsto \mathcal{A} \mapsto \mathcal{M} \mapsto \mathcal{S}$. However, $\mathcal{A}$ modifies the route of the route reply into $\mathcal{D} \mapsto \mathcal{C} \mapsto \mathcal{E} \mapsto \mathcal{Z} \mapsto \mathcal{B} \mapsto \mathcal{A} \mapsto \mathcal{M} \mapsto \mathcal{P} \mapsto \mathcal{Q} \mapsto \mathcal{R} \mapsto \mathcal{S}$ to prevent $\mathcal{M}$ from delivering the route reply to $\mathcal{S}$. Another variation of the route modification that is not seen in this instance is the deletion of the valid intermediate nodes from the route. The timeout for the route discovery at $\mathcal{S}$


Step3: Adversary 'A' inserts the identity of invalid nodes into the route so that the route reply is prevented from reaching 'S' and hence enforcing 'S' to initiate a new route discovery
Step2: 'C' prevents the route request with valid sequence number from reaching 'D' due to the lower value
Step1: Adversary 'A' re-broadcasts the route request with maliciously increased sequence number
RREQ(S,D,3,(S,M,A))
RREQ(S,D,3,(S,M,A,B))
RREQ(S,D,2,(S,M))
A
B
Z
RREQ(S,D,3,(S,M,A,B,Z))
RREP(D,S,(D, C,E,Z,B,A,M,S)
RREP(D,S,(D, C,E,Z,B,A,M,S)
E
RREQ(S,D,3,(S,M,A,B,Z,E))
RREQ(S,D,2,(S))
M
RREP(D,S,(D,C,E, Z,B,A,M,P,Q,R,S)
RREP(D,S,(D, C,E,Z,B,A,M,S)
RREQ(S,D,3,(S, M,A,B,Z,E,C))
RREP(D,S,(D,C,E, Z,B,A,M,P,Q,R,S)
RREP(D,S,(D, C,E,Z,B,A,M,S)
S
C
D
RREP(D,S,(D, C,E,Z,B,A,M,S)
RREQ(S,D,2,(S))
RREQ(S,D,2, (S,O,X,N))
N
O
RREQ(S,D,2,(S,O))
X
RREQ(S,D,2,(S,O,X))

(a) Modification Attack – Disruption of Route Discovery


Step4: 'N' drops the data packets generated by 'S' since it struggles to locate the non-existent 'P' as directed in the route
Step2: 'C' prevents the route request with valid sequence number from reaching 'D' due to low value
Step1: Adversary 'X' re-broadcasts the route request with maliciously increased sequence number
Step3: 'X' inserts the identity of invalid nodes into the downstream fragment of the route so that 'S' forwards data packets along the corrupted route.
RREQ(S,D,4,(S,M,A))
RREQ(S,D,4,(S,M,A,B))
RREQ(S,D,4,(S,M))
A
B
Z
RREQ(S,D,4,(S,M,A,B,Z))
M
E
RREQ(S,D,4,(S,M,A,B,Z,E))
RREQ(S,D,4,(S))
RREQ(S,D,5,(S, O,X,N,C))
S
RREQ(S,D,4,(S))
RREQ(S,D,5,(S,O,X,N))
C
D
RREP(D,S,(D, C,N,X,O,S)
RREP(D,S,(D,C,R, Q,P,N,X,O,S)
RREQ(S,D,4,(S,O))
RREQ(S,D,5,(S,O,X))
RREP(D,S,(D,C, N,X,O,S)
N
DATA((D,C,R, Q,P,N,X,O,S)
O
X
DATA((D,C,R, Q,P,N,X,O,S)
RREP(D,S,(D,C, R,Q,P,N,X,O,S)
DATA((D,C,R, Q,P,N,X,O,S)
RREP(D,S,(D, C,N,X,O,S)
DATA((D,C,R, Q,P,N,X,O,S)

(b) Modification Attack – Disruption of Data Flow

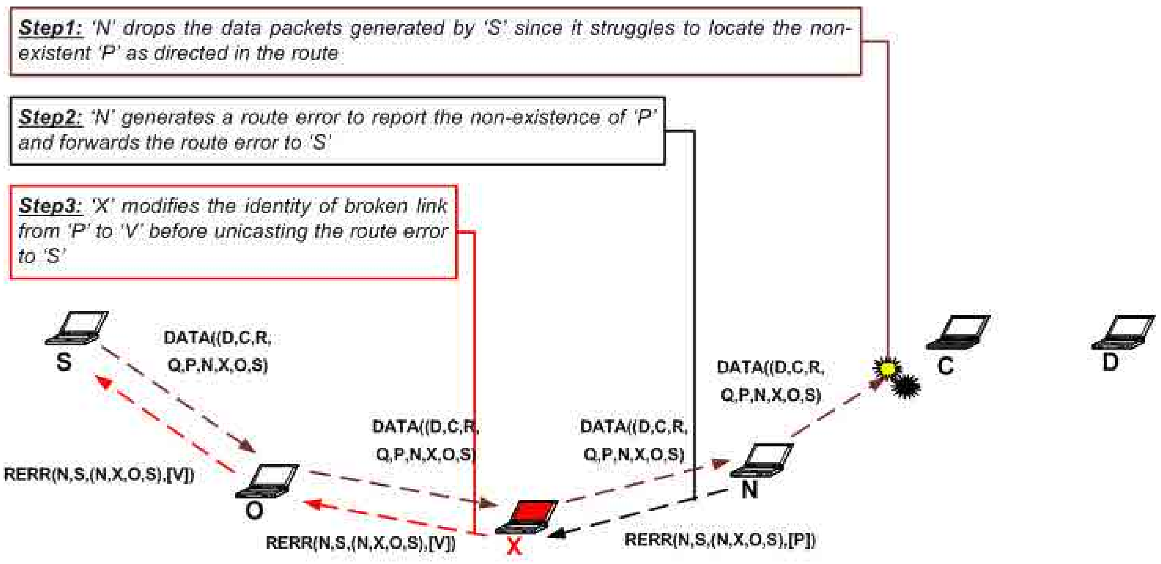
FIGURE 5.1: Route Discovery and Data Flow Attacks.

would eventually trigger $\mathcal{S}$ to initiate a new route discovery cycle, which accordingly meets the objective of $\mathcal{A}$, *i.e.* to prevent $\mathcal{S}$ from discovering a route to $\mathcal{D}$.

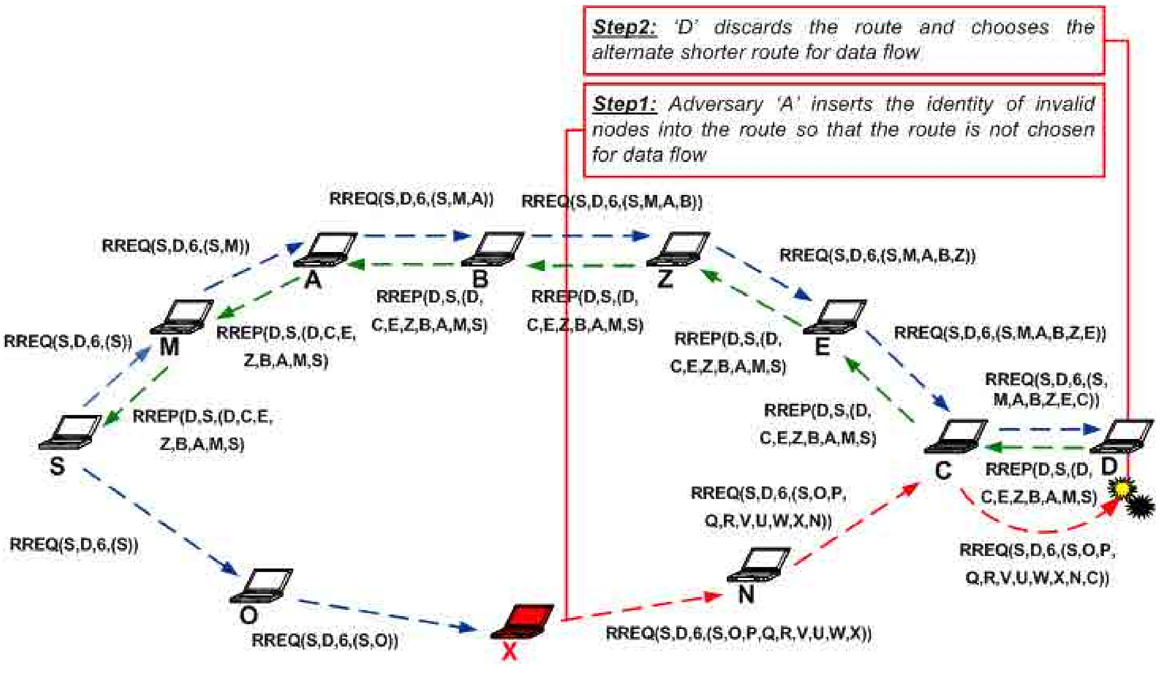
5.2.2 Data Flow Disruption

Similar to the route discovery disruption, the adversary disrupts the data flow by incrementing the route request's sequence number and modifying the route of the corresponding route reply. Incremented sequence number ensures that the route through which the malicious route request has been propagated becomes the active route for the data flow. Recall that the malicious route request must be propagated through a shorter path or a congestion-free with the least delay to suppress the propagation of the route requests with a valid sequence number. The adversary then modifies the route of the corresponding route reply, but differs from the route discovery disruption attack by modifying the upstream fragment of the route reply (the fragment of the route that is in between the adversary and the initiator of the route reply). The modification later triggers one of the nodes that is positioned between the adversary and destination to discard the data packets. As mentioned in Chapter 2, a variation of the above attack is the *blackhole attack*, in which an adversary modifies the route header to embed itself in the active route to drop the data packets. The *grayhole attack* is a variant of the blackhole attack in which the data packets are dropped selectively.

As seen in Figure 5.1(b), the adversary $\mathcal{X}$ increments the sequence number of the route request and propagates it through a shorter path so that the malicious route request is recorded at all the downstream intermediate nodes, including the destination $\mathcal{D}$. The adversary $\mathcal{X}$ then modifies the corresponding route reply by inserting the identity of the invalid nodes to the upstream fragment of the route reply to arrive at $\mathcal{S} \mapsto \mathcal{O} \mapsto \mathcal{X} \mapsto \mathcal{N} \mapsto \mathcal{P} \mapsto \mathcal{Q} \mapsto \mathcal{R} \mapsto \mathcal{C} \mapsto \mathcal{D}$. The data packets propagated through the corrupted route entraps $\mathcal{N}$ to communicate with a non-existent $\mathcal{P}$ and results in data flow disruption.


Step1: 'N' drops the data packets generated by 'S' since it struggles to locate the non-existent 'P' as directed in the route
Step2: 'N' generates a route error to report the non-existence of 'P' and forwards the route error to 'S'
Step3: 'X' modifies the identity of broken link from 'P' to 'V' before unicasting the route error to 'S'
S
DATA((D,C,R,Q,P,N,X,O,S)
RERR(N,S,(N,X,O,S),[V])
O
DATA((D,C,R,Q,P,N,X,O,S)
RERR(N,S,(N,X,O,S),[V])
X
DATA((D,C,R,Q,P,N,X,O,S)
RERR(N,S,(N,X,O,S),[P])
N
DATA((D,C,R,Q,P,N,X,O,S)
C
D

(a) Modification Attack – Disruption of Route Maintenance


Step2: 'D' discards the route and chooses the alternate shorter route for data flow
Step1: Adversary 'A' inserts the identity of invalid nodes into the route so that the route is not chosen for data flow
RREQ(S,D,6,(S,M,A))
RREQ(S,D,6,(S,M,A,B))
RREQ(S,D,6,(S,M))
A
B
Z
RREQ(S,D,6,(S,M,A,B,Z))
RREP(D,S,(D,C,E,Z,B,A,M,S)
RREP(D,S,(D,C,E,Z,B,A,M,S)
RREQ(S,D,6,(S))
M
RREP(D,S,(D,C,E,Z,B,A,M,S)
RREP(D,S,(D,C,E,Z,B,A,M,S)
E
RREQ(S,D,6,(S,M,A,B,Z,E))
RREP(D,S,(D,C,E,Z,B,A,M,S)
RREQ(S,D,6,(S,M,A,B,Z,E,C))
S
RREP(D,S,(D,C,E,Z,B,A,M,S)
C
RREP(D,S,(D,C,E,Z,B,A,M,S)
D
RREQ(S,D,6,(S,O,P,Q,R,V,U,W,X,N))
RREQ(S,D,6,(S))
RREQ(S,D,6,(S,O,P,Q,R,V,U,W,X,N,C))
N
O
RREQ(S,D,6,(S,O))
X
RREQ(S,D,6,(S,O,P,Q,R,V,U,W,X))

(b) Modification Attack – Gratuitous Detour Attack

Figure 5.2: Route Maintenance and Gratuitous Detour Attacks.

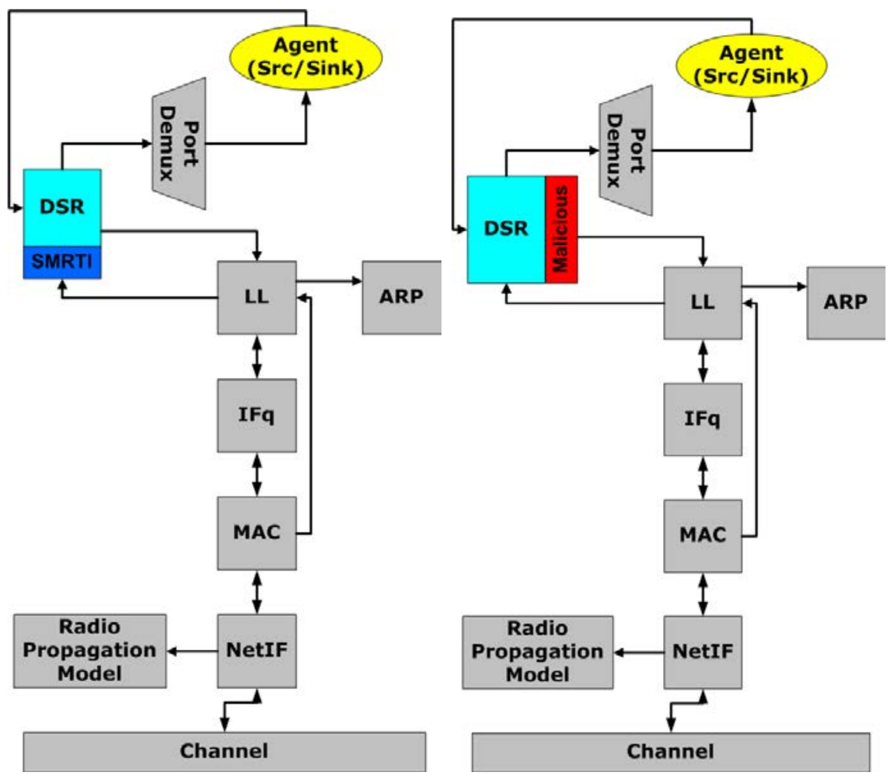
5.2.3 Route Maintenance Disruption

We have also modelled the adversary to disrupt the route maintenance by modifying the route error's header, because very few models explicitly cope with the route maintenance attacks described in Chapter 2. In the case of the data flow disruption attack (detailed in §5.2.2), the mobile node that struggles to deliver the data packets to a non-existent node generates a route error to report the incident to the source. Since the adversary is positioned between the source and the reporting node, it effectively modifies the identity of broken link that is reported in the route error. Therefore, the route error fails to remove the corrupted link from the route cache of the source and allows the adversary to maintain the disruption of the data flow. Also, the modified route error can coincide with a valid link that may, in turn, cause the removal of the valid link from the route cache of the source.

In Figure 5.2(a), the adversary $\mathcal{X}$ intercepts the route error and modifies the broken link of the route error from $\mathcal{P}$ to $\mathcal{V}$. This causes $\mathcal{S}$ to trim the routes that contain the link $\mathcal{N} \mapsto \mathcal{V}$, and effectively allows $\mathcal{S}$ to retain the corrupted route $\mathcal{S} \mapsto \mathcal{O} \mapsto \mathcal{X} \mapsto \mathcal{N} \mapsto \mathcal{P} \mapsto \mathcal{Q} \mapsto \mathcal{R} \mapsto \mathcal{C} \mapsto \mathcal{D}$ from the route cache. For this reason, the source succumbs to a data flow disruption attack by propagating data packets along the corrupted route.

5.2.4 Gratuitous Detour Attack

In contrast to the disruption attacks, the adversary can also perform a gratuitous detour attack to ensure that the route is not chosen as the active route for the data flow. In the case of the gratuitous detour attack, the identity of the invalid nodes is inserted into the route so that the lengthened route encourages the destination to prefer an alternative shorter route for the route reply. This effectively relieves the adversary from forwarding packets for other nodes. Recollect that the gratuitous detour attack is a variation of the detour attack discussed in Chapter 2 and is likely to be performed by selfish nodes to save their battery resources. However, the success rate of the gratuitous detour attack is likely to decrease, if, (a) *the alternate routes are longer*



(a) Malicious Mobile Node (b) SMRTI-enabled Mobile Node

FIGURE 5.3: Mobile Node – SMRTI and Adversary Extensions.

than the maliciously-lengthened route and, (b) *the destination replies to all received route requests to form multiple paths.*

In Figure 5.2(b), the adversary $\mathcal{X}$ lengthens the route contained in the route request by inserting the identity of the invalid nodes into the route. Assuming that the option for multiple paths is disabled at the DSR, the destination $\mathcal{D}$ would then discard the longer route $\mathcal{S} \mapsto \mathcal{O} \mapsto \mathcal{P} \mapsto \mathcal{Q} \mapsto \mathcal{R} \mapsto \mathcal{V} \mapsto \mathcal{U} \mapsto \mathcal{W} \mapsto \mathcal{X} \mapsto \mathcal{N} \mapsto \mathcal{C} \mapsto \mathcal{D}$ and prefer the shorter route $\mathcal{S} \mapsto \mathcal{M} \mapsto \mathcal{A} \mapsto \mathcal{B} \mapsto \mathcal{Z} \mapsto \mathcal{E} \mapsto \mathcal{C} \mapsto \mathcal{D}$ for the data flow. Therefore, $\mathcal{X}$ is effectively relieved from forwarding packets for other nodes.

5.3 Simulation Setup

We have used the NS2 to investigate the performance of the SMRTI and the DSR nodes against the malicious nodes. We have implemented the SMRTI and the adversary models as an extension to the NS2's pre-packaged DSR at the C++ layer. On other hand, the NS2's OTcl interpreter is used to define, configure and set-up the routing protocol, application-based traffic, network topology, simulation scenarios and the parameters for the DSR, SMRTI and adversary models. By inheriting the simulation setup from §4.4, a mobile node with the DSR as the routing protocol is known as the *DSR node*. The adversary model is implemented as an inbuilt module within the DSR and can be activated or deactivated from the OTcl interpreter using simulation scripts. A DSR node with the adversary model activated is known as a *malicious node* (Figure 5.3(a)). The SMRTI is implemented as a separate set of wrapper C++ classes and called from the DSR depending on whether it is activated from the OTcl interpreter's scripts; a SMRTI-enabled DSR node is known as a *SMRTI node* (Figure 5.3(b)). We have used the Rational Rose Enterprise for the design of the UML class diagrams and the code generation for the adversary and the SMRTI models. The class diagrams are available at *Appendix A* and the code snippets at *Appendix B*. *Appendix C* contains the simulation and trace analysis scripts that have been employed for the simulations (§5.4). In the following, we present the general simulation parameters, options that have been enabled and disabled for the DSR, parameters that have been finalised for the SMRTI and, lastly, the simulation scenarios. The simulation parameters for the DSR, SMRTI and adversary are summarised in Table 5.1. The NS2's simulation parameters follow from Table 3.3; the parameters being presented for the NS2 in Table 5.1 are those that are amended for the purpose of evaluating the performance of the SMRTI nodes. Otherwise, the background materials presented for the NS2 in §4.4 also apply in this context.

TABLE 5.1: Simulation Parameters for NS2, DSR, SMRTI, and Adversary

NS2 Parameters	
Minimum Duration for a CBR Connection	20s
Maximum Duration for a CBR Connection	40s
DSR Parameters	
Promiscuous Monitoring	True
Bidirectional Links	True
Snoop Route Errors	False
Snoop Routes	False
Salvage Broken Links	False
Salvage Bad Route Replies	False
Reply from Route Cache	False
Propagate Route Errors with Route Request	False
Zero-ring Search for Route Requests	False
Reply only to First Seen Route Request	False
Flow State	False
SMRTI Parameters	
Trust Metric	$[-1, +1]$
Reputation Metrics	$[-1, +1]$
Trustworthy Nodes	$(0.75, +1]$
Selectively Well-behaving Nodes	$(0.50, 0.75]$
Selectively Misbehaving Nodes	$(0.25, 0.50]$
Malicious Nodes	$(0, 0.25]$
Persistent Malicious Nodes	$[-1, 0)$
Threshold-limit (Δ)	0.50
Initial Reputations (R_0)	0.51
Segment Interval (θ)	0.25
Neg(event, action)	0.10
Pos(event)	0.02
Reputation Decay/Growth Factor (δ)	0.01
Constraints	$\{R_0 \geqslant \Delta\}; \{Neg > Pos > \delta\}$
Malicious Parameters	
Probability of Malicious Action	100%
Distribution of Malicious Action	Random

5.3.1 General Simulation Parameters

The simulation parameters that are specified in this section apply to all types of mobile nodes (DSR, SMRTI and malicious nodes) until otherwise specified. Since the MANET lacks a standard benchmark for defining the simulation parameters, we have inherited the simulation parameters that were predominantly adopted in the related security models [159, 164, 302].

Recollect from Chapter 3 that the *random way-point model* is adopted as the mobility model for the mobile nodes, where the velocity was uniformly chosen between 0 and the maximum velocity, V_{max}. Similarly, the CBR flows are chosen to study the performance of the SMRTI in the presence of the connectionless communication. These flows are dynamically varied between randomly-chosen set of senders and receivers, because we observed such a design to discovers the multiple routes and exposes the DSR and the SMRTI nodes to the multiple malicious nodes. We allocated 300 s for the simulation duration, and 20 s and 40 s as the minimum and maximum durations for the CBR flows, respectively. This realistically permitted a few terminating and commencing flows to overlap rather than to follow a batch execution. Analogous to the §4.4, the V_{max} and pause times are varied between 0 m/s to 50 m/s and 0 s to 40 s, respectively, depending on the simulation scenario. The simulation area is varied from $500 * 500m^2$ to $5000 * 5000m^2$. Nevertheless, the total number of nodes for all simulation scenarios has been fixed at 100.

5.3.2 Extensions and Options in DSR

The NS2 incorporates a few extensions for the DSR that have been incrementally-proposed in recent years. In the following, we briefly list those extensions and we then detail those extensions that have been considered for our simulations. The NS2's version of the DSR ignores the unidirectional routes, so that the mobile nodes can respond back with acknowledgements to the received data packets, as noted in IEEE 802.11. Henceforth, the implementation does not provide an option for the destination to invoke a route request to the source. The NS2 also incorporates a *flow-state* extension

[156], where an identity is established for a flow to reduce the source routing overhead. The other extensions include whether:

- Intermediate nodes can, (a) *snoop route errors forwarded by other nodes*, (b) *snoop routes forwarded by other nodes*, (c) *send gratuitous route replies*, (d) *salvage a broken link using an existing route from route cache*, (e) *consult route cache before propagating a route request* and, (f) *salvage bad route replies.*

- Source nodes can, (a) *propagate the last-seen route error together with the next route request* and, (b) *broadcast a non-propagating route request as the first step during each route discovery.*

- Destination nodes can reply only to the first-seen route request.

- Source, intermediate and destination nodes can promiscuously monitor the wireless medium.

Given that only bidirectional links are considered, we have enabled the destination DSR nodes to respond to all route requests rather than to only the first-seen route request. We have also enabled the DSR nodes to assist the SMRTI model to promiscuously monitor the transmissions of the other nodes. We have disabled the remaining extensions because we are only interested in studying the performance of the basic DSR protocol and the role of the SMRTI in enhancing the routing decisions of the DSR.

5.3.3 Simulation Parameters for SMRTI

In our simulations, the *neg(event, action)* that corresponds to a negative value awarded for a malicious action is fixed at a constant value for simplicity. The parameter, *Pos(event)*, that attributes the positive value awarded for a benign behaviour is chosen as a multiple of the *neg(event, action)* to make misbehaving unattractive. We have chosen a uniform value 0.50 as the threshold-limit (Δ) for all event-specific contexts. The initial reputation values (direct, observed and recommended) for the SMRTI nodes are optimistically fixed at 0.51. This enables the SMRTI nodes to effectively detect the malicious nodes that deviate from a normal behaviour at the point of deployment.

In terms of segmentation, the segment S_3 $\{i.e.\ 0.50 \geqslant S_3 > 0.25\}$ constitutes *selectively well-behaving nodes*, while the segment S_2 $\{i.e.\ 0.75 \geqslant S_2 > 0.50\}$ constitutes *selectively misbehaving benign nodes*. The segment S_1 $\{i.e.\ +1 \geqslant S_1 > 0.75\}$ attributes the list of nodes that are considered trustworthy by a SMRTI node. Finally, the segments S_i $\{0.25 \geqslant S_4 > 0,\ 0 > S_5 \geqslant 0.25, \cdots, 0.75 > S_8 \geqslant -1\}$ attributes malicious nodes, where the ascending order of segments corresponds to the ascending order of maliciousness.

5.3.4 Simulation Scenarios

The following two compositions, (a) the *SMRTI and malicious nodes* and, (b) the *DSR and malicious nodes*, are simulated with identical parameters for each of the simulation scenarios mentioned below. The total number of nodes for each of the above compositions is fixed at 100, although the proposition between the malicious and the SMRTI (or DSR) nodes may vary depending on the simulation scenario. All these have enabled us to compare the performance of the SMRTI and the DSR nodes. The performance comparison between the SMRTI and the DSR compositions for every scenario is derived from an average of 20 executions. The simulation scenarios considered for the performance analysis are given below:

Scenario I In this scenario, we evaluated the performance of the SMRTI and the DSR nodes against the increasing proportions of malicious nodes from 0 to 100 in increments of 10 with a pause time of 10 s, V_{max} of 20 m/s and simulation area of $1500 * 1500m^2$. Note that the proportion of the SMRTI or the DSR nodes decreases with the increasing proportion of malicious nodes to maintain the total count of nodes at 100. The objective of this scenario is to discover the proportion of malicious nodes after which the performance of the SMRTI and the DSR nodes fall noticeably.

Scenario II This scenario presents the impact of mobility on the performance of the SMRTI (or DSR) nodes in the presence of malicious nodes. We have maintained the proportion of the malicious nodes to the SMRTI (or DSR) nodes to three uniformly-distributed values, (a) *25%*, (b) *50%* and, (c) *75%* of the total nodes in the network. For each distribution, we then analysed the performance of the SMRTI (or DSR) nodes by varying V_{max} from 0 m/s to 50 m/s in increments of 5 m/s with pause time of 10 s, and simulation area of $1500 * 1500m^2$.

Scenario III Similar to the previous scenario, we retained the proportion of the malicious nodes to the SMRTI (or DSR) nodes to three uniformly-distributed values, (a) *25%*, (b) *50%* and, (c) *75%* of the total nodes in the network. For each of the distributions, we then analysed the performance of the SMRTI (or DSR) by varying the pause time from 0 s to 40 s in increments of 4 s. This scenario produces insight into the influence of the pause time on the performance of the mobile SMRTI (or DSR) nodes.

Scenario IV In contrast to Scenario III, we varied the simulation area from $500 * 500m^2$ to $5000 * 5000m^2$ in increments of $500 * 500m^2$ but retained the values of the remaining parameters. This scenario evaluates the impact of the network connectivity and node density on the performance of the SMRTI (or DSR) nodes in the presence of the malicious nodes.

5.3.5 Performance Metrics

The performance metrics evaluated for each scenario are presented below:

- **Packet Delivery Ratio (PDR)** is the average ratio of total number of the CBR packets received by the destination to the total number of the CBR packets sent by the source.
- **Successful Route Discovery (SRD)** is the average route discovery cycles initiated by the source for establishing a CBR flow with the destination.
- **Latency** is the average time taken by a CBR packet to travel from the source to the destination.

However, the performance metrics, such as the packet or byte overhead are not evaluated in our simulations because the SMRTI nodes do not generate additional packets or headers to communicate recommendations as in the related models. As demonstrated below, the SMRTI nodes inherit the routing specifications of the DSR without any alteration.

5.4 Simulation Results

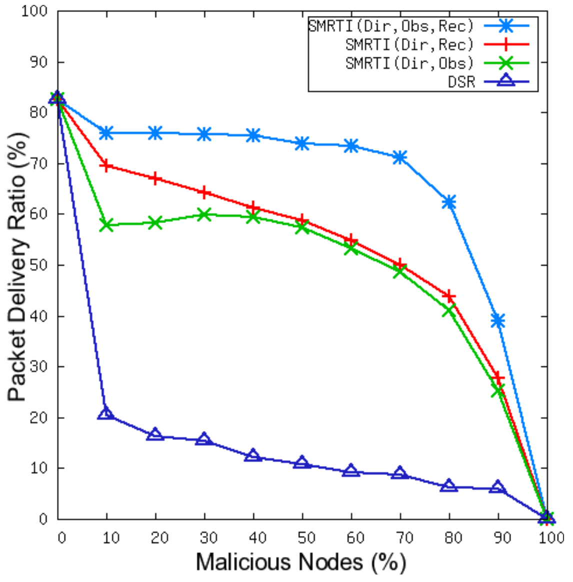
In this section, we discuss the performance metrics obtained for each scenario and then analyse the performance of the SMRTI nodes in comparison with the DSR nodes, based on those metrics.

5.4.1 Scenario I

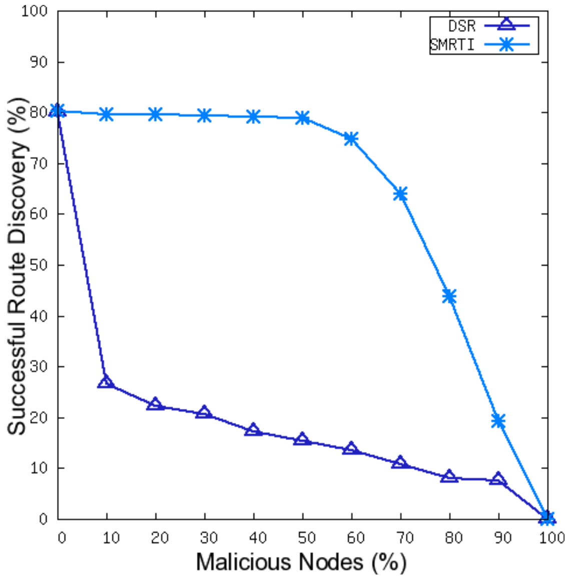
As detailed earlier, this scenario explores the impact of the increasing proportion of malicious nodes on the performance of the SMRTI and the DSR nodes.

5.4.1.1 PDR

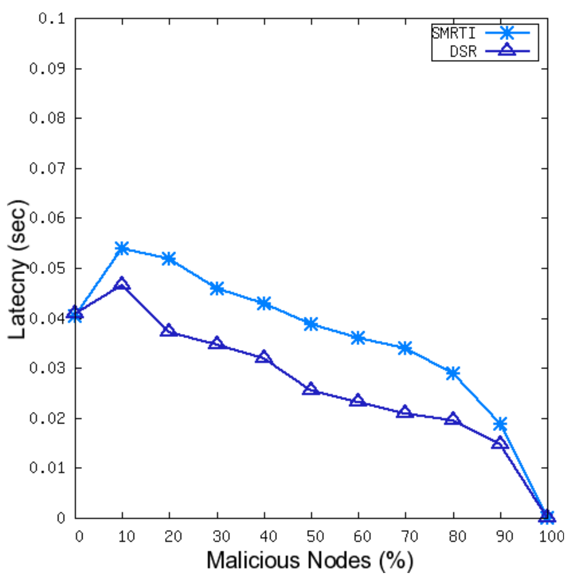
As seen in Figure 5.4(a), the SMRTI nodes outperform the DSR nodes. For example, the PDR of the DSR nodes decreases from 82% to 20% on introducing 10% malicious nodes. The drop in the PDR primarily results from the route discovery disruption attack caused by the malicious nodes. As discussed in §5.2.1, the malicious nodes modify the route request's sequence number and the route reply's route to disrupt the route discovery cycles in the DSR nodes. In such a situation, the maliciously incremented sequence number prevents the propagation of a route request that not only contains the valid sequence number, but also belongs to the same route discovery cycle. This is because the DSR nodes consider a route request with sequence numbers less than or equal to the maliciously-incremented sequence number as a stale route request. This situation persists until the sequence number of a future route request exceeds the modified sequence number. Furthermore, the modification of the routes to disrupt the CBR flows (§5.2.2) also reduces the PDR of the DSR nodes. In such a



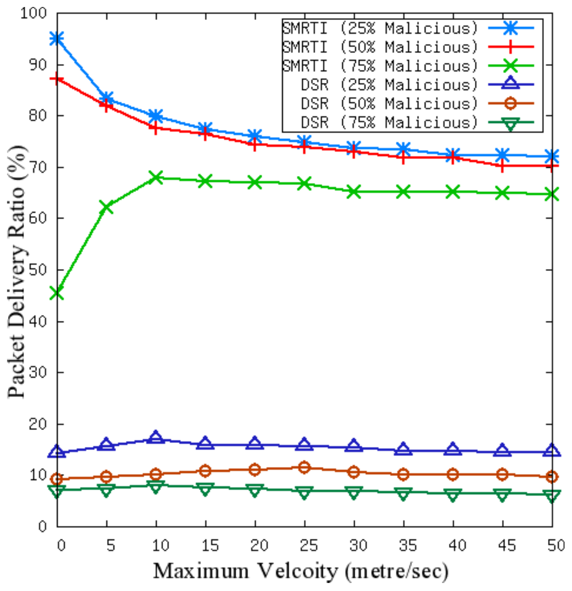
(a) Scenario I – PDR Vs Malicious Nodes



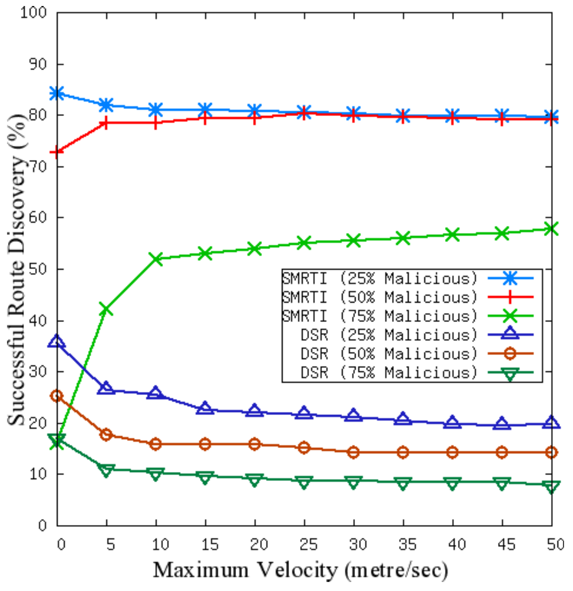
(b) Scenario I – SRD Vs Malicious Nodes



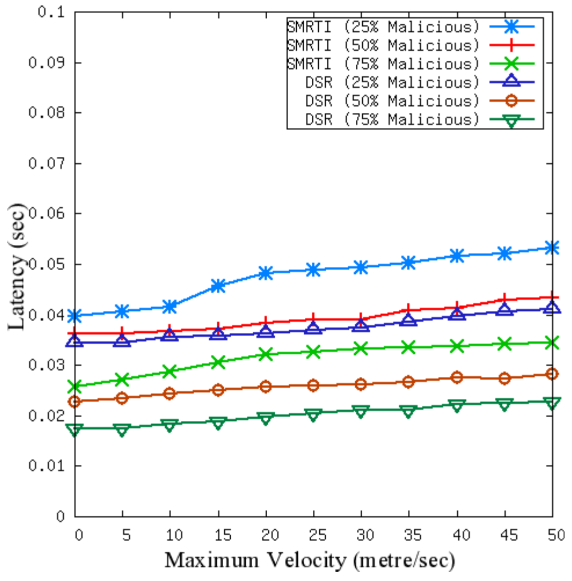
(c) Scenario I – Latency Vs Malicious Nodes



(d) Scenario II – PDR Vs V_{max}



(e) Scenario II – SRD Vs V_{max}



(f) Scenario II – Latency Vs V_{max}

FIGURE 5.4: Performance of SMRTI Nodes against Adversaries and V_{max}

situation, the malicious nodes add unreachable nodes to a route (or delete nodes from a route) so that the broken links are introduced to the route. Finally, the increase in the proportion of the malicious nodes further disrupts the PDR of the DSR nodes, because the only routes that can be established between the DSR nodes are restricted to one or two-hops in length.

In Figure 5.4(a), the curve *'SMRTI (Dir, Obs)'* represents the PDR of the SMRTI nodes based on the evidence captured for the direct and observed reputations. As detailed in §5.2, the SMRTI nodes capture the evidence for a next-hop's misbehaviour (such as the modification of either a route request's sequence number or route reply's route) in their direct reputation and, accordingly, append the malicious next-hop to their reject-list. Therefore, the SMRTI nodes are able to effectively discard the route reply received from malicious next-hops and hence defend against route discovery and data flow disruption attacks. They also capture evidence for the benign behaviour exhibited by the next-hops during the route discovery cycles and the data flows. For example, the SMRTI nodes capture evidence for benign behaviours when the next-hops forward packets without performing any malicious modifications. Furthermore, the SMRTI nodes augment the direct reputation held for the next-hops by using the observed reputation captured for those next-hops in which they observe the misbehaviours of those next-hops towards their common neighbours. For instance, the SMRTI nodes capture evidence when a malicious neighbour modifies a route request received from a common neighbour. In such an instance, the observing SMRTI nodes ignore the route request broadcasted by the malicious neighbour. Thus, the direct reputation in combination with the observed reputation enables the SMRTI nodes to exclude the routes that contain malicious neighbours and also prevents them from receiving and sending packets to malicious neighbours. The PDR reduces notably when the proportion of the malicious nodes exceeds 50%. This results from the higher availability of the malicious neighbours.

The curve *'SMRTI (Dir, Rec)'* in Figure 5.4(a) displays the PDR of the SMRTI nodes based on the evidence captured for the direct and recommended reputations. In comparison with the observed reputation, the recommended reputation allows the

SMRTI nodes to better identify other multi-hop SMRTI nodes by deriving recommendations for those nodes based on the incoming trustworthy packets. Thus, the recommended reputation in combination with the direct reputation enables the SMRTI nodes to establish trustworthy routes free from malicious nodes. When the proportion of the malicious nodes exceeds 50%, the PDR reduces notably for the reason mentioned earlier. The combined effect of the observed and recommended reputations can be seen in the curve *'SMRTI (Dir, Obs, Rec)'* (Figure 5.4(a)), where the combined effect renders a superior PDR for up to 70% malicious nodes. From here onwards, we consider this combined effect of all the reputations (direct, observed and recommended) to study the performance of the SMRTI nodes.

In Figure 5.4(a), the DSR and the SMRTI nodes exhibit a similar PDR in the absence of malicious nodes because the latter eliminates further overheads, such as the dissemination of additional packets for the communicating recommendations. We also noticed that the PDR of the SMRTI and DSR nodes never reaches 100%, even in the absence of malicious nodes, for the following reasons. First, few destinations are unreachable because the random way-point mobility disrupts the uniform distribution of the nodes. Second, the nodes tend to choose shorter paths that generally lead to a high congestion. Finally, the network-centric nodes undergo a heavy contention because most of the traffic flows through the centre of the network to connect to the boundary nodes. Solutions based on load balancing and QoS [158, 232, 363, 392] can be employed to improve the performance in such situation; however, they are beyond the scope of our thesis.

5.4.1.2 SRD

Similar to the PDR curve observed for the DSR nodes in Figure 5.4(a), the SRD curve of the DSR nodes falls steeply with the introduction of the 10% malicious nodes in Figure 5.4(b). Malicious nodes collapse the SRD of the DSR nodes by injecting incremented sequence numbers and modified routes. Note that the probability of encountering a malicious node increases as the proportion of the malicious nodes also increases. In addition, the proportion of the DSR nodes is also reduced with the increase in proportion of the malicious nodes, so that the total count of nodes is maintained at 100. The reduction further adds to the disruption of the route discovery. Although similar characteristics are observed for the SRD and PRD curves of the DSR nodes, these curves fail to match their values for the following reason. Since the SRD attributes the reception of a route reply as a successful route discovery, it is prone to include modified routes other than the valid routes. Note that the malicious nodes insert those modified routes to disrupt the CBR flows or to launch blackhole attacks. Alternatively, the PDR attributes the successful reception of the CBR packets at the destination DSR for which the discovered routes must be valid and active, and hence the mismatch between the values of the PDR and SRD curves.

In Figure 5.4(b), the SMRTI nodes exhibit a high SRD owing to their ability to defend against malicious nodes that disrupt the route discovery cycles and CBR flows. Although the SRD imitates the characteristics of the PRD in Figure 5.4(a), it exhibits a different magnitude for the following reason, the *SMRTI nodes invalidate a route discovery if a malicious node is encountered in the route.* While the proportion of malicious nodes increases, the proportion of the SMRTI nodes falls and therefore the low proportion of the SMRTI nodes struggles to discover a valid route around the malicious nodes.

5.4.1.3 Latency

The latency curves shown in Figure 5.4(c) correspond to the PDR curves (*SMRTI (Dir, Obs, Rec)* and the *DSR*) displayed in Figure 5.4(a). Since the PDR of the DSR nodes results primarily from the active routes that are one or two-hops in length, the CBR packets take less time to travel through the DSR nodes and thus the lower latency. On the other hand, the CBR flows of the SMRTI nodes attribute to a higher latency for the following reasons, (a) the *SMRTI nodes require a longer duration to discover trustworthy routes as the proportion of malicious nodes increases* and, (b) *the trustworthy routes tend to be longer in hop length, because the SMRTI nodes route mostly around the malicious nodes.* This can be confirmed in Figure 5.4(c), where both the DSR and the SMRTI nodes have the same latency value in the absence of malicious nodes. Also there may be negligible time taken for making trust decisions at every SMRTI node; howeverm this has not been considered in our simulations because the introduced time is insignificant owing to low computational and storage overheads.

5.4.2 Scenario II

Here, we study the impact of mobility on the performance of the SMRTI and DSR nodes against the fixed proportions of the malicious nodes.

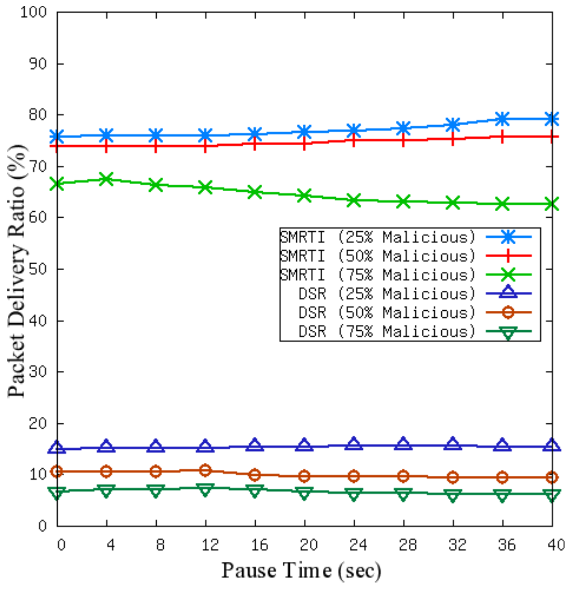
5.4.2.1 PDR

The DSR nodes perform poorer than the SMRTI nodes against the fixed proportions of the malicious nodes (25%, 50%, and 75%) with varying V_{max} in Figure 5.4(d). The PDR of the DSR nodes decreases with the increasing proportions of the malicious nodes from 25% to 75%, which can be confirmed from the curves, (a) *'DSR (25% Malicious)'*, (b) *'DSR (50% Malicious)'*, and (c) *'DSR (75% Malicious)'*. The DSR nodes exhibit similar characteristics for other performance metrics (SRD and latency), which we will explain in detail in the following sections. The increments to V_{max} in the intervals of 5 m/s from 0 m/s to 50 m/s exhibit an insignificant influence on the slope of each PDR curve. The reason relies on the fact that the DSR nodes accumulate the PDR mostly from one or two-hop based CBR flows that are unlikely to be influenced by the varying mobility.

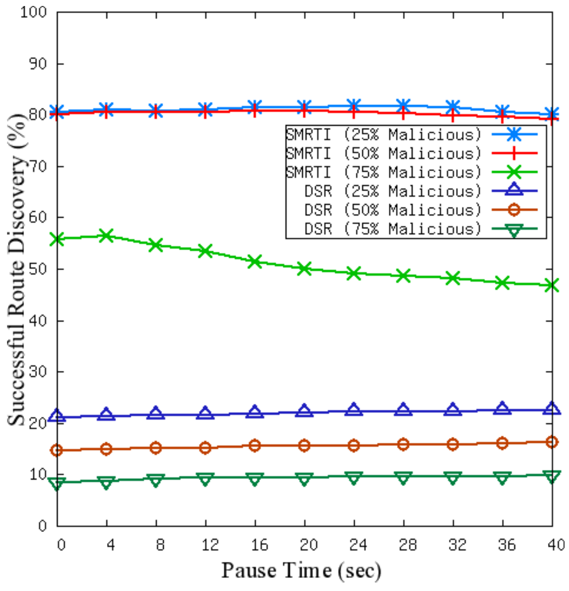
As observed from the PDR curves in Figure 5.4(d), (a) *'SMRTI (25% Malicious)'*, (b) *'SMRTI (50% Malicious)'* and, (c) *'SMRTI (75% Malicious)'*, the SMRTI nodes perform better against the 25% than the 50% and 75% malicious nodes. Similar characteristics are inherited for other performance metrics, such as the SRD and latency, which will be discussed below. When V_{max} is incremented in 5 m/s from 0 m/s to 50 m/s, the PDR curves of the SMRTI nodes exhibit a significant change in their slopes that is different to the PDR curves observed for the DSR nodes. Especially, the change in slope is evident for the PDR curves when the SMRTI nodes are less mobile between 0 m/s to 10 m/s.

In Figure 5.4(d), the SMRTI nodes exhibit an excellent PDR when the proportion of the malicious nodes is not greater than 50% and the V_{max} is fixed at 0m/s. First, the immobility (at 0m/s) allows the SMRTI nodes to a capture recurring identical set of evidence for the malicious and benign behaviours of the neighbours. Second, the SMRTI nodes sustain a high PDR at 0m/s because the established routes are not disturbed by mobility. Although the evidence captured for the recommended reputation may vary because of the dynamic CBR flows, the recurring evidence captured for the direct and observed reputations influences the SMRTI nodes to deepen the separation between the malicious and benign neighbours. Therefore, the SMRTI nodes accurately evaluate the trustworthiness of the neighbours and, henceforth, make precise decisions. This enables the SMRTI nodes to forward the route discovery and the CBR packets only to the trustworthy neighbours. As the mobility increases, new neighbours are introduced to the environment of the SMRTI nodes and lead to the availability of new evidence. Further increases in V_{max} in increments of 5 m/s from 10 m/s to 50 m/s lead to broken links that results in the loss of data packets and consequently induce new route discovery cycles. All these collectively contribute to the reduced PDR.

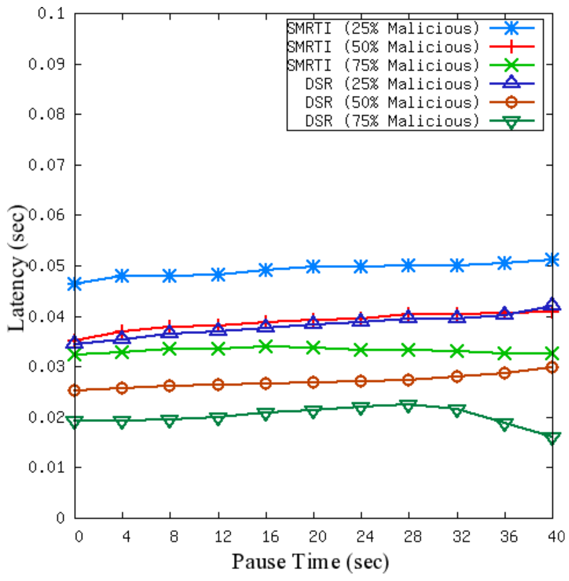
In contrast to the above cases, the SMRTI nodes perform poorly against 75% of malicious nodes at 0 m/s in Figure 5.4(d). Recall that immobility fixes the position of the nodes, thereby causing the SMRTI nodes to capture the recurring evidence for the behaviours (benign and malicious) of the neighbours in their direct and observed reputations. Although the recurring evidence effectively deepens the separation between the malicious and benign neighbours, the presence of three-fourths of the malicious nodes (on average) in the environment means that the SMRTI nodes struggle to find trustworthy neighbours for forwarding the route discovery and the CBR packets. Hence, the SMRTI nodes exhibit a poor PDR, regardless of their capacity to effectively distinguish malicious neighbours from benign neighbours. Ironically, the PDR for the SMRTI nodes improves with mobility because the mobility increases the probability of the SMRTI nodes to meet other SMRTI nodes. Further increments to V_{max} introduce broken links to the CBR flows, which prevents additional improvements to the PDR.



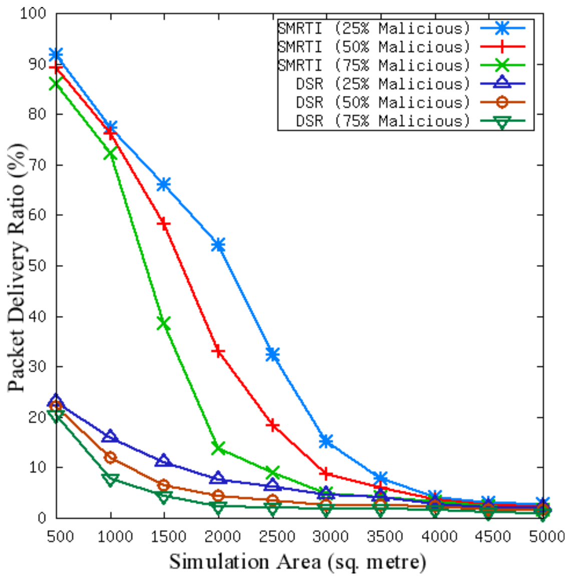
(a) Scenario III – PDR Vs Pause Time



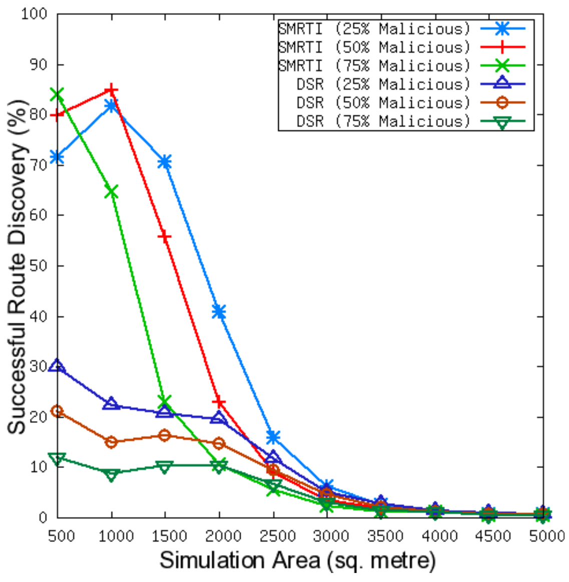
(b) Scenario III – SRD Vs Pause Time



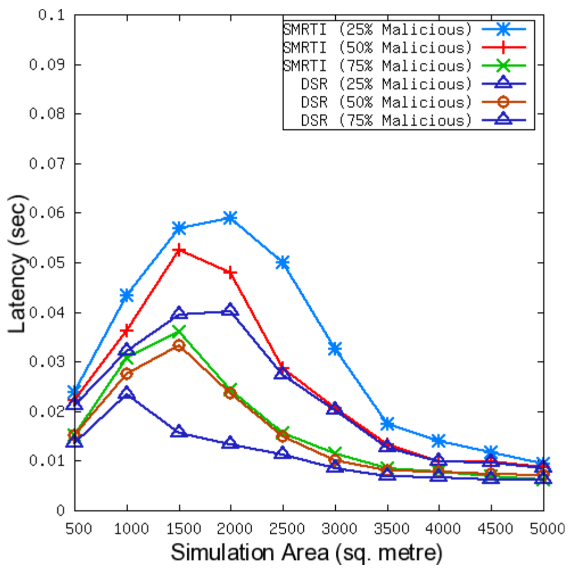
(c) Scenario III – Latency Vs Pause Time



(d) Scenario IV – PDR Vs Simulation Area



(e) Scenario IV – SRD Vs Simulation Area



(f) Scenario IV – Latency Vs Simulation Area

FIGURE 5.5: Performance of SMRTI Nodes against Pause Time and Node Density

5.4.2.2 SRD

Figure 5.4(e) presents the SRD for both the SMRTI and the DSR nodes. The capacity of the DSR nodes to gradually discover routes falls as the V_{max} increments to 20 m/s. However, beyond 20 m/s, the SRD of the DSR nodes remain uninfluenced by the increasing V_{max}. The DSR nodes exhibit a low SRD (*i.e.* low success rate in discovering routes) because of their inability to distinguish the valid route requests from those with maliciously-incremented sequence numbers. Nevertheless, the DSR nodes exhibit a peak SRD at 0 m/s owing to the immobile static network topology. Interestingly, the peak value observed for each of the SRD curves fails to map to the corresponding PDR curves observed for the DSR nodes in Figure 5.4(d). As detailed in §5.4.1.2, the DSR nodes consider a successful route discovery as any route discovery that establishes a route between the source and target DSR nodes. Hence, both modified and valid routes constitute towards the SRD. Since the PDR is derived from only valid routes, there is an inconsistency between the SRD and PDR curves.

As shown in Figure 5.4(e), the SMRTI nodes are better than the DSR nodes at discovering valid routes because of their ability to reject route requests that contain maliciously-incremented sequence numbers, and to discard the route replies that contain the modified routes. The SRD curves of the SMRTI nodes reach either a peak or a floor value at 0 m/s depending on the proportion of malicious nodes present in the network. The SRD curve *'SMRTI (25% Malicious)'* exhibits a peak value at 0 m/s, which, in turn, demonstrates the capability of the SMRTI nodes to establish valid routes among the less-populated malicious nodes. Alternatively, the SRD curves, (a) the *'SMRTI (50% Malicious)'* and, (b) the *'SMRTI (75% Malicious)'* correspond to the floor values at 0 m/s. This illustrates the difficulty that the SMRTI nodes have to undergo to establish valid routes among the densely populated malicious nodes. Another interesting finding in the graph is the influence of the increasing mobility in assisting the SMRTI nodes to discover valid routes amidst the densely populated malicious nodes.

5.4.2.3 Latency

In Figure 5.4(f), the latency for both the DSR and SMRTI nodes uniformly ascends with increasing V_{max}. Mobility is known not only to induce broken links, but also to trigger source nodes to initiate route discoveries to fix those broken links. The SMRTI nodes exhibit an increased latency towards the higher end of V_{max} in the curve *'SMRTI (25% Malicious)'* because they successfully discover fresh routes even if the increasing mobility introduces broken links. However, the high proportion of malicious nodes restricts the SMRTI nodes from discovering fresh routes, which can be seen from the curves, (a) *'SMRTI (50% Malicious)'* and (b) *'SMRTI (75% Malicious)'*. Therefore, the CBR packets that are buffered within the SMRTI nodes as a result of broken links are restricted from constituting towards the PDR and hence towards the latency.

5.4.3 Scenario III

In this scenario, we study the impact of pause time on the performance of SMRTI and DSR nodes against the fixed proportions of malicious nodes.

5.4.3.1 PDR

The PDR of the DSR nodes is no better than the PDR of the SMRTI nodes in Figure 5.5(a). As expected, the PDR of the DSR nodes decreases with the increasing proportion of malicious nodes that can be seen from the following curves, (a) *'DSR (25% Malicious)'*, (b) *'DSR (50% Malicious)'*, and (c) *'DSR (75% Malicious)'*. The PDR of the DSR nodes exhibits a marginal variation to the pause time that is incremented in intervals of 4 s from 0 s to 40 s. First, the lack of a rest in between the successive mobility at the lower end of pause time and the probability of enlisting a malicious node to the route either introduces broken links or restricts the CBR flows to the routes that are one or two-hops in length. Alternatively, a prolonged rest (that nearly equals the duration of the CBR flow) between successive mobility exposes the DSR nodes to a fixed set of malicious nodes and, once again, forces the DSR nodes to derive the PDR from the one or two-hop based routes.

As seen in Figure 5.5(a), the SMRTI nodes demonstrate a high PDR against malicious nodes in the presence of a varying pause time. The PDR of the SMRTI nodes improves as the pause time increases in increments of 4 s from 0 s to 40 s, provided the proportion of the malicious nodes is not greater than 50% of the total nodes in the network. This can be observed from the curves, (a) *'SMRTI (25% Malicious)'*, and (b) *'SMRTI (50% Malicious)'*. The basis for rise in the PDR results from the increase in pause time (*i.e.* as the duration for which the SMRTI nodes are kept at rest between successive mobility increases) locks the SMRTI nodes to the same set of neighbours and hence yields the recurring identical sets of evidence. In particular, the PDR of the SMRTI nodes is higher when the pause time is 40 s because the pause time equates to the maximum duration assigned for a CBR flow. However, the SMRTI nodes exhibit different characteristics when the proportion of the malicious nodes is greater than 50% of the total nodes in the network. This can be observed from the curve *'SMRTI (75% Malicious)'* in Figure 5.5(a). The PDR of the SMRTI nodes gradually decreases with the increasing pause time, because the SMRTI nodes are locked among the malicious nodes until the completion of the CBR flows, that is, the SMRTI nodes struggle to establish valid routes regardless of precisely identifying the malicious neighbours. In Figures 5.4(d) and 5.5(a), we also discovered that a high V_{max} and low pause time, and a low V_{max} and high pause time produce similar impacts on the SMRTI nodes.

5.4.3.2 SRD and Latency

The DSR nodes exhibit a poor SRD in comparison to the SRD of the SMRTI nodes in Figure 5.5(b). The SRD of the DSR nodes decreases with the increasing proportion of malicious nodes, which can be observed from the following curves, (a) *'DSR (25% Malicious)'*, (b) *'DSR (50% Malicious)'*, and, (c) *'DSR (75% Malicious)'*. However, the SRD of the DSR nodes demonstrates a marginal increase when the pause time is incremented in intervals of 4 s from 0 s to 40 s. The marginal increase results from the stable one or two-hop routes that are supported by the prolonged rest introduced between the successive mobilities.

The SMRTI nodes display a high SRD in Figure 5.5(b) owing to their ability to enlist only trustworthy nodes for the data flow. The SRD of the SMRTI nodes remain almost unaffected, whilst varying the pause time especially when the proportion of malicious nodes is fixed to 25% and 50%, which can be seen from the following curves, (a) *'SMRTI (25% Malicious)'* and, (b) *'SMRTI (50% Malicious)'*. However, the SMRTI nodes are subject to the influence of the pause time when the proportion of the malicious nodes is increased to 75%, and can be seen from the curve *'SMRTI (75% Malicious)'*. Recall that, on average, three-fourths of the malicious nodes are available in the environment of the SMRTI nodes and the prolonged rest between the successive mobilities lock the SMRTI nodes to the environment of the malicious nodes.

As seen in Figure 5.5(c), the DSR nodes demonstrate a lower latency compared to the SMRTI nodes because of the use of the one or two-hop routes for the CBR flows. The latency curves of the DSR nodes (*DSR (25% Malicious), DSR (50% Malicious),* and *DSR (75% Malicious)*) and the SMRTI nodes (*SMRTI (25% Malicious), SMRTI (50% Malicious),* and *SMRTI (75% Malicious)*) demonstrate similar characteristics against all proportions of the malicious nodes (25%, 50%, and 75%) in the presence of a varying pause time. A high latency is observed for both the DSR and SMRTI nodes at the top end of the pause time as long as the greatest proportion of the malicious nodes is 50%. The prolonged rest between the successive mobility attributes to stable links that, in turn, constitute towards a high PDR and therefore, a high latency. Although it is reasonable to expect a similar characteristic for the latency curves of the DSR and SMRTI nodes when the proportion of the malicious nodes is set to 75%, an opposite characteristic is unexpectedly observed because of the high proportion of malicious nodes in the environment of the DSR and SMRTI nodes.

5.4.4 Scenario IV

This scenario demonstrates the impact of the simulation area and hence, the node density on the performance of the SMRTI and DSR nodes against the fixed proportions of malicious nodes.

5.4.4.1 PDR

In Figure 5.5(d), both the DSR and SMRTI nodes exhibit a peak PDR at the summit of the node density. A high node density results nodes covering more than half of the simulation area ($500 * 500m^2$) with their 250 m transmission range and therefore establish one-hop CBR flows. The SMRTI nodes achieve a high PDR owing to their ability to communicate only with trustworthy neighbours and to exclude the malicious neighbours, while the DSR nodes display a low PDR caused by their vulnerability to disruption attacks. Increasing the simulation area to $1500 * 1500m^2$ reduces the PDR of both the DSR and SMRTI nodes owing to the increased initiation of the route discoveries because of the broken links. The SMRTI nodes exhibit a relatively poorer PDR as the simulation area increases, because of the reduced density of the trustworthy nodes in their environment. Further increases in simulation area disperse the network into clusters, such that the performance is realised only from intra-cluster communications. The DSR and SMRTI nodes exhibit an identical performance when the network becomes sparsely connected from $3500*3500m^2$ onwards. We observe from the PDR curves that expanding the simulation area disconnects the entire network beyond $4500 * 4500m^2$.

5.4.4.2 SRD and Latency

Figure 5.5(e) demostrates on how the ability of the DSR and SMRTI nodes to discover valid routes reduces with the increasing simulation area. The SMRTI nodes demonstrate a higher capability for discovering valid routes at $500 * 500m^2$, even if the network is composed of a high proportion of malicious nodes. When the simulation area is increased to $1000 * 1000m^2$, the capability of the SMRTI nodes surprisingly declines with the decreasing proportion of malicious nodes (75%, 50%, and 25%). Although the SMRTI nodes can effectively shortlist malicious neighbours in the case of the 25% malicious nodes, the route discovery attempts to learn the existence of the multi-hop malicious neighbours among the densely populated mobile nodes, leading to a low SRD. Even if the SMRTI nodes holds a similar approach against the 75% malicious nodes, the low proportion of trustworthy neighbours and the low rate of route discovery attempts renders a relatively high SRD. Recall that the SRD attributes the ratio of the successful route discoveries to the initiated route discoveries and, therefore, the high SRD value from the few initiated route discoveries produced correspondingly successful routes against the 75% malicious nodes. However, as seen in Figure 5.5(e), the PDR of the SMRTI nodes against the 75% malicious nodes is lower than the PDR obtained against the 50% and 25% malicious nodes, respectively. As the simulation area increases, the SMRTI nodes exhibit the expected characteristics by displaying a high SRD against the low proportion of malicious nodes. A further expansion to the simulation area not only disconnects the network, but also restricts the SMRTI nodes from exhibiting the same SRD as the DSR nodes.

Figure 5.5(f) shows the higher latency for SMRTI for the reasons observed in §5.4.1.3, such as, (a) *a longer duration required for finding trustworthy routes as the proportion of the malicious nodes increases* and, (b) *the trustworthy routes tend to be longer when the SMRTI nodes must route around the malicious nodes.* The latency curves for the DSR and SMRTI nodes gradually increase to a peak value as the node density decreases and then the latency curves display a gradual decrease beyond the peak value. A lower latency is observed at both the ends of the simulation area ($500 * 500m^2$ and $5000 * 5000m^2$) because most routes are one-hop in length. At the lower end of the simulation area ($500 * 500m^2$), the high node density yields one-hop communications. Alternatively, at the higher end of simulation area ($5000 * 5000m^2$), the intra-cluster communications yields one-hop communications. The peak value at the mid-range simulation areas corresponds to the PDR obtained from the multi-hop routes.

In summary, this scenario not only portrays the impact of node density in determining the performance of the SMRTI nodes, but also presents the insight that the scope of an attack is relevant as long as the network is operational. This confirms that the defence mechanism designed against the attackers fail to render anything significant when the network is inoperative.

5.4.5 Analysis of Selective Behaviours

In this section, we first analyse the role of the reputation decay/growth factor δ in shaping the reputation of a node when there is lack of evidence for the node's behaviour. We then discuss the significance of categorising selectively-behaving nodes into separate segments. Recall from §4.7 in Chapter 4 that δ assists a node to exponentially decrement or increment the reputation (direct, observed or recommended) of another node depending on, (a) *whether the reputation value is at least* Δ and, (b) *the period for which there has been no evidence.* Selectively-behaving nodes include both selectively well-behaving and misbehaving nodes, and the line of separation between them depends on whether their behaviour is benign or malicious.

Here, we present only the, (a) *variation caused by δ when there is no direct evidence for a neighbour* and, (b) *significance of direct evidence in categorising selectively-behaving neighbours.* Note that the same approach holds for the observed and recommended evidence. In parallel, we also demonstrate the relationship between values, (a) *neg(event, action)* and, (b) *pos(event)*, where the values correspond to the malicious and benign behaviours, respectively, and define the slope of the reputation curve that summarises those behaviours. However, we maintain the constraint that the value assigned for the malicious behaviour must be greater than the value assigned for the benign behaviour to make the malicious behaviour unattractive. We further introduce another constraint in which *pos(event)* is always greater than δ. This prevents malicious neighbours from taking advantage of the mobility to increase their reputations at the other nodes. It also prevents the lack of evidence from overriding the previously held opinion for benign neighbours.

5.4.5.1 Benign and Malicious Neighbours

In Figure 5.6, the fragment AB of the curve *'Benign (P = 0.02)'* summarises the evidence collected for a benign neighbour, where the slope of AB is determined by the values assigned for, (a) *benign behaviour*, $pos(event) = 0.02$ and, (b) *malicious behaviour*, $neg(event, action) = 0.1$. Since the curve saturates beyond the point B

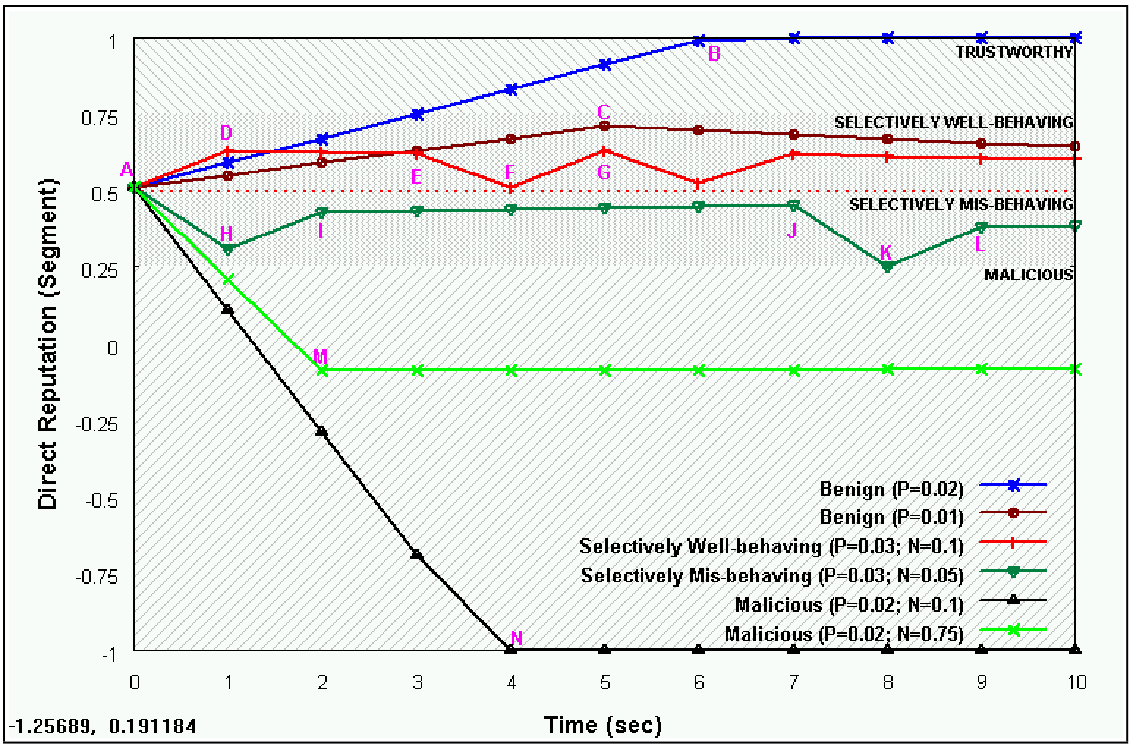


FIGURE 5.6: Analysis of Selective Behaviours.

(*i.e.* reaching the maximum reputation value +1), further evidence collected for the benign behaviours fails to influence the direct reputation for the reasons explained in §4.5.1 in Chapter 4. Alternatively, if the benign neighbour moves away, then the reputation begins to fade towards the floor of the segment S_1 (*i.e.* $+1 \geqslant S_1 > 0.75$) depending on the duration of the separation and δ (here, it is fixed at 0.01).

The curve *'Benign* ($P = 0.01$)*'* presents the effect of increasing the difference between *neg(event, action)* and *pos(event)*, where *pos(event)* now equates to δ. Such a pessimistic design would cause a benign neighbour to exhibit a consistently benign behaviour to earn a trustworthy reputation. In other words, the direct reputation of the benign neighbour falls within a selectively well-behaving region as seen from the fragment AC of the curve in Figure 5.6. Beyond the point C, the curve demonstrates how mobility degrades the direct reputation proportionally to the duration of the separation. The curves, (a) *'Malicious* ($N = 0.1$)*'* and, (b) *'Malicious* ($N = 0.75$)*'* imitate the characteristics of the curves, (a) *'Benign* ($P = 0.02$)*'* and *'Benign* ($P = 0.01$)*'* respectively, except for the malicious neighbours. However, they exhibit a steep down slope (AM and AN in Figure 5.6) because of the high negative values assigned for malicious behaviour, while the *pos(event)* is kept constant at 0.02.

5.4.5.2 Selectively-behaving Neighbours

In Figure 5.6, the benefit of categorising neighbours that exhibit sporadic malicious or benign behaviours can be observed from the regions, (a) *selectively well-behaving* and (b) *selectively-misbehaving.* The selectively well-behaving region hosts neighbours that perform irregular malicious behaviour and prevents them mixing up with the trustworthy neighbours. Although selectively well-behaving neighbours are unsuitable for highly sensitive communications, they can be suitable for less-sensitive communications. The curve *'Selectively well-behaving ($P = 0.03, N = 0.1$)'* summarises the evidence captured for a selectively well-behaving neighbour. It represents the neighbour's initial benign behaviours in the fragment AD and the effect of the reputation fading caused by the separation in fragment DE. Note that a single malicious behaviour is enough to bring down the reputation considerably (fragment EF), and then the neighbour is required to compensate the fall with benign behaviours (fragment FG) that are determined by the ratio of the *pos(event)* and the *neg(event, action).* Here, the *pos(event)* and *neg(event, action)* are fixed at 0.03 and 0.1, respectively.

On other hand, selectively-misbehaving region encompasses neighbours that are malicious but exhibit infrequent benign behaviours not to be blacklisted at other observing neighbours. Such selectively misbehaving neighbours may be considered for situations where there is no available path to a destination. The curve *'Selectively-misbehaving ($P = 0.03, N = 0.05$)'* summarises the evidence collected for a selectively-misbehaving neighbour, where the fragment AH presents the relaxed evaluation of the monitored malicious behaviour using the neg(event, action) = 0.05. The fragment HI denotes the ease with which the neighbour gains a direct reputation owing to the low difference between the *pos(event)* and *neg(event, action).* As the neighbour moves away, the neighbour's direct reputation grows by δ, rather than fading because the direct reputation is lower than Δ (§4.7 from Chapter 4). Note that the fragments JK and KL duplicate AH and HI, respectively.

5.4.6 Discussion

It is difficult to compare the performance of the SMRTI model with the related models, because the related models have been analysed using other simulators, such as the Glomosim [287], Opnet [280], Qualnet [378], and MatLab [249]. Furthermore, our simulation parameters reflect a large-scale simulation setup in terms of the total number of nodes, and a varying degree of V_{max}, pause time, simulation area and dynamic CBR flows. Also, the related models consider other routing protocols, such as the AODV. The Pirzada *et al.* model [308, 310] is closer to our simulation setup in terms of the simulator, attacker model, routing protocol and parameters such as, the V_{max}, pause time, transmission range, mobility model, traffic type, payload size and packet rate. Pirzada *et al.* fixed their simulation area to $1000*1000m^2$, the total number of nodes to 50 with a maximum of 20 malicious nodes, the CBR connections to 30, and simulation time to 3600 s.

The PDR for the Pirzada *et al.* model is marginally over 50% and almost near 40% in the presence of the 20% and 40% malicious nodes, respectively. In contrast, the SMRTI nodes outperform their model for the same proportion of the malicious nodes by delivering more than 70% of the CBR packets. This is because the combined effect of the SMRTI's detection component to detect malicious and benign neighbours and the reaction component to use only trustworthy neighbours for communication. In addition, the Pirzada *et al.* model incurs a 20% packet overhead and a 5% byte overhead caused by the dissemination of the trust values as the recommendations together with the route requests. Conversely, the SMRTI nodes do not incur any overhead, because they never disseminate the additional packets nor headers to communicate the recommendations.

In summary, it is evident that the SMRTI nodes are capable of not only defending against malicious nodes, but also of efficient eliminating the overhead. However, the SMRTI nodes will fail to exhibit a similar performance in the presence of spoofing attacks, for which an efficient key management mechanism and secure routing protocols are required to authenticate the identity of nodes. We will discuss the integration of the SMRTI nodes, key management and secure routing protocols in Chapter 7.

5.5 Conclusion

In this chapter, we have successfully analysed the performance of our novel trust management model known as the SMRTI using large-scale NS2-based simulations. First, we defined the scope of the simulation by describing the route modifications that include malicious increments of the route request's sequence number, and the addition or deletion of nodes into or from a route. We have then shown how adversaries can take advantage of route modifications to perform one of the following attacks, (a) *a route discovery disruption*, (b) *a detour*, (c) *a data flow disruption*, (d) *a blackhole*, (e) *a grayhole*, (f) *a route error disruption* and, (g) *a gratuitous detour*. Second, we established the necessary context for simulations by presenting a concise overview of the NS2 simulator, the wireless extension of the NS2, components of a wireless mobile node and additional tools. Given that there is no benchmark for setting up the simulation parameters, we reasoned the choice of the simulation parameters. We then discussed the optional extensions that are available in the DSR and presented valid arguments for the selection of the few extensions that are relevant to our analyses. In sequence, we also described the simulation parameters that were chosen for the SMRTI and malicious nodes. Finally, we described the various simulation scenarios and performance metrics that were considered for each scenario.

The SMRTI nodes demonstrate a superior performance (PDR and SRD) to the DSR nodes in all scenarios. In particular, the SMRTI nodes exhibit a better performance against the varying proportions of the malicious nodes for the following reasons, (a) *the detection of both benign and malicious neighbours using direct evidence and also from observed evidence even without interacting with those neighbours*, (b) *the identification of the multi-hop benign nodes using recommended evidence, such that the collected evidence is free from issues related to recommendations* and, (c) *the reactive decision policies that enhance the security of the routing.* We also found that the increase in the proportion of the malicious nodes decreases the performance of the SMRTI nodes owing to the lower proportion of the SMRTI nodes. We discovered that the SMRTI nodes deliver a higher performance at immobility, provided their proportion is at least equal to the total number of the malicious nodes. It is important to note that a similar performance was observed at the high pause time. Alternatively, if the proportion of malicious nodes exceeded the total number of the SMRTI nodes, then the mobility assisted the SMRTI nodes to enhance their performance. Note that similar characteristic was also observed at the low pause time. Although the SMRTI nodes demonstrate a peak performance at the high node densities, their performance falls once the network becomes sparsely connected, with a performance analogous to the DSR nodes when the network is completely disconnected. Since latency is proportional to the delivered performance, the SMRTI nodes exhibited a higher latency in comparison with the DSR nodes for the following reasons, (a) *a longer duration was required to discover the trustworthy routes at a high proportion of malicious nodes* and, (b) *the use of longer routes, because the SMRTI nodes routed around the malicious nodes.*

We have also covered the notion of uncertainty in a node's trust relationship with another node, which arises at the time of separation and in the absence of evidence. In particular, we have studied the role of δ in shaping a trust relationship during the absence of evidence. We have also pointed out the benefits of classifying selectively-behaving nodes into separate categories. The simulation parameters, such as δ, *pos(event)* and *neg(event, action)*, strongly depend on the environment for which the MANET is designed and, henceforth, we leave those parameters for further investigation for prototype implementers. One of the main goals of our simulation is to filter those parameters that require further investigation and strongly influence the level of the trust relationship established between the nodes. For instance, we believe that the initial reputation values strongly determine the manner in which the nodes establish the trust relationships among them. Furthermore, SMRTI's parameters are not uniform across the network, rather they vary from one node to another. For this reason, these parameters are not required to converge before the network stabilisation. At any given point in the network's lifetime, a SMRTI-enabled node's parameters neatly summarises its experiences with other nodes, but only based on the opportunities that were available to it to learn the behaviour of those nodes. In other words, more experiences translate to higher confidence and vice versa. Therefore, the accuracy of a SMRTI-enabled node's parameters is solely governed by the volume of experiences and not primarily on time. For instance, the SMRTI-enabled node may be positioned within the vicinity of several neighbours for a long period, but all these nodes could be bereaved from witnessing or routing any communication. In such a situation, the SMRTI-enabled node would not have learnt the behaviours of those neighbours even after being in their vicinity for a longer period.

Finally, we have also compared and contrasted our SMRTI model with one of the closely-related trust model from the literature. All these confirm that the SMRTI renders a promising solution to effectively defend against route modifications and thereby to enhance the security of the communications. In our future work, we foresee adapting the SMRTI to other reactive, proactive and hybrid protocols, apart from the integration of the SMRTI with a secure routing protocol in Chapter 7.

6

Trust Integrated Cooperation Architecture for MANET

6.1 Introduction

A review of the literature throws insight into the characteristics of detection-reaction based systems, (a) *their accuracy depends on how good their evidence collection approaches are* and, (b) *their effectiveness is as good as their decision-making policies.* Recall that such a capability was demonstrated in the SMRTI (Chapter 4), where it was confined to collecting evidence that was accurate and to capitalising on the inherent features of the MANET without imposing an additional layer of inter-nodal communication. In addition, the SMRTI also displayed a framework to evaluate such collected evidence and then to formulate the evaluation results into opinions for other mobile nodes. Remember that these opinions are later used to compute the trustworthiness of those mobile nodes. Most importantly, the SMRTI demonstrated the ability to identify the appropriate routing contexts for which the trust-based decisions are required, and then facilitated the trust-enhanced decisions for such routing contexts by determining the trustworthiness of the mobile nodes that are involved in them. Alternatively, Chapter 3 introduced and detailed our obligation-based fellowship technique to detect and defend against both packet drop and flooding attacks exhibited by adversaries such as the malicious and selfish nodes. Notably, both these models have been designed to take advantage of the inherent promiscuous feature of the MANET, where an embedded IDS-based monitor collects the evidence for the benign and malicious behaviours. However, the operations of the SMRTI and the fellowship have clearly indicated the possibility for the SMRTI and the fellowship to complement each other's design. In other words, they have displayed further space for enhancements in their design that could improve the security of the MANET.

For instance, the fellowship has shown a remarkable potential for defending against packet drop and flooding attacks. Regardless that it meets its design objective in Chapter 3, there is always adequate scope for improvement, especially when there is the possibility for the fellowship to enhance its service-oriented [1] decisions by taking advantage of the SMRTI's trust-based decisions. Similarly, the SMRTI has successfully demonstrated its capability to determining the trustworthiness of other mobile nodes based on the collected evidence, and thereby using such evidence-based trust metrics to make trust-enhanced decisions for the different routing contexts. In spite of attaining its design objective, the extensible design of the SMRTI has always 'called-in' for further improvements. These include innovative approaches that can be introduced to collect new types of evidence and the corresponding policies that can be used to make better trust-based decisions. We envisage that an integration of our fellowship and SMRTI models would open the door for the fellowship to feed the evidence for the DoS (packet

[1]Providing a relay service by forwarding packets on the behalf of previous-hop, and ensuring that the wireless medium is shared with its neighbours. In this chapter, we will commonly refer to this fellowship-based behaviour of a mobile node as *service*.

dropping and flooding) attacks into the SMRTI, so that the trustworthiness of the other mobile nodes are better evaluated. On the other hand, the SMRTI can assist the fellowship in making trust-enhanced decisions, such as whether to, (a) *provide a service to a previous-hop* and, (b) *trust a next-hop for propagating a packet on its behalf.*

Therefore in this chapter, we present the integration details of the fellowship and the SMRTI, especially from the perspective of the DSR's operations. In the next §6.2, we will review the fellowship from Chapter 3 and the SMRTI from Chapter 4 to establish the groundwork for the integration. Section 6.3 will describe the interface details of the fellowship and the SMRTI, and the possible design alternatives. We will then demonstrate the system operation from the interface perspective of DSR, fellowship and SMRTI in §6.4. Finally, §6.5 presents some concluding remarks.

6.2 Background

This section briefly summarises our fellowship and SMRTI for the sake of completeness and can be optionally skipped for the ease of continuity.

6.2.1 Fellowship Model

As stated in Chapter 3, the fellowship operates on the notion of the obligation that mobile nodes are required to contribute service towards a network to derive such service back from the network. The model itself is composed of three components, (a) *rate-limitation*, (b) *enforcement* and, (c) *restoration*.

The rate-limitation component in a fellowship-enabled mobile node has a pre-defined reception threshold (τ) for every previous-hop that can be located in its environment. Here, τ defines the bandwidth that is offered for forwarding packets on behalf of a previous-hop. The value of τ is indirectly conveyed to a previous-hop over a period of intervals, which is referred to as the negotiation threshold (ρ). Any previous-hop that fails to achieve its packet transmission rate with τ within ρ intervals is then forced to witness those packets that exceed τ being discarded per interval. In addition, the rate-limitation reduces its resource commitment (based on its contribution share (η))

for the previous-hop in proportion to every packet that is discarded after ρ^{th} interval, *i.e.* once the previous-hop is deemed as a flooder. Note that η is used to determine whether or not to forward packets on the behalf of the previous-hop in the future. Once the η for previous-hop becomes negative, the rate-limitation ignores all requests that are received from the previous-hop. This condition remains valid until the previous-hop demonstrates benign behaviours for a duration that is sufficient to exponentially raise its η to a positive value at the fellowship-enabled node. Although a uniform τ is assumed for every previous-hop, it can vary for each previous-hop depending on either the service offerred by the previous-hop in the past or the availability of a wireless medium for the packet transmission.

Analogous to the rate-limitation's τ and ρ, the enforcement component relies on the transmission threshold (ν) and the packet forwarding ratio (γ) to detect the packet dropping behaviour of the next-hop neighbours. It then deems them as packet droppers if η becomes negative. For every packet that is transmitted to the next-hop and for every packet that is subsequently forwarded by the next-hop on its behalf, the enforcement component updates γ. Whenever the enforcement component witnesses its γ for a next-hop falling below ν, it then deems the next-hop as a packet dropper and accordingly reduces its commitment for the next-hop by reducing η. Similar to the rate-limitation's decision to refrain from accepting packets from a previous-hop whenever η for the previous-hop is negative, the enforcement component refrains from transmitting packets to a next-hop whenever η for the next-hop is below zero.

Finally, a fellowship-enabled node's restoration component steps-in to add or contribute further service to a previous-hop if it had promised to forward packets on the behalf of previous-hop but had failed to do so owing to unforeseen conditions, such as contention or congestion. The restoration component completes the process by increasing its η for the previous-hop and notifying the rate-limitation to lower τ for all previous-hops. Refer to Chapter 3 for further details on how all these components react to different types of adversaries under various possible scenarios.

6.2.2 SMRTI Model

As stated in Chapter 4, the SMRTI focuses on establishing a trust management system for the MANET such that it operates within the framework of the MANET without imposing additional inter-nodal communications and extracts all available evidence from the network to make trust-enhanced routing decisions. In particular, the SMRTI collects evidence from the direct interactions with the neighbours, (a) *by observing the interactions that are confined to any two neighbours* and, (b) *deriving recommendations by deducing the relationships between any two multi-hop nodes from the manner in which one mobile node offers service to another.* The SMRTI then formulates such evidence into opinions that are later used to make decisions, such as, whether those mobile nodes are trustable in a routing context. Remember that the SMRTI relies on routing policies to identify mobile nodes that are involved in a routing context. In addition, the SMRTI also resolves the notion of ignorance that enters into trust relationships, especially when the nodes are separated from each other because of mobility.

6.3 Interface Integration

In this section, we discuss possible design alternatives for integrating the interfaces of the SMRTI and the fellowship models. In particular, we detail the interface integration between the components of the fellowship and the SMRTI.

It is well understood that the SMRTI forms the core of a detective-reactive layer in our envisaged TEAM. Similarly, the secure routing and key management systems are anticipated to form the core of the preventive layer of our TEAM (which will be discussed in the next chapter). However, the fellowship plays a unique role in our two-layered TEAM because of its internal structure. The fellowship is embedded with both preventive and detective-reactive approaches, contrary to the secure routing and key management systems that have been solely designed for prevention purposes and the IDS and the trust management systems that have been dedicated for detection-reaction purposes. Therefore, we consider this integration to be the initial point for sandwiching the prevention and detection-reaction layers of our envisaged TEAM.

6.3.1 Elimination of Contribution-share (η)

To begin with, η is depreciated from being symbolised as the representation of the network service that is offered to other mobile nodes. Recall from Chapter 3 that η is only used by a fellowship-enabled node to keep track of its resource that is committed for other mobile nodes. Note that the main objective of allocating a fixed resource for other mobile nodes is to offer a notable service so that the nature of those mobile nodes can be determined from their exhibited behaviours. In other words, as long as there exists an alternative metric that can facilitate a fellowship-enabled node to determine the behaviour of other mobile nodes, then the transition from η to a new alternative must be viable without affecting the core operation of the fellowship. Given that the SMRTI has successfully demonstrated the capability of its trust metric to measure and portrait the behaviour of the other mobile nodes, the SMRTI's trust metric is permitted to take over the functionality of η in our integrated model.

Therefore, instances where decisions (such as, whether to, (a) *accept packets from a previous-hop* and, (b) *forward packets to a next-hop*) were made depending on the condition whether η was negative, will from now onwards be over-ridden in the integrated model with the condition that whether or not the trust metric exceeds Δ. In sequence, the design replaces the exponential improvement of η (3.3) with the reputation-update operation (4.7) to facilitate the blacklisted upstream or downstream neighbours to rejoin the network, provided the neighbours exhibit benign behaviours since being blacklisted.

6.3.2 Amendments to Rate-limitation Component

Let us now look at the integration of the rate-limitation component into the detection and reaction components of the SMRTI. Recollect that the basic principle of collecting evidence for a previous-hop's behaviour and consequently deeming whether or not such evidence affirms that the previous-hop is a flooder, rests with the rate-limitation.

This is analogous to the approach deployed during the integration of an IDS-based monitor with the SMRTI. Note that the monitor holds the responsibility for promiscuously capturing packets from its environment and then determining whether those packets have undergone a malicious route modification before forwarding the evidence to the detection component. Henceforth, the rate-limitation is amended to forwarding to the SMRTI's detection component of all evidence corresponding to a previous-hop's flooding or complying behaviour. The evidence that is obtained from the direct interactions with the previous-hop is passed on to the SMRTI's detection component in the form of either *pos(packet reception)* or *neg(packet reception, flooding)* depending on the behaviour exhibited by the previous-hop. The SMRTI's detection component then aggregates the result (*i.e. pos(packet reception)* or *neg(packet reception, flooding)*) into the *accrued direct reputation* (4.4).

Although the fellowship's rate-limitation *asynchronously* interacts with the SMRTI's detection component, it *synchronously* queries the SMRTI's reaction component to determine the value for τ. Let us consider the possible instances that would be required for setting up a value for τ, (a) *initialising τ for all previous-hops with a uniform value*, (b) *updating τ for a previous-hop based on the service offered by the previous-hop in the past*, (c) *honouring a previous-hop's request to increase the value of τ or the offered service*, (d) *lowering τ for all previous-hops depending on the notification received from restoration component that the contention for transmission medium is high* and, (e) *increasing the value of τ for all previous-hops once the channel availability increases.* Among these possible cases, we believe that the last two cases warrant a node-specific τ rather than a blanket approach for raising or lowering the value of τ. This is appropriate, especially when a trust model exists to specify the trustworthiness of a previous-hop.

6.3.2.1 Case A

The first case requires the fellowship's rate-limitation to initialise a uniform value for τ for all unknown previous-hops. This is based on the confirmation from the SMRTI's reaction component that the trust metric for those previous-hops has been initialised to a default value. However, for a previous-hop — where the integration-enabled mobile node has never made a one-to-one interaction, but has collected observed and recommended evidence through promiscuous observations and recommendations in the past — the integration-enabled mobile node's SMRTI sub-system advocates the fellowship sub-system to initialise a value for τ that is proportional to the trustworthiness held for the previous-hop.

6.3.2.2 Case B

The rate-limitation performs a minimal interaction with the SMRTI's reaction component in this case because the integration-enabled mobile node is expected to respond back to a previous-hop with a service that is equivalent to the service derived from the previous-hop in the past. Although it can be argued that the service offered to a previous-hop should be proportional to the trustworthiness held for the previous-hop at any given point of time, the fellowship's notion of obligation that any service derived from a previous-hop must be reciprocated over-rides such a design alternative.

6.3.2.3 Case C

Whenever a previous-hop requests an integration-enabled mobile node to forward more packets on its behalf (or to increase the offered service), the integration-enabled mobile node's rate-limitation component interacts with the SMRTI's reaction component and honours or dishonours the request depending on the trustworthiness held for the previous-hop. An exception to the above may be expected if the mobile node is subjected to congestion. However, such an exception would violate this premise, because if the integration-enabled mobile node was not consistent in forwarding packets on the behalf of the previous-hop, then the previous-hop would not be submitting such a request to increase the service.

6.3.2.4 Cases D and E

These cases are the resultant of either the wireless medium being unavailable owing to congestion or being available, because of fewer neighbours being positioned in the environment. Note that the restoration component's notification about the wireless medium's contention rate serves as the indicator for channel availability to the rate-limitation component. Contrary to the approach deployed so far by the fellowship, the integration-enabled mobile node uses the SMRTI's reaction component to shortlist the neighbours based on their trustworthiness and then advocates the fellowship's rate-limitation to share the available bandwidth in proportion to their trustworthiness. This approach not only encourages cooperation with other mobile nodes, but also strengthens the trust relationships held with them.

6.3.3 Amendments to Enforcement Component

Unlike the rate-limitation, the enforcement component requires minimal changes for the integration into the SMRTI's detection and reaction components. Note that the rate-limitation component is supposed to query the SMRTI's reaction component to gain knowledge of a previous hop's trustworthiness. Alternatively, the enforcement component can be tailored to *synchronously* query the trustworthiness of a previous-hop and a next-hop before accepting a packet for forwarding on the behalf of the previous-hop and prior to transmitting a packet to the next-hop for propagation. Given that the trustworthiness of the previous-hop and the next-hop exceeds Δ and the next-hop's γ exceeds ν, the enforcement component moves the packet to the restoration component. It proactively measures ν for every packet that is transmitted to the next-hop via the restoration component. For the instance where γ fails to meet ν, the enforcement component *asynchronously* reports the evidence for the packet drop behaviour to the SMRTI's detection component, which is then aggregated to the *accrued direct reputation* held for the next-hop (4.4). Otherwise the enforcement component imitates the operations detailed in Chapter 3.

6.3.4 Amendments to Restoration Component

Similar to the enforcement component, the restoration component inherits all those operations that were detailed in Chapter 3. However, the only improvement that is

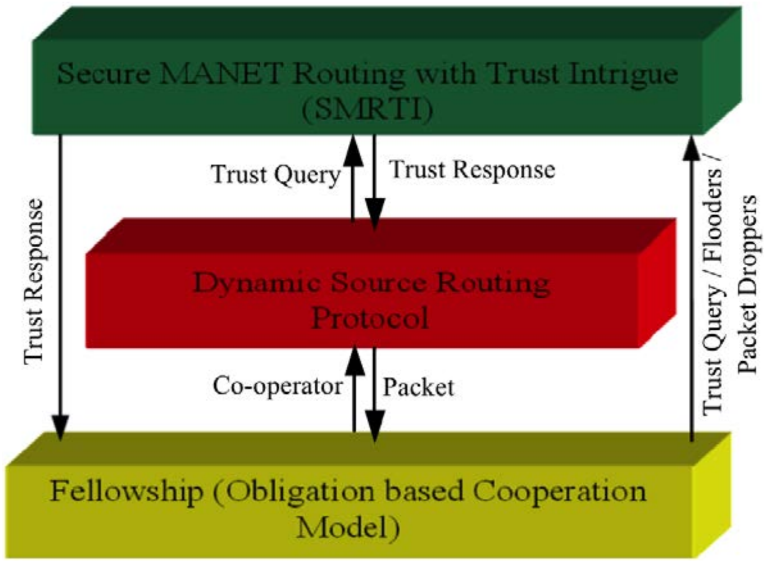

Secure MANET Routing with Trust Intrigue (SMRTI)
Trust Query
Trust Response
Trust Response
Dynamic Source Routing Protocol
Trust Query / Flooders / Packet Droppers
Co-operator
Packet
Fellowship (Obligation based Cooperation Model)

Figure 6.1: Integrated Architecture of SMRTI and Fellowship.

incorporated into the restoration component because of the integration with the SMRTI is to prioritise the packets for retransmission based on the trustworthiness held for the previous-hop neighbours.

6.4 System Operation

This section presents a detailed system operation between the integrated model (the fellowship and the SMRTI) and the DSR as shown in Figure 6.1. The system operation details the different interactions that take place among the interfaces of the SMRTI, the fellowship and the DSR. Note that it is exhaustive to cover all possible scenarios in this section, given that the significant cases have been already discussed in previous chapters. To begin with, whenever an integration-enabled mobile node receives packets from a previous-hop, the fellowship's rate-limitation component becomes activated to determine the trustworthiness of the previous-hop and also to measure the rate of the incoming packets. So long as the SMRTI's reaction component responds to the fellowship's rate-limitation with a note that the previous-hop is trustable and provided the previous-hop's requests are within τ those incoming packets are passed on to the DSR.

However, the same may not hold true if the previous-hop is not trustworthy, in which case, those incoming packets are discarded. Alternatively, the previous-hop's packets that exceed τ are discarded for ρ intervals, and after the ρ^{th} interval, for every packet that is discarded for the same reason, the rate-limitation reports the flooding behaviour to the SMRTI's detection component as a *direct evidence* towards the previous-hop's flooding behaviour. Recall that a fellowship-enabled previous-hop would always learn τ within ρ intervals and reduce its packet transmission rate accordingly. The same holds true for a selfish previous-hop, because being in the network is its main objective, rather than hindering the network's functionality.

Once the previous-hop has been confirmed as trustworthy and those requests received from the previous-hop are deemed to be within τ, the DSR interacts with the SMRTI's reaction component to learn the trustworthiness of those packets. In other words, the DSR requests the SMRTI to evaluate the trustworthiness of the source and the destination nodes. Recall that an intermediate mobile node participates in a communication flow only for the sake of propagating packets to and fro between the source and the destination nodes. Provided a packet is trusted, the DSR then passes the packet on to the enforcement component with a note to deliver the packet to the next-hop as directed by the route.

Similar to the fellowship's rate-limitation component, the enforcement component is expected to poll the SMRTI's reaction component to determine the trustworthiness of the next-hop. Given that the next-hop is trustable, the enforcement component then moves the packet to the restoration component that, in turn, polls the availability of the wireless medium for transmitting the packet to the next-hop. In sequence, the enforcement component collects evidence for the next-hop's cooperation with the help of an IDS-based monitor, *i.e.* whether the next-hop forwards the packet on its behalf. For the instance, where the next-hop fails to show a sign of cooperation, the enforcement component is tailored to check for a list of items before deeming the next-hop as a packet dropper. The check includes, (a) *the contention rate for the wireless medium*, (b) *the γ for the next-hop*, (c) *the rate at which requests are transmitted to the next-hop* and, (d) *whether the next-hop forwards other packets on its behalf that are destined* *to a different downstream node.* As mentioned in Chapter 3, if the next-hop fails to convince that the packet was not dropped deliberately, the enforcement component then communicates the direct evidence for next-hop's packet dropping behaviour to the SMRTI's detection component. Note that this communication of information to the SMRTI's detection component takes place for every packet that is confirmed to be dropped by the next-hop. Finally, the restoration component takes the availability of the wireless medium into account to tune the rate at which the packets are transmitted to the next-hop or received from the previous-hop, where the tuning is proportional to the trustworthiness held for the next-hop and the previous-hop, respectively.

6.5 Conclusion

In this chapter, we have successfully shown the capability of both the SMRTI and the fellowship to complement each other's design. For instance, the fellowship has shown its precise potential to take advantage of the SMRTI's trust-enhanced decisions. On the other hand, the SMRTI has effectively demonstrated its capacity to integrate the evidence received from the fellowship so that the evaluation of the trustworthiness of the mobile nodes is improved. This has been further demonstrated in the presence of the DSR. However, the performance evaluation of the fellowship-integrated SMRTI is deferred to Chapter 9, where subjective-logic is incorporated into the SMRTI and the performance analysis is then evaluated in the presence of the AODV protocol.

7 TEAM: Trust Enhanced Security Architecture for MANET

7.1 Introduction

Several secure routing protocols [21, 75, 76, 105, 120, 136, 150, 154, 155, 159, 161–164, 179, 180, 215, 250, 260, 282, 284, 337, 367, 397, 399, 439, 440, 445, 446, 455] have been proposed for the MANET to authenticate the intermediate nodes and to verify the integrity of the discovered paths. However, the success and failure of these secure routing protocols relies heavily on the feasibility of a key management scheme adopted for their operation. Recall from Chapter 2 that key management schemes in MANET can be classified into three categories, (a) *a public-key based scheme*, (b) *a symmetric-key based scheme* and, (c) *a group-key based scheme*. Regardless that these key management schemes attempt to establish secret associations between the mobile nodes through various approaches (such as, certificate-based [18, 100, 166], threshold-cryptography based [74, 101, 202, 203, 218, 240, 268, 293, 343, 344, 423, 428, 437, 451, 458, 465], identity-based [54, 55, 96, 110, 152, 196, 431], self-generated certificates [396, 398], key imprinting [37, 360], *etc*), they either fail to meet the scalability requirement or to provide a feasible bootstrap mechanism. Otherwise, they struggle to accommodate the self-organised characteristic of the MANET, where the nodes are not only mobile, but can also enter and leave the MANET at any time during the lifetime of the network. The issue is further aggravated in the MANETs that allow the inclusion of repenting malicious nodes or reclaimed compromised nodes. Furthermore, they also struggle in the area of revoking or refreshing secret associations. Therefore, in this chapter, we adopt an appropriate key management scheme that meets the self-organised characteristic of the MANET and accordingly tailors a secure routing protocol depending on whether the network is a pure or managed MANET.

In the following section, we will recommend the key management mechanism and secure routing protocol that can be deployed for the pure MANETs. In §7.3, we will present the key management mechanism adapted for the managed MANETs and then our secure routing protocol in §7.4. Section 7.5 will present the interface integration between the SMRTI, and key management and secure routing sub-systems in a managed MANET. Finally, we will conclude this chapter with some remarks and anticipated future work in §7.6.

7.2 Pure MANET – Preventive Mechanisms

In the case of a pure MANET, we assume that an offline CA exists for bootstrapping secure communications among the mobile nodes. This is feasible because a mobile node's certificate is used only for the purpose of authentication so that spoofing-related attacks can be defeated prior to their deployment. Alternatively, an identity-based key management scheme can be adopted for authenticating the mobile nodes. Although several secure routing protocols exist that can successfully exploit this setup, we refer to the BISS [397] for its capability to establish secure associations in the MANET. It then transforms its operation to the Ariadne [164] if secure associations exist among all the mobile nodes that participate in a route discovery. Alternatively, the BISS can be made to transform its operation to the SRP [282] with minor modifications for cases where end-to-end authentication is required instead of the hop-by-hop based authentication resulting from the battery constraints. A concise explanation of these secure routing protocols can be found in Chapter 2.

7.3 SMG: Scalable Multi-service Key Management

Nevertheless, public-key based schemes that are used for broadcasting authentication during a route discovery phase cannot provide access control for sensitive communications. The access control for communications may be a requirement for the managed MANETs where communications can be categorised based on their sensitivity level, and therefore would require mobile nodes with the appropriate security clearance to participate in such sensitive communications. For example, the MANETs that are deployed as a part of a military operation may reflect such a property. Another example is where a source node seeks a route discovery to a destination but presets a condition that only trusted intermediate nodes can participate in the communication. Therefore, we adapt Zhang et. al.'s [448] ***Scalable Multi-service Group Key Management Scheme (SMG)*** to the MANET, which is an ID-based key management scheme that has an inherent traitor-tracing scheme. Furthermore, each mobile node is required to have only one private key that remains constant throughout its lifetime. The key management scheme has other properties, such as, (a) *a scalability that enables mobile nodes to join and leave the network at their will*, (b) *it guarantees forward and backward secrecy* and, (c) *it permits the addition of new sensitivity levels for communications without the need for any rekeying process*. We will then propose a secure routing protocol that uses the SMG to secure the communications based on the sensitivity level requested by the source mobile nodes.

In the following, we briefly introduce the system setup at a mobile node and the encryption and decryption process between the mobile nodes. We then present the adaptations that emerge from the self-organised characteristics of the MANET. Appendix D presents further details for the encryption and decryption processes, and [448] can be referred for an in-depth analysis of the SMG.

7.3.1 System Initialisation

In our adaptation of the SMG, a mobile node establishes different levels of sensitivity for the communications by labelling or grouping other mobile nodes according to those sensitivity levels. Let l denote the cardinality of the different sensitivity levels[1] applied for its communications that are denoted as $r_1, r_2, \cdots, r_l$. For each sensitivity level, the mobile node then groups other mobile nodes into a *Communication Group (CG)* that is denoted as $CG_1, CG_2, \cdots, CG_l$. For instance, the sensitivity levels may be considered as *unclassified* (r_1), *classified* (r_2), *confidential* (r_3), *protected* (r_4), *secret* (r_5) and *top secret* (r_6), and then the corresponding communication groups are $CG_1, CG_2, CG_3, CG_4, CG_5$ and CG_6 respectively. The mobile node can dynamically join, leave and switch other mobile nodes from one communication group to another, dynamically based on the feedback from the SMRTI, which will be discussed in §7.4.

Prior to the network deployment, a mobile node generates various parameters and a matrix *S($\mathcal{A}$)* for managing its communication groups. Let us explore those parameters in the context of a mobile node $\mathcal{A}$ and the reviewers are encouraged to refer to [448] for completeness.

- A master secret key $s^{\mathcal{A}} \in \mathbb{Z}$
- A public key $P^{\mathcal{A}}_{pub} = (s^{\mathcal{A}} P^{\mathcal{A}})$ that is based on an ID-based encryption algorithm [56]. Here $P^{\mathcal{A}} \in \mathbb{G}^{\mathcal{A}}_1$, where $\mathbb{G}^{\mathcal{A}}_1$ is an additive group and alternatively $\mathbb{G}^{\mathcal{A}}_2$ is a

[1]Since the cardinality of the sensitivity level is proportional to the application to which a MANET is applied, specific details related to the types of sensitivity levels are left to the implementer. However, the lowest level that can be visualised for a communication is where no restrictions apply and any mobile node is allowed to participate as an intermediary in the communication.

multiplicative group. These groups have an order $p^{\mathcal{A}}$, such that $p^{\mathcal{A}} = (2q^{\mathcal{A}}) + 1$, and $p^{\mathcal{A}}$ and $q^{\mathcal{A}}$ are large primes.

- A participant key $S_{\mathcal{I}}^{\mathcal{A}} = (s^{\mathcal{A}} Q_{\mathcal{I}}^{\mathcal{A}})$ for another mobile node $\mathcal{I}$, where $Q_{\mathcal{I}}^{\mathcal{A}} = H_1(\mathcal{I})$. Here $H_1(\mathcal{I})$ is a one-way function, such that $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1^{\mathcal{A}}$ and alternatively $H_2 : \mathbb{G}_1^{\mathcal{A}} \rightarrow \{0,1\}^*$,
- A matrix S of order $n * l$, $S(\mathcal{A}) = \begin{pmatrix} S_{11}^{\mathcal{A}} & S_{12}^{\mathcal{A}} & \cdots & S_{1k}^{\mathcal{A}} & \cdots & S_{1l}^{\mathcal{A}} \\ S_{21}^{\mathcal{A}} & S_{22}^{\mathcal{A}} & \cdots & S_{2k}^{\mathcal{A}} & \cdots & S_{2l}^{\mathcal{A}} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ S_{\mathcal{M}1}^{\mathcal{A}} & S_{\mathcal{M}2}^{\mathcal{A}} & \cdots & S_{\mathcal{M}k}^{\mathcal{A}} & \cdots & S_{\mathcal{M}l}^{\mathcal{A}} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ S_{\mathcal{N}1}^{\mathcal{A}} & S_{\mathcal{N}2}^{\mathcal{A}} & \cdots & S_{\mathcal{N}k}^{\mathcal{A}} & \cdots & S_{\mathcal{N}l}^{\mathcal{A}} \end{pmatrix}$

In the above matrix *S($\mathcal{A}$)*, $S_{\mathcal{M}k}^{\mathcal{A}} = 1$, if the mobile node $\mathcal{M}$ is considered for a communication flow $CG_k^{\mathcal{A}}$ and $S_{\mathcal{M}k}^{\mathcal{A}} = 0$ for the opposite, where $1 \leqslant \mathcal{M} \leqslant n$. Parameter n is initialised to twice the size of anticipated network for reasons to be explained later, while k denotes the sensitivity of a communication flow, such that, $1 \leqslant k \leqslant l$.

Before mobile nodes can enter into the network, they are pre-loaded with $\mathcal{A}$'s public key $P_{pub}^{\mathcal{A}}$, and each one of them (*i.e.* node $\mathcal{I}$) with $\mathcal{A}$'s participant key $S_{\mathcal{I}}^{\mathcal{A}}$. Note that $\mathcal{A}$ authorises the participation of $\mathcal{I}$ in its communication, for instance in $CG_k^{\mathcal{A}}$ by activating $S_{\mathcal{I}k}^{\mathcal{A}} = 1$ for which it includes $S_{\mathcal{I}}^{\mathcal{A}}$ during the encryption process. Similar to $\mathcal{A}$, every mobile node $\mathcal{J}$ generates its parameters and matrix *S($\mathcal{J}$)*, and pre-loads $P_{pub}^{\mathcal{J}}$ and $S_{\mathcal{I}}^{\mathcal{J}}$ on to other mobile nodes (for example, node $\mathcal{I}$) before the deployment of network.

7.3.2 Encryption-Decryption Process

Let $SK_k^{\mathcal{A}} \in \{0,1\}^*$ be the session secret key used for a communication flow that belongs to group $CG_k^{\mathcal{A}}$. As shown in (D.4) in Appendix D, $Auth_k^{\mathcal{A}}$ can be generated and sent to mobile nodes that belongs to the group $CG_k^{\mathcal{A}}$. All mobile nodes that are authorised by node $\mathcal{A}$ would be able to decrypt $Auth_k^{\mathcal{A}}$ in order to obtain $SK_k^{\mathcal{A}}$ and then would be able to authenticate $\mathcal{A}$.

On receiving an $Auth_k^{\mathcal{A}}$ embedded route request, a mobile node can decrypt the session key as shown in (D.5) in Appendix D. However, if a mobile node $\mathcal{V}$ is not chosen as a participant for the communication flow that belongs to $CG_k^{\mathcal{A}}$ by the source node $\mathcal{A}$, then $\mathcal{V}$ will not be able to get the session key. This is because for any $x_{\mathcal{V}}^{\mathcal{A}}$ that does not belong to $CG_k^{\mathcal{A}}$, then the following holds true: $\sum_{\mathcal{I}=0}^{\mathcal{M}} (a_{\mathcal{I}k}^{\mathcal{A}} x_{\mathcal{V}k}^{\mathcal{A}}) \neq 0 \bmod p^{\mathcal{A}}$ as given in Appendix D.

7.3.3 Discussion

In [448], Zhang *et al.* refer to the *Bilinear Diffie-Hellman (BDH)* and the *factorisation problems* to characterise the security of this scheme. Note that the security of the BDH has been proved using a *random oracle model* in [56]. Let us now study the possibilities that may affect the system setup, (a) *the addition or deletion of a sensitivity level*, (b) *the inclusion or removal of mobile nodes from the MANET* and, (c) *dynamically varying the selection of mobile nodes for a communication flow.*

From the perspective of a mobile node $\mathcal{A}$, the addition or deletion of a sensitivity level can be administered by simply including or removing a column to its matrix *S($\mathcal{A}$)* (§7.3.1). However, such additions or deletions may not be required in a managed MANET where the sensitivity levels for the communication flows are expected to be predefined before the network deployment. A managed network that is void of sensitivity levels (*i.e.* where all communication flows are treated at the same level) is characterised by mobile nodes having a matrix of the order $n * l$, with $l = 1$.

Let us now consider the case where nodes can enter and leave the network at will. In this case, the inclusion or removal of nodes from the network must produce a negligible impact in the matrix *S($\mathcal{I}$)* of any participating mobile node $\mathcal{I}$, because of the assumption that n in $n * l$ is initialised to twice the size of network. Therefore, even if the network size is only $\frac{n}{2}$, the design to initialise every mobile node $\mathcal{I}$ with parameters that are scalable up to n occupants implicitly resolves the raised concern. For example, if a new mobile node $\mathcal{V}$ (such that $\frac{n}{2} < \mathcal{V} \leqslant n$) is introduced into the network, then a mobile node $\mathcal{A}$ would be able to implicitly include or exclude $\mathcal{V}$ in its communication flow that belongs to a sensitivity level k, (where $1 \leqslant k \leqslant l$) by simply enabling or disabling $S^{\mathcal{A}}_{\mathcal{V}k}$ in its matrix $S(\mathcal{A})$. A similar argument holds valid for removing a mobile node $\mathcal{H}$ (where $1 \leqslant \mathcal{H} \leqslant \frac{n}{2}$) from the network.

Finally, choosing a mobile node $\mathcal{V}$ for a communication flow that has a sensitivity level e (such that $1 \leqslant e < l$) and then switching over $\mathcal{V}$ to another communication flow that has a different sensitivity flow f (such that $e < f \leqslant l$) is characterised by enabling or disabling $S^{\mathcal{A}}_{\mathcal{V}e}$ and $S^{\mathcal{A}}_{\mathcal{V}f}$. However, such a switch over is not without cost, if the mobile nodes are expected to perform the computation given in (D.4) in Appendix D for every change in $CG^{\mathcal{A}}_{k}$. Alternatively, the mobile nodes can be pre-loaded with the result of such computations prior to the network deployment that would not only save their computation overhead but also facilitate their scalability. Although the heavy communication overhead can be off-loaded to the pre-deployment stage, the scheme is subject to a significant storage overhead caused by the requirement to store the following items, (a) *the participant keys $S^{\mathcal{A}}_{\mathcal{I}}$ that are required for every other mobile node $\mathcal{I}$*, (b) *the public keys $P^{\mathcal{I}}_{pub}$ of every other mobile node $\mathcal{I}$* and, (c) *the participant key $S^{\mathcal{I}}_{\mathcal{A}}$ that is generated by ever other mobile node $\mathcal{I}$.* Such a storage overhead should be a negligible concern, given the remarkable growth in storage mediums in recent years.

7.4 Scasec: Secure Routing in Managed MANET

Considering the DSR as the base routing protocol, the route discovery has two stages, (a) *the source mobile node floods the network with route request (RREQ) packets* and (b) *the destination mobile node responds back with a route reply (RREP) packet to the source node.* In this section, we present our *'**Sca**lable multi-service group key based **sec**ure routing protocol'* for the DSR that is referred as the ***Scasec***.

The Scasec is designed to provide the following, (a) *a destination to authenticate the source and intermediate nodes and also to verify the integrity of the discovered route before participating in a communication flow*, (b) *a source to authenticate the destination and intermediate nodes and also to verify the integrity of the discovered route before accepting a route for communication flow* and, (c) *the intermediate nodes*

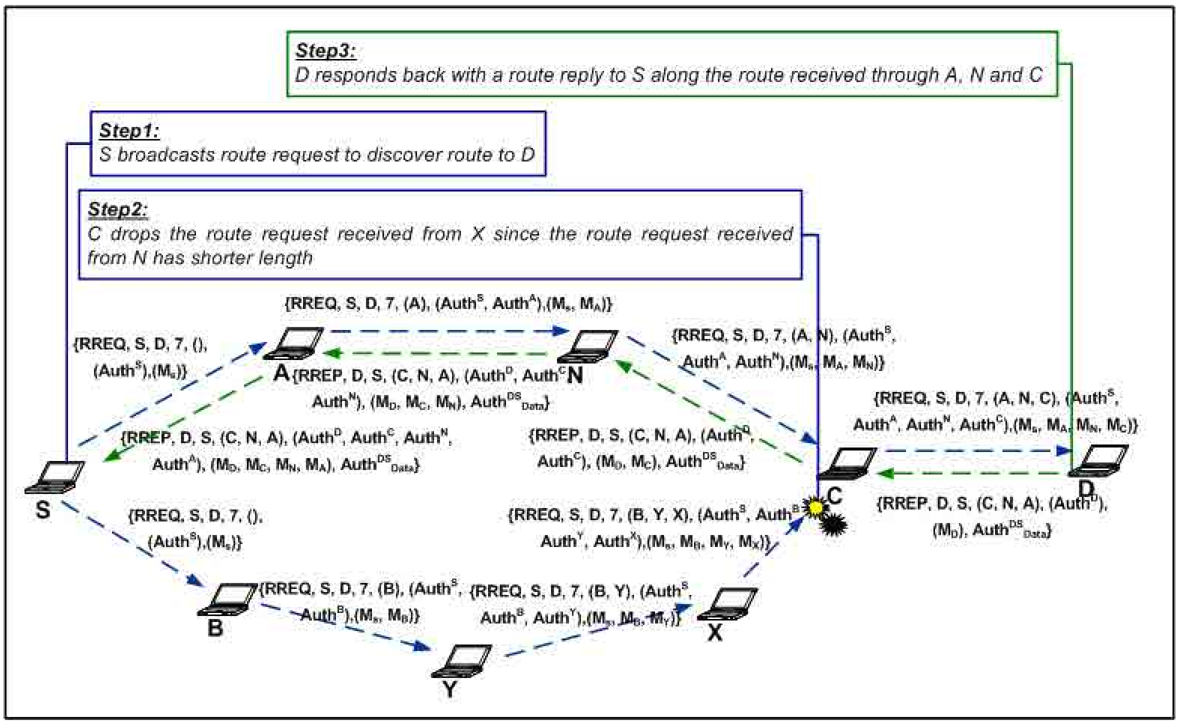

Step3:
D responds back with a route reply to S along the route received through A, N and C
Step1:
S broadcasts route request to discover route to D
Step2:
C drops the route request received from X since the route request received from N has shorter length
{RREQ, S, D, 7, (A), (Auth^S, Auth^A),(M$_s$, M$_A$)}
{RREQ, S, D, 7, (), (Auth^S),(M$_s$)}
{RREP, D, S, (C, N, A), (Auth^D, Auth^C, Auth^N), (M$_D$, M$_C$, M$_N$), Auth$^{DS}{}_{Data}$}
{RREQ, S, D, 7, (A, N), (Auth^S, Auth^A, Auth^N),(M$_s$, M$_A$, M$_N$)}
{RREQ, S, D, 7, (A, N, C), (Auth^S, Auth^A, Auth^N, Auth^C),(M$_s$, M$_A$, M$_N$, M$_C$)}
{RREP, D, S, (C, N, A), (Auth^D, Auth^C, Auth^N, Auth^A), (M$_D$, M$_C$, M$_N$, M$_A$), Auth$^{DS}{}_{Data}$}
{RREP, D, S, (C, N, A), (Auth^D, Auth^C), (M$_D$, M$_C$), Auth$^{DS}{}_{Data}$}
{RREQ, S, D, 7, (), (Auth^S),(M$_s$)}
{RREQ, S, D, 7, (B, Y, X), (Auth^S, Auth^B, Auth^Y, Auth^X),(M$_s$, M$_B$, M$_Y$, M$_X$)}
{RREP, D, S, (C, N, A), (Auth^D), (M$_D$), Auth$^{DS}{}_{Data}$}
{RREQ, S, D, 7, (B), (Auth^S, Auth^B),(M$_s$, M$_B$)}
{RREQ, S, D, 7, (B, Y), (Auth^S, Auth^B, Auth^Y),(M$_s$, M$_B$, M$_Y$)}
S
A
N
C
D
B
Y
X

Figure 7.1: Route Discovery using Scasec.

to authenticate the originator of the control packets (RREQ or RREP) and also all the upstream mobile nodes together with the integrity of the discovered route. The aim of such a design is to enable the Scasec to prevent the modification and fabrication attacks. Note that the modification attacks include the insertion or deletion of control information to or from the route headers, while the fabrication attacks focus on the injection of spoofed control packets. The notations used in Scasec are as follows:

- $SK^{\mathcal{S}}$ denotes the secret key (§7.3.2) used by node $\mathcal{S}$ to generate the message authentication code $HMAC$,
- $HMAC_{SK^{\mathcal{S}}}(M)$ denotes the computation of the HMAC for message M using the key $SK^{\mathcal{S}}$,
- $M_{\mathcal{S}}$ is the resultant $HMAC_{SK^{\mathcal{S}}}(M)$ generated by node $\mathcal{S}$.

7.4.1 Route Request (RREQ) Propagation in Scasec

Let us consider the scenario shown in Figure 7.1, where node $\mathcal{S}$ initiates a route discovery to node $\mathcal{D}$ and the corresponding RREQ traverses along the paths, (a) $\mathcal{S} \mapsto \mathcal{A} \mapsto \mathcal{N} \mapsto \mathcal{C} \mapsto \mathcal{D}$ and (b) $\mathcal{S} \mapsto \mathcal{B} \mapsto \mathcal{Y} \mapsto \mathcal{X} \mapsto \mathcal{C} \mapsto \mathcal{D}$. Let us also assume that no sensitivity level is marked for this communication flow. In other words, every node $\mathcal{I}$ that participates in the route discovery (including the source and destination) generates a broadcast authenticator $Auth^{\mathcal{I}}$ that can be authenticated by every other mobile node $\mathcal{J}$ in the network. For instance, node $\mathcal{S}$ generates $Auth_1^{\mathcal{S}}$ by setting up $S_{\mathcal{J}1}^{\mathcal{A}} = 1$ for all other mobile nodes in its matrix *S(*$\mathcal{S}$*)*, where $\{(1 \leqslant \mathcal{J} \leqslant n) \wedge (\mathcal{J} \neq \mathcal{S})\}$ and $CG_1^{\mathcal{A}}$ that corresponds to the first column of matrix *S(*$\mathcal{S}$*)* represents a communication flow of zero sensitivity. For simplicity, we denote the authenticator $Auth_1^{\mathcal{S}}$ of a communication flow that has a zero sensitivity level as $Auth^{\mathcal{S}}$ and hence $Auth^{\mathcal{I}}$ for $Auth_1^{\mathcal{I}}$. Therefore, we defer the possibility of a source node establishing a communication flow that has a specific sensitivity level k and the pre-determination of intermediate nodes that would be included in the communication group $CG_k^{\mathcal{A}}$ to the §7.5. More appropriately, such cases will be discussed during the interface integration of the SMG and the Scasec with the SMRTI, because the trustworthiness of the mobile nodes is warranted before considering the mobile nodes for a communication flow that has the sensitivity level k, where $1 \leqslant k \leqslant l$.

As seen from Table 7.1, an RREQ contains the following fields, ⟨*RREQ tag, initiator, target, RREQ identifier, intermediate node list, authenticator list, HMAC list*⟩. *Initiator* and *target* refer to the source and destination mobile nodes in a communication flow. Similar to the DSR, the source sets the *RREQ identifier* to a value that is fresh while initiating a route discovery. The *intermediate node list* represents the path taken by an RREQ, while the *authenticator list* contains a list of the encrypted secret keys that are appended by every intermediate mobile node when the RREQ propagates from the source. The *HMAC list* holds the HMAC generated by every intermediate mobile node using the secret key encrypted within the authenticator list.

When source $\mathcal{S}$ broadcasts the RREQ in Scasec, its neighbours authenticate $\mathcal{S}$ via

$\mathcal{S}:$	$M_{\mathcal{S}} = HMAC_{SK^{\mathcal{S}}}\Big(RREQ, \mathcal{S}, \mathcal{D}, id, (), (Auth^{\mathcal{S}}), ()\Big)$
$\mathcal{S} \rightarrow *:$	$\Big\{RREQ, \mathcal{S}, \mathcal{D}, id, (), (Auth^{\mathcal{S}}), (M_{\mathcal{S}})\Big\}$
$\mathcal{A}:$	$M_{\mathcal{A}} = HMAC_{SK^{\mathcal{A}}}\Big(RREQ, \mathcal{S}, \mathcal{D}, id, (A), (Auth^{\mathcal{S}}, Auth^{\mathcal{A}}), (M_{\mathcal{S}})\Big)$
$\mathcal{A} \rightarrow *:$	$\Big\{RREQ, \mathcal{S}, \mathcal{D}, id, (\mathcal{A}), (Auth^{\mathcal{S}}, Auth^{\mathcal{A}}), (M_{\mathcal{S}}, M_{\mathcal{A}})\Big\}$
$\mathcal{N}:$	$M_{\mathcal{N}} = HMAC_{SK^{\mathcal{N}}}\Big(RREQ, \mathcal{S}, \mathcal{D}, id, (\mathcal{A}, \mathcal{N}), (Auth^{\mathcal{S}}, Auth^{\mathcal{A}}, Auth^{\mathcal{N}}), (M_{\mathcal{S}}, M_{\mathcal{A}})\Big)$
$\mathcal{N} \rightarrow *:$	$\Big\{RREQ, \mathcal{S}, \mathcal{D}, id, (\mathcal{A}, \mathcal{N}), (Auth^{\mathcal{S}}, Auth^{\mathcal{A}}, Auth^{\mathcal{N}}), (M_{\mathcal{S}}, M_{\mathcal{A}}, M_{\mathcal{N}})\Big\}$
$\mathcal{C}:$	$M_{\mathcal{C}} = HMAC_{SK^{\mathcal{C}}}\Big(RREQ, \mathcal{S}, \mathcal{D}, id, (\mathcal{A}, \mathcal{N}, \mathcal{C}), (Auth^{\mathcal{S}}, Auth^{\mathcal{A}}, Auth^{\mathcal{N}}, Auth^{\mathcal{C}}), (M_{\mathcal{S}}, M_{\mathcal{A}}, M_{\mathcal{N}})\Big)$
$\mathcal{C} \rightarrow *:$	$\Big\{RREQ, \mathcal{S}, \mathcal{D}, id, (\mathcal{A}, \mathcal{N}, \mathcal{C}), (Auth^{\mathcal{S}}, Auth^{\mathcal{A}}, Auth^{\mathcal{N}}, Auth^{\mathcal{C}}), (M_{\mathcal{S}}, M_{\mathcal{A}}, M_{\mathcal{N}}, M_{\mathcal{C}})\Big\}$

Table 7.1: Route Request Broadcast in Scasec.

$Auth^{\mathcal{S}}$ and then the integrity of the RREQ using $M_{\mathcal{S}}$ by extracting $SK^{\mathcal{S}}$ from $Auth^{\mathcal{S}}$. Subsequently, they respond back to $\mathcal{S}$ with a route for $\mathcal{D}$ if one exists. Alternatively, if no route exists for $\mathcal{D}$, they re-broadcast the RREQ after appending their identity to the intermediate node list, $Auth^{\mathcal{I}}$ to authenticator list, and $M_{\mathcal{I}}$ to the HMAC list. For instance, node $\mathcal{A}$ appends its identity, $Auth^{\mathcal{A}}$, and $M_{\mathcal{A}}$ before re-broadcasting the RREQ. Therefore, all downstream nodes that follow node $\mathcal{A}$ would not only be able to authenticate both $\mathcal{S}$ and $\mathcal{A}$, but also the integrity of the route using $M_{\mathcal{S}}$ and $M_{\mathcal{A}}$. Once $\mathcal{D}$ receives the RREQ, it imitates the operations carried out by the upstream intermediate nodes to authenticate and verify the discovered path, except that it does not append any fields to the RREQ, because the RREQ is not propagated further. On authenticating $\mathcal{S}$ and all upstream intermediate nodes and conducting multiple verification of the route using the HMAC list, $\mathcal{D}$ records the route in its route cache for responding back with a RREP. Note that the failure of either authentication or verification of the RREQ at any point during the hop-by-hop travel would prevent the RREQ from reaching $\mathcal{D}$.

$\mathcal{D}:$	$M_{\mathcal{D}} = HMAC_{SK^{\mathcal{D}}}\Big(RREP, \mathcal{D}, \mathcal{S}, (\mathcal{C}, \mathcal{N}, \mathcal{A}), (Auth^{\mathcal{D}}), (), Auth^{\mathcal{DS}}_{Data}\Big)$
$\mathcal{D} \rightarrow \mathcal{C}:$	$\Big\{RREP, \mathcal{D}, \mathcal{S}, (\mathcal{C}, \mathcal{N}, \mathcal{A}), (Auth^{\mathcal{D}}), (M_{\mathcal{D}}), Auth^{\mathcal{DS}}_{Data}\Big\}$
$\mathcal{C}:$	$M_{\mathcal{C}} = HMAC_{SK^{\mathcal{C}}}\Big(RREP, \mathcal{D}, \mathcal{S}, (\mathcal{C}, \mathcal{N}, \mathcal{A}), (Auth^{\mathcal{D}}, Auth^{\mathcal{C}}), (M_{\mathcal{D}}), Auth^{\mathcal{DS}}_{Data}\Big)$
$\mathcal{C} \rightarrow \mathcal{N}:$	$\Big\{RREP, \mathcal{D}, \mathcal{S}, (\mathcal{C}, \mathcal{N}, \mathcal{A}), (Auth^{\mathcal{D}}, Auth^{\mathcal{C}}), (M_{\mathcal{D}}, M_{\mathcal{C}}), Auth^{\mathcal{DS}}_{Data}\Big\}$
$\mathcal{N}:$	$M_{\mathcal{N}} = HMAC_{SK^{\mathcal{N}}}\Big(RREP, \mathcal{D}, \mathcal{S}, (\mathcal{C}, \mathcal{N}, \mathcal{A}), (Auth^{\mathcal{D}}, Auth^{\mathcal{C}}, Auth^{\mathcal{N}}), (M_{\mathcal{D}}, M_{\mathcal{C}}), Auth^{\mathcal{DS}}_{Data}\Big)$
$\mathcal{N} \rightarrow \mathcal{A}:$	$\Big\{RREP, \mathcal{D}, \mathcal{S}, (\mathcal{C}, \mathcal{N}, \mathcal{A}), (Auth^{\mathcal{D}}, Auth^{\mathcal{C}}, Auth^{\mathcal{N}}), (M_{\mathcal{D}}, M_{\mathcal{C}}, M_{\mathcal{N}}), Auth^{\mathcal{DS}}_{Data}\Big\}$
$\mathcal{A}:$	$M_{\mathcal{A}} = HMAC_{SK^{\mathcal{A}}}\Big(RREP, \mathcal{D}, \mathcal{S}, (\mathcal{C}, \mathcal{N}, \mathcal{A}), (Auth^{\mathcal{D}}, Auth^{\mathcal{C}}, Auth^{\mathcal{N}}, Auth^{\mathcal{A}}), (M_{\mathcal{D}}, M_{\mathcal{C}}, M_{\mathcal{N}}), Auth^{\mathcal{DS}}_{Data}\Big)$
$\mathcal{A} \rightarrow \mathcal{S}:$	$\Big\{RREP, \mathcal{D}, \mathcal{S}, (\mathcal{C}, \mathcal{N}, \mathcal{A}), (Auth^{\mathcal{D}}, Auth^{\mathcal{C}}, Auth^{\mathcal{N}}, Auth^{\mathcal{A}}), (M_{\mathcal{D}}, M_{\mathcal{C}}, M_{\mathcal{N}}, M_{\mathcal{A}}), Auth^{\mathcal{DS}}_{Data}\Big\}$

Table 7.2: Route Reply Unicast in Scasec.

7.4.2 Route Reply (RREP) Propagation in Scasec

Analogous to the RREQ, the destination $\mathcal{D}$ appends $Auth^{\mathcal{D}}$ and $M_{\mathcal{D}}$ to the authenticator and the HMAC lists, respectively. This enables the intermediate nodes ($\mathcal{C}$, $\mathcal{N}$ and $\mathcal{A}$) to authenticate $\mathcal{D}$ and to verify the integrity of the RREP. Similarly, the intermediate nodes ($\mathcal{C}$, $\mathcal{N}$ and $\mathcal{A}$) append their authenticators ($Auth^{\mathcal{C}}$, $Auth^{\mathcal{N}}$, and $Auth^{\mathcal{A}}$) and the HMACs ($M_{\mathcal{C}}$, $M_{\mathcal{N}}$, and $M_{\mathcal{A}}$) to the RREP for the purpose authentication and integrity verification. On the successful authentication and verification of the RREP, all downstream nodes that follow $\mathcal{D}$ record the route in their route cache. Although the RREP is analogous to the RREQ, it differs from the RREQ by traversing a defined path and carrying an encrypted session key $SK^{\mathcal{DS}}_{Data}$ in $Auth^{\mathcal{DS}}_{Data}$ to facilitate the confidentiality for the subsequent data flow. As we will explain later, $\mathcal{D}$'s $SK^{\mathcal{DS}}_{Data}$ will only be shared with $\mathcal{S}$ to render a confidentiality service for the data flow.

An alternate design for the above approach is to enable $\mathcal{D}$ and all intermediate nodes ($\mathcal{C}$, $\mathcal{N}$, and $\mathcal{A}$) to generate an authenticator that can only be authenticated by them and $\mathcal{S}$. In other words, they can be made to generate $Auth^{\mathcal{J}}$ for RREP, such that

$S^{\mathcal{J}}_{\mathcal{I}1} = 1$ in matrix *S($\mathcal{I}$)*, where $\mathcal{I}, \mathcal{J} \in \{\mathcal{D}, \mathcal{C}, \mathcal{N}, \mathcal{A}, \mathcal{S}\}$ and $\mathcal{I} \neq \mathcal{J}$. Although this design provides a closed approach that allows only the mobile nodes that had participated in the RREQ phase to confirm their participation in the RREP phase, it does prevent the mobile nodes from *snooping* valid routes in their environment. Given that the design of the key management and secure routing sub-systems is not the primary focus of our thesis, we leave this design alternative to the implementor's choice.

7.4.3 Secure Data Flow using Scasec

Source $\mathcal{S}$ chooses a route that is shorter in length if multiple routes have been discovered. As we will present in the next §7.5, the same situation would warrant a feedback from the SMRTI and therefore a higher trustworthy route would be chosen among the available routes. Recall from §7.3.2 that the SMG permits a mobile node to choose a list of other mobile nodes with which it can establish a session or a secret key. Interestingly, at the extremity of the SMG, for example, a mobile node $\mathcal{D}$ can be made to generate a secret key $SK^{\mathcal{DS}}_{Data}$ that can only be decrypted by another mobile node $\mathcal{S}$. In other words, $\mathcal{D}$ can be made to generate $Auth^{\mathcal{DS}}_{Data}$ such that $S^{\mathcal{D}}_{\mathcal{I}1} = 0$ for all $\mathcal{I}$, where $\mathcal{I}$ satisfies the following: $\{(1 \leqslant \mathcal{I} \leqslant n) \wedge (\mathcal{I} \neq \mathcal{D}) \wedge (\mathcal{I} \neq \mathcal{S})\}$. Therefore, once $\mathcal{S}$ accepts the RREP, it extracts $SK^{\mathcal{DS}}_{Data}$ from $Auth^{\mathcal{DS}}_{Data}$ which is only shared with $\mathcal{D}$. Node $\mathcal{S}$ later uses $SK^{\mathcal{DS}}_{Data}$ to secure the data flow from eavesdropping.

Given that the parameters can be pre-loaded to the mobile nodes in a managed MANET, each mobile node $\mathcal{I}$ can be prevented from computing $Auth^{\mathcal{IJ}}_{Data}$ for every other mobile node $\mathcal{J}$ in the network. Otherwise, each mobile node $\mathcal{I}$ can be pre-loaded with $Auth^{\mathcal{IJ}}_{Data}$ for every other mobile node $\mathcal{J}$, where $\mathcal{I}$ and $\mathcal{J}$ satisfy the following: $\{(1 \leqslant \mathcal{I} \leqslant n) \wedge (1 \leqslant \mathcal{J} \leqslant n) \wedge (\mathcal{I} \neq \mathcal{J})\}$. Furthermore, these parameters can be represented in a separate diagonal matrix (*data keying matrix K($\mathcal{I}$)*) instead of superimposing them as another column in the communication group matrix *S($\mathcal{I}$)*. The resultant data keying matrix *K($\mathcal{I}$)* of the order $n * l$ can be visualised as:

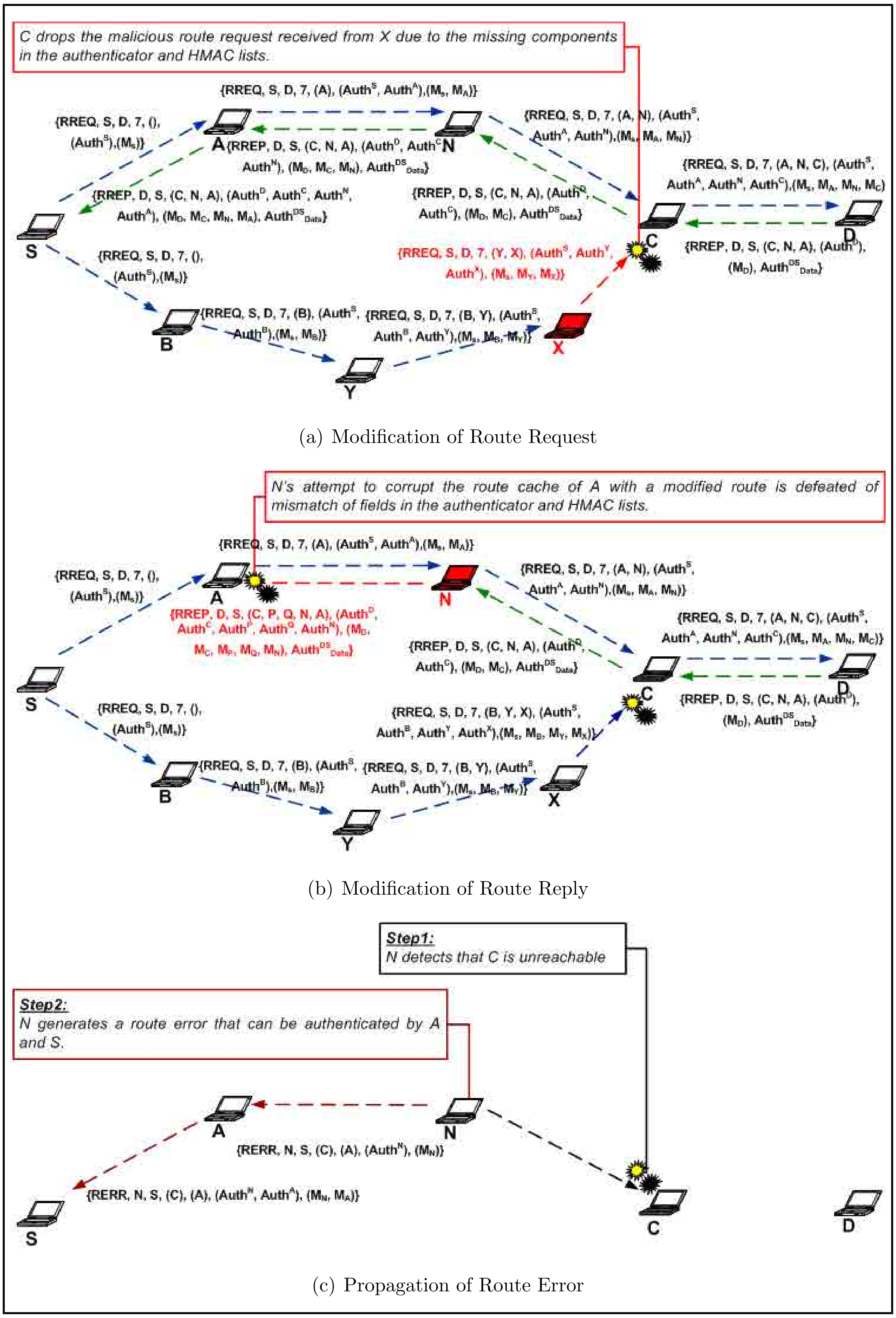
$$K(\mathcal{I}) = \begin{pmatrix} S^{\mathcal{I}}_{11} = 1 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & S^{\mathcal{I}}_{\mathcal{M}k} = 1 & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & S^{\mathcal{I}}_{\mathcal{N}l} = 1 \end{pmatrix}$$

7.4.4 Discussion

It is evident from the above operation that the Scasec defends against both the modification and fabrication attacks. As shown in Figure 7.2(a), the route that is modified by the malicious node $\mathcal{X}$ becomes unravelled because of the missing components in the authenticator and the HMAC lists. Consequently, the malicious RREQ gets discarded at subsequent hop. Alternatively, if $\mathcal{X}$ attempts to insert a fabricated RREQ with an increased *id* that contains a shorter route with the aim to disrupting the route discovery, then the *id* field of the RREQ comes into play to prevent such fabrication attacks. As seen in Figure 7.2(b), the modification becomes much more difficult in the case of the RREP owing to the unicast nature.

Although not discussed earlier, a route error (RERR) is triggered by a mobile node to a source node whenever it learns that it has a broken link with its next-hop. The format of an RERR is, ⟨*RERR tag, initiator, target, unreachable node, intermediate node list, authenticator, HMAC*⟩. Let us consider the scenario in Figure 7.2(c) where node $\mathcal{N}$ reports to $\mathcal{S}$ that $\mathcal{C}$ is unreachable. As the RERR traverses the upstream nodes along the path $\mathcal{N} \mapsto \mathcal{A} \mapsto \mathcal{S}$, then each upstream node (*i.e.* $\mathcal{A}$ and $\mathcal{S}$) authenticates the reporting node $\mathcal{N}$ via $Auth^{\mathcal{N}}$ and verifies the RERR using $M_{\mathcal{N}}$. Recall that $Auth^{\mathcal{N}}$ and $M_{\mathcal{N}}$ protect the RERR against the modification of the RERR.

In summary, it is apparent that the key management and secure routing sub-systems successfully defend against well-known attacks such as the modification and fabrication of the network layer packets by means of an authentication and integrity verification. However, if the previous scenario is closely observed, then it is obvious that the upstream nodes (including the source) are incapable of verifying the credibility


C drops the malicious route request received from X due to the missing components in the authenticator and HMAC lists.

N's attempt to corrupt the route cache of A with a modified route is defeated of mismatch of fields in the authenticator and HMAC lists.

Step1:
N detects that C is unreachable

Step2:
N generates a route error that can be authenticated by A and S.

S
A
N
C
D
B
Y
X
{RREQ, S, D, 7, (), (Auth^S),(M_s)}
{RREQ, S, D, 7, (A), (Auth^S, Auth^A),(M_s, M_A)}
{RREQ, S, D, 7, (A, N), (Auth^S, Auth^A, Auth^N),(M_s, M_A, M_N)}
{RREQ, S, D, 7, (A, N, C), (Auth^S, Auth^A, Auth^N, Auth^C),(M_s, M_A, M_N, M_C)}
{RREP, D, S, (C, N, A), (Auth^D), (M_D), Auth^DS_Data}
{RREP, D, S, (C, N, A), (Auth^D, Auth^C), (M_D, M_C), Auth^DS_Data}
{RREP, D, S, (C, N, A), (Auth^D, Auth^C, Auth^N), (M_D, M_C, M_N), Auth^DS_Data}
{RREP, D, S, (C, N, A), (Auth^D, Auth^C, Auth^N, Auth^A), (M_D, M_C, M_N, M_A), Auth^DS_Data}
{RREQ, S, D, 7, (B), (Auth^S, Auth^B),(M_s, M_B)}
{RREQ, S, D, 7, (B, Y), (Auth^S, Auth^B, Auth^Y),(M_s, M_B, M_Y)}
{RREQ, S, D, 7, (Y, X), (Auth^S, Auth^Y, Auth^X), (M_s, M_Y, M_X)}
{RREP, D, S, (C, P, Q, N, A), (Auth^D, Auth^C, Auth^P, Auth^Q, Auth^N), (M_D, M_C, M_P, M_Q, M_N), Auth^DS_Data}
{RREQ, S, D, 7, (B, Y, X), (Auth^S, Auth^B, Auth^Y, Auth^X),(M_s, M_B, M_Y, M_X)}
{RERR, N, S, (C), (A), (Auth^N), (M_N)}
{RERR, N, S, (C), (A), (Auth^N, Auth^A), (M_N, M_A)}

(a) Modification of Route Request

(b) Modification of Route Reply

(c) Propagation of Route Error

Figure 7.2: Scasec Defence against Route Discovery related Attacks.

of an RERR. Furthermore, they are susceptible to being continuously exposed to the behaviours of the malicious nodes, regardless that they have already learnt of the presence of those malicious nodes. This is the key factor that warrants a *detective-reactive* approach to render a *second-layer of defence*. In the following §7.5, we will look into the interface integration of preventive and detective-reactive sub-systems with the objective of realising a *unified two-layer trust enhanced security architecture for the MANET*.

7.5 TEAM

In this section, we present our TEAM where the SMG and the Scasec are integrated with the fellowship and the SMRTI to make trust-enhanced routing decisions. Analogous to Chapter 6, the TEAM concentrates on the interface integration, where the Scasec is made to provide evidence to the SMRTI and, in turn, together with the SMG, it is also made to take advantage of the SMRTI's feedback to make trust-enhanced decisions. Although the interface integration is detailed from the perspective of the SMG and the Scasec, the description applies to any indentity-based or offline-CA based key management schemes and secure routing protocols. Note that our thesis primarily focuses on the design of a trust-enhanced secure framework; while the design of the key management and secure routing sub-systems are supplementary components in our design. Given that extensive research has been conducted in the area of key management and that it is not the only focus in our thesis, we have inherited the well-reviewed proposals for pure MANETs and adapted a well-studied SMG for the managed MANETs.

7.5.1 Interface Integration

Rather than visualising the Scasec, the SMG and the DSR as separate sub-systems, we can consider them as an amalgamated hybrid sub-system for the purpose of interface integration because of their tight coupling and dependency on each other.

Initially, the main integration objective is to facilitate the Scasec to *asynchronously* report benign and malicious behaviours to the SMRTI's detection component. As

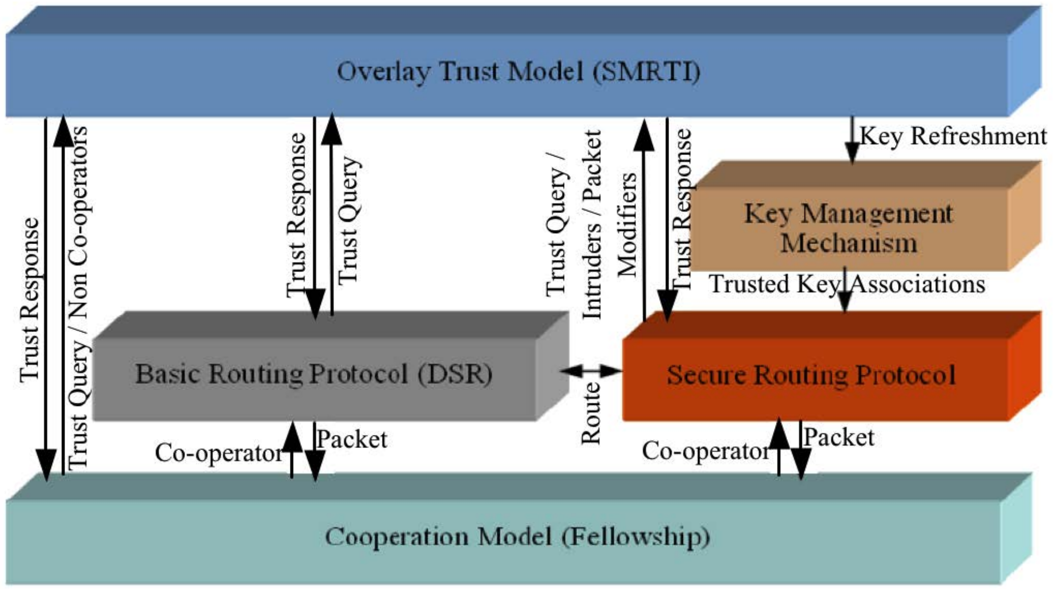


Figure 7.3: Trust Enhanced Secure Architecture for MANET.

depicted in Figure 7.3, the Scasec is integrated with the fellowship so that it processes incoming packets after they have been sanitised by the rate-limitation component, and dispatches the packets for transmission once the enforcement component has given the *go ahead* signal. The Scasec authenticates those incoming packets and verifies their integrity using the SMG's parameters. Successful authentication and verification enable the Scasec to report the result to the SMRTI's detection component as evidence for the previous-hop's benign behaviour (*pos(packet reception)*), which is then aggregated into the *accrued direct reputation* held for the previous-hop (4.4). Alternatively, the failure to authenticate and verify the integrity of the received packets causes the Scasec to report the result to the SMRTI as evidence for the previous-hop's malicious behaviour (*neg(packet reception, modification/fabrication)*). Regardless of whether the packet is modified or injected by any upstream malicious node (other than the previous-hop), the previous-hop is held responsible for such modification or injection because it was bound to authenticate and verify the integrity of all the packets before transmitting them. Therefore, if an incoming packet is discovered to be maliciously modified or injected, then the previous-hop is held accountable for either exhibiting such behaviour or colluding with such malicious nodes. Furthermore, the Scasec is made to process packets that have been promiscuously captured by the enforcement component via the IDS module to determine whether the next-hop has forwarded those packets without any malicious modification. Recall that the fellowship's enforcement component can only determine whether the next-hop has forwarded packets on its behalf, but not whether those forwarded packets have been maliciously modified or not.

Liaising with the DSR, the Scasec *synchronously* queries the SMRTI's reaction component to determine the, (a)*trustworthiness of received packet (*i.e. $\mathcal{S}$ *and* $\mathcal{D}$*)*, (b) *trustworthiness of the discovered or snooped route* and, (c) *highest trustworthy route among multiple the available routes.*

Similarly, the SMG liaises with the DSR to query the SMRTI's reaction component for the purpose of identifying the trustworthiness of the other mobile nodes. Depending on the results received from the SMRTI, those mobile nodes are then catered for the appropriate sensitivity level k and, accordingly as intermediate nodes into a communication flow that belong to a communication group ($CG_k^{\mathcal{I}}$), where $1 \leqslant k \leqslant l$. In summary, the SMG performs only *synchronous* communications with the SMRTI for the purpose of regenerating $Auth_k^{\mathcal{I}}$ for a communication flow that has the sensitivity level k by enabling or disabling $S_{\mathcal{J}k}^{\mathcal{I}}$ in its matrix $S(\mathcal{I})$ depending on the trustworthiness held for every other mobile node $\mathcal{J}$.

7.5.2 System Operation

Let us now explore the system operation between the integrated sub-systems (SMRTI, fellowship, SMG, and Scasec) and the DSR. Recall that the Scasec and the SMG are used to discover secure paths and therefore to provide secure communications, while the fellowship is used to defend against both flooding and packet drop attacks. The TEAM integrates these sub-systems through the SMRTI, which *asynchronously* collects evidence of the trustworthiness for the other mobile nodes from these sub-systems (Scasec, SMG and fellowship) and synchronously responds to their requests. This factor, in turn, enables them to react accordingly to the behavioural changes of those mobile nodes. Let us consider the path $\mathcal{S} \mapsto \mathcal{A} \mapsto \mathcal{N} \mapsto \mathcal{C} \mapsto \mathcal{D}$ from Figure 7.1, in which $\mathcal{S}$ is the source and $\mathcal{D}$ is the destination for the communication flow. Nodes

$\mathcal{A}$, $\mathcal{N}$ and $\mathcal{C}$ are the intermediaries that form the path from $\mathcal{S}$ to $\mathcal{D}$.

Node $\mathcal{S}$ initially checks its route cache for a route to $\mathcal{D}$ whenever it wishes to send a data packet to $\mathcal{D}$. On finding one or more routes, the DSR passes the route(s) to the SMRTI to evaluate their trustworthiness. Remember that the trust evaluation for a route is performed by evaluating the trustworthiness of all nodes that are embedded in the route, except for the evaluating node (*i.e.* $\mathcal{S}$). The route with highest level of trustworthiness is chosen for the communication flow and passed back to the DSR, provided there is more than one trustworthy route for $\mathcal{D}$. The DSR then requests the Scasec to incorporate the necessary security services through its cryptographic suite and shared key from the SMG. Alternatively, if all routes are untrustworthy or there is no available route to D, then the SMRTI responds back to the DSR with a decision to initiate a new route discovery to $\mathcal{D}$.

Let us now consider the role of the TEAM in an event such as a route request, route reply, route error or data flow at one of the intermediate nodes ($\mathcal{A}$, $\mathcal{N}$ and $\mathcal{C}$). To begin with, the rate-limitation component of the fellowship sub-system at every intermediate node queries the SMRTI to learn the trustworthiness of the previous-hop. This is in accordance with the expectation that the incoming packets are not exposed to malicious actions only if the previous-hop is trustworthy. As long as the previous-hop is assured to be trustworthy and the rate of the incoming packets from the previous-hop is within τ, the rate-limitation passes those packets to the DSR. Alternatively, if the previous-hop has moved into an untrustworthy zone, then the incoming packets from the previous-hop are discarded as detailed in §3.3.4.1. However, if the previous-hop is deemed trustworthy but seen to flood packets that exceed τ, then those packets that exceed τ are discarded for ρ intervals and then all packets after ρ^{th} interval for the same reason. All these actions are reported to the SMRTI's detection component as *direct evidence* towards the previous-hop's direct reputation.

Once packets from a cooperating and trustworthy previous-hop reache the DSR, it transfers them to the Scasec, which then authenticates the list of the nodes traversed by those packets and their integrity. On finding evidence of any modification or fabrication attack, the Scasec then reports the evidence for the malicious behaviour on the previous-hop's direct reputation. Alternatively, the packets that are deemed as free from modification and fabrication attacks are evaluated for their trustworthiness. In other words, the Scasec moves the authenticated packets to the DSR, which then queries the SMRTI's reaction component to evaluate the trustworthiness of those packets. This is because the intermediate nodes ($\mathcal{A}$, $\mathcal{N}$, and $\mathcal{C}$) forward packets only for the sake of $\mathcal{S}$ and $\mathcal{D}$ and, furthermore, the trustworthiness held for $\mathcal{S}$ and $\mathcal{D}$ could have changed either way because it is non-monotonic. In addition, the SMRTI is requested to evaluate the trustworthiness of the route contained in the packet, provided it is a route reply event that confirms the activeness of the route. In the instance where the route evaluates to trustworthiness, the DSR of the intermediate nodes ($\mathcal{A}$, $\mathcal{N}$, and $\mathcal{C}$) records the route in its route cache. Once the DSR confirms via the SMRTI that the packet is trustable, it passes the packet to the fellowship's enforcement component instead of calling in the Scasec to append the necessary security fields to the packet as a part of the preparation for transmission. Although the Scasec can be designed to compute the necessary authenticator and HMAC fields before moving the packet to the fellowship's enforcement, such a design is discouraged because the Scasec's computation can go void if the fellowship's enforcement component learns from the SMRTI that the next-hop is untrustworthy.

At the intermediate nodes ($\mathcal{A}$, $\mathcal{N}$, and $\mathcal{C}$), the fellowship's enforcement component polls the SMRTI's reaction component to determine the trustworthiness of the next-hop. This is to meet the expectation that the packet will reach $\mathcal{D}$ without being exposed to malicious behaviour, but only if the next-hop is trustworthy. However, this step is not applicable for a route request event owing to the broadcast nature of the route request. On confirmation from the enforcement component that the next-hop is trustable, the DSR then calls for the Scasec to append the necessary security fields to the packet before moving the packet to the fellowship's restoration component. Finally, the restoration component takes the availability of the wireless medium into account to tune the rate at which the packets can be transmitted to the next-hop or received from the previous-hop, where the tuning is proportional to the trustworthiness held for the next-hop and the previous-hop, respectively.

Orthogonally, the enforcement component also collects the evidence for the next-hop's cooperation with the help of an IDS-based monitor, that is, whether the next-hop has forwarded the packet on its behalf in the future. In the instance where the next-hop fails to show a sign of cooperation, the enforcement component is tailored to check for a list of items before deeming the next-hop as a packet dropper. Those items include (a) *the contention rate for the wireless medium*, (b) *the γ for next-hop*, (c) *the rate at which requests are transmitted to the next-hop* and, (d) *whether the next-hop forwards the remaining packets (that are destined to a different downstream node) on its behalf*. As mentioned in Chapter 3, if the next-hop fails to convince that the packet was not dropped deliberately, the enforcement component then communicates the *direct evidence* for next-hop's packet dropping behaviour to the SMRTI's detection component. Note that this communication of the evidence takes place for every packet that is confirmed to be dropped by the next-hop. Nevertheless, if the next-hop forwards the packet without performing any malicious behaviour then the enforcement component forwards the *direct evidence* for the next-hop's benign behaviour to the SMRTI's detection component. Note that fellowship's enforcement component liaises with the Scasec to determine whether the next-hop has forwarded the packet on its behalf without any malicious modification.

Once packets reach it, $\mathcal{D}$ conducts similar investigations, such as whether, (a) *the incoming packets exceed τ*, (b) *the incoming packets are free from modification and fabrication attacks* and, (c) *the previous-hop and $\mathcal{S}$ are trustworthy* using its subsystems (the fellowship, the SMRTI, the Scasec and the SMG). The sequence of system operations explained thus far holds valid for networks that are exposed to a high density of malicious nodes. Optionally, a few evaluations can be turned off depending on the degree of the maliciousness expected in the environment.

7.6 Conclusion

In this chapter, we have successfully put forth the notion of adopting an offline CA-based key management scheme for the pure MANETs and consequently, the feasibility of building up a secure routing protocol such as the BISS based upon the certificates provided by the offline-CA. We have discussed how such a combined scheme not only incorporates the self-organised characteristics of the pure MANETs but also takes advantage of the inherent mobility to expand the secure associations among the mobile nodes and to establish secure communications. We considered these solutions for the prevention layer because they meet the requirements of the pure MANETs by taking advantage of the inherent issues that otherwise disadvantage the possibility of such solutions.

In the case of the managed MANETs, we adapted the SMG as the key management scheme instead of inheriting the readily-available symmetric or asymmetric-based key management schemes. Notably, the offline CA-based key management scheme that was recommended for the pure MANETs can well be successfully tailored for the managed MANETs. However, all these key management schemes failed to render at least any one of the following features, (a) *the broadcast authentication*, (b) *the one-to-one shared secret association*, (c) *the secured group communications*, (d) *scalability* or, (e) *the multi-service, such as communications based on different sensitivity levels*. Later, we discussed how the SMG operates in the context of mobile nodes. SMG implicitly addresses *sybil attacks*, where a malicious node takes multiple identities to subvert the reputation system. Recall that even in the absence of SMG or any key management system, SMRTI effectively defends against sybil attacks because of its design to exclude the opinions of other nodes and only to make subjective decisions. In other words, SMRTI honours the identity that exhibits benign behaviours (and excludes the identity that is associated with malicious behaviours) in the presence of sybil attacks.

We then presented the Scasec as the secure routing protocol for the managed MANETs and then its capability for discovering secure paths between the source and destination nodes. We also demonstrated the strength of the Scasec in the presence of modification and fabrication attacks and, in addition, its ability to take advantage of the SMG for establishing end-to-end secure communications to defend against eavesdroppers. However, if the network is prone to wormhole attacks, then proposal, such as, packet leashes [162] can be integrated.

Finally, we introduced our TEAM and discussed the interface integrations between the preventive and detective-reactive sub-systems. We focused primarily at the level of enabling preventive the sub-systems from taking advantage of the SMRTI's feedback so that they could make trust-enhanced decisions. Alternatively, those preventive sub-systems were designed to complement the SMRTI by feeding in evidence for both the benign and the malicious behaviours of the other mobile nodes. All these rendered a tight-coupling among the sub-systems. At the end, we demonstrated the success of our *unified dual-layer trust-enhanced architecture* in terms of its system operation. Note that our resultant dual-layer architecture is, indeed, a framework that portrays the feasibility of integrating the various preventive and detective-reactive approach based sub-systems so that not only can the prevailing attacks be defended, but also the emerging vulnerabilities can be resolved. In summary, the TEAM realistically integrates the various preventive and detective-reactive based sub-systems to arrive at the envisaged unified two-layer trust enhanced architecture and also provides the platform for the modular integration of the futuristic preventive and detective-reactive sub-systems for the purpose of defending against emerging attacks.

8 Mitigating Flooding Attacks in Anonymous Communications

8.1 Introduction

A few mission critical applications such as battlefield communications require a high degree of anonymous communication. Several techniques [330] have been proposed to achieve anonymous communication in computer networks. Recently, there is an increasing interest in developing protocols that can achieve a high degree of anonymity for communications in the MANET [9, 59–61, 176, 177, 199, 200, 348, 352, 374, 450, 460]. The main aim of anonymous communication is to achieve a high resistance to eavesdropping and traffic analysis. In general, most of the proposed techniques are based on public key cryptography and/or based on Chaum's [92, 93] *mix-net* technique. The principle behind Chaums mix-net technique is that the traffic from a source to a destination has to pass through one or more *mixes*, while a mix relays data from different end-to-end connections. However, unlike the traditional routers, a mix reorders and re-encrypts the incoming packets in such a way that the corresponding outgoing packets are unrelated. This is to thwart the attempts of an attacker to follow or map an end-to-end connection.

Although a reasonable degree of anonymity can be achieved in wired networks, there are several additional challenges for achieving anonymity in the MANET. These are due to the nature of the communication in a wireless medium and the limited computational capability of the mobile nodes. Furthermore, a mix-net should be capable of providing anonymity even if some of the mixes are compromised. The main reason for this requirement is that in a hostile environment such as a battlefield, there is a greater probability for the roaming nodes or mixes to be captured by enemies. Hence, the traffic has to be passed through a greater number of mixes to lower the probability of compromising the anonymity. On the other hand, relaying data traffic through too many mixes can increase the average latency and decrease the average data delivery ratio. This would also consume greater resources in the mobile nodes. Therefore, it is necessary to achieve a balance between the high degree of anonymity and low latency.

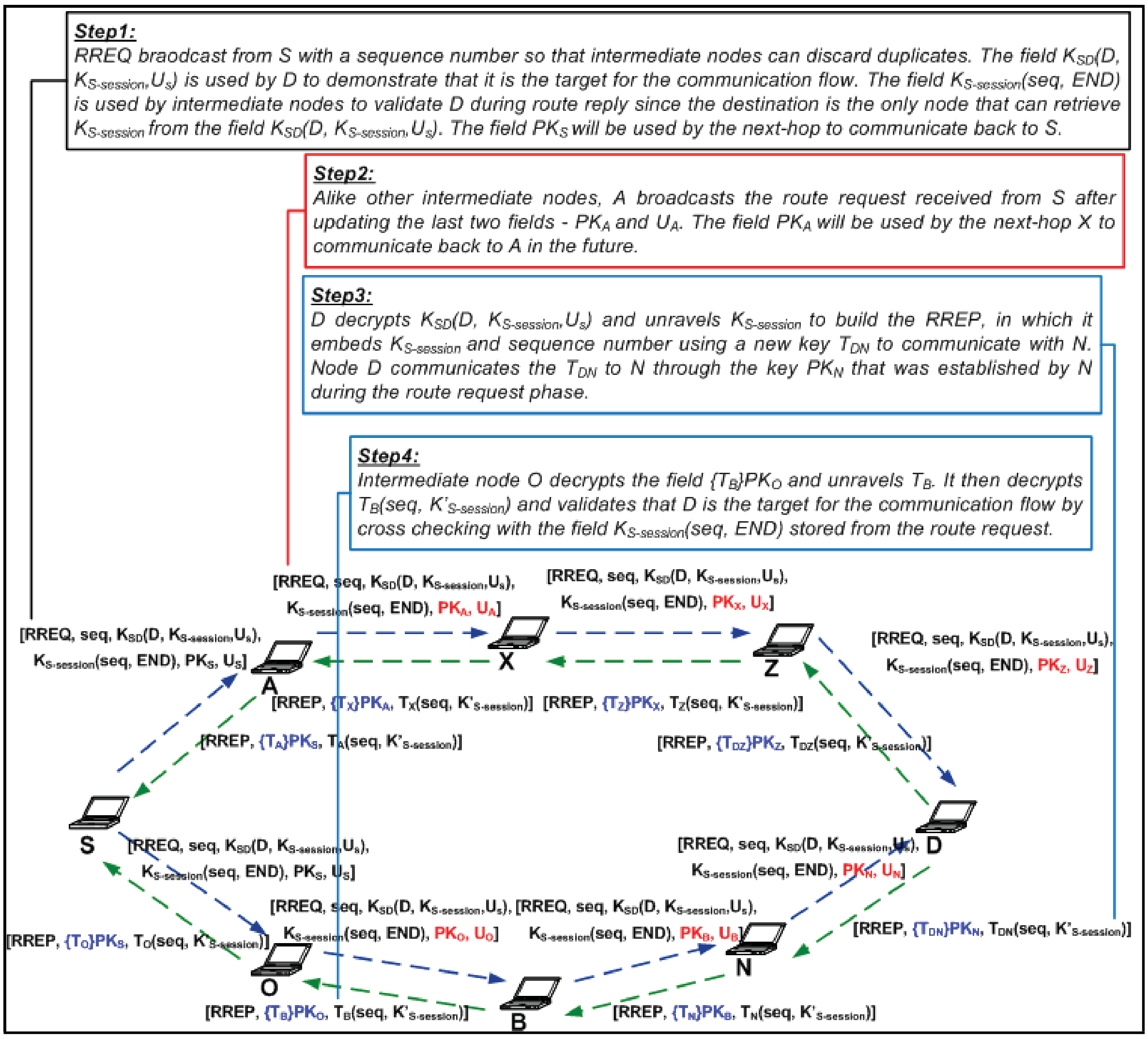
Some of the recently proposed protocols [9, 59–61, 176, 177, 199, 200, 348, 352, 450] are successful in achieving a different degree of anonymity with a reasonable level of latency. However, there are some significant disadvantages associated with these approaches. First, they are inefficient in terms of performance. Second, they make it extremely straightforward for an attacker to perform different types of attacks, such as flooding and packet drop attacks, within the network. Currently available security tools [11, 12, 369, 370, 388, 389] that can detect these abnormalities cannot be used in the case of anonymous communication. We will discuss these issues in more detail in §8.3.

In this chapter, we will consider the ***Anonymous Secure Routing (ASR)*** protocol [460] and analyse how an attacker can severely degrade the performance of the network by performing flooding and packet drop attacks. We will then demonstrate the adaptation of the fellowship model (Chapter 3) to mitigate the flooding attacks in the MANET that supports anonymous communications. The main reason for opting for the ASR among the anonymous protocols is that it is simple, light in weight, provides a higher degree of identity anonymity, location privacy and route anonymity, while ensuring the security of the established routes in the MANET. Although we analyse packet drop attacks in this chapter, we concentrate on mitigating only the flooding attacks. The rationale behind dealing with only the flooding attacks and not the packet drop attacks is that it is hard for an observer to relate an ingoing packet with an outgoing packet from an observed node. Note that correlating an ingoing packet with an outgoing packet is a requirement for detecting packet drop attacks that, on other hand, contradicts the requirements of the anonymity service.

This chapter is organised as follows. In §8.2, we present an overview of the ASR protocol. In §8.3, we explain on how an attacker can perform flooding and packet drop attacks and why it is difficult to deal with these attacks. In §8.4, we present our adaptation of the fellowship model to deal with flooding attacks in the MANET that supports anonymous communications. Section 8.5 then analyses the performance of the adapted fellowship model through the NS2 simulations. Finally, §8.6 provides the concluding remarks.

8.2 Anonymous Secure Routing (ASR) Protocol

The route establishment process in the ASR protocol is similar to the route establishment process in the DSR protocol. The main difference between the ASR and the DSR protocols is that the route request in the ASR contains neither an initiator's address nor a target's address. In addition, the intermediary nodes do not append their identity to the route request. In the following section, we confine how the multi-hop anonymous routes are established between a source (initiator) and destination (target) nodes. For more detail on data transmissions, route maintenance and the analysis of the degree of anonymity, we encourage the reviewer to refer to [460].



Step1:
RREQ broadcast from S with a sequence number so that intermediate nodes can discard duplicates. The field $K_{SD}(D, K_{S\text{-}session}, U_S)$ is used by D to demonstrate that it is the target for the communication flow. The field $K_{S\text{-}session}(seq, END)$ is used by intermediate nodes to validate D during route reply since the destination is the only node that can retrieve $K_{S\text{-}session}$ from the field $K_{SD}(D, K_{S\text{-}session}, U_S)$. The field PK_S will be used by the next-hop to communicate back to S.

Step2:
Alike other intermediate nodes, A broadcasts the route request received from S after updating the last two fields - PK_A and U_A. The field PK_A will be used by the next-hop X to communicate back to A in the future.

Step3:
D decrypts $K_{SD}(D, K_{S\text{-}session}, U_S)$ and unravels $K_{S\text{-}session}$ to build the RREP, in which it embeds $K_{S\text{-}session}$ and sequence number using a new key T_{DN} to communicate with N. Node D communicates the T_{DN} to N through the key PK_N that was established by N during the route request phase.

Step4:
Intermediate node O decrypts the field $\{T_B\}PK_O$ and unravels T_B. It then decrypts $T_B(seq, K'_{S\text{-}session})$ and validates that D is the target for the communication flow by cross checking with the field $K_{S\text{-}session}(seq, END)$ stored from the route request.

[RREQ, seq, $K_{SD}(D, K_{S\text{-}session}, U_s)$, $K_{S\text{-}session}(seq, END)$, PK_S, U_S]
[RREQ, seq, $K_{SD}(D, K_{S\text{-}session}, U_s)$, $K_{S\text{-}session}(seq, END)$, PK_A, U_A]
[RREQ, seq, $K_{SD}(D, K_{S\text{-}session}, U_s)$, $K_{S\text{-}session}(seq, END)$, PK_X, U_X]
[RREQ, seq, $K_{SD}(D, K_{S\text{-}session}, U_s)$, $K_{S\text{-}session}(seq, END)$, PK_Z, U_Z]
A
X
Z
[RREP, $\{T_X\}PK_A$, $T_X(seq, K'_{S\text{-}session})$]
[RREP, $\{T_Z\}PK_X$, $T_Z(seq, K'_{S\text{-}session})$]
[RREP, $\{T_A\}PK_S$, $T_A(seq, K'_{S\text{-}session})$]
[RREP, $\{T_{DZ}\}PK_Z$, $T_{DZ}(seq, K'_{S\text{-}session})$]
S
[RREQ, seq, $K_{SD}(D, K_{S\text{-}session}, U_s)$, $K_{S\text{-}session}(seq, END)$, PK_S, U_S]
[RREQ, seq, $K_{SD}(D, K_{S\text{-}session}, U_s)$, $K_{S\text{-}session}(seq, END)$, PK_N, U_N]
D
[RREQ, seq, $K_{SD}(D, K_{S\text{-}session}, U_s)$, $K_{S\text{-}session}(seq, END)$, PK_O, U_O]
[RREQ, seq, $K_{SD}(D, K_{S\text{-}session}, U_s)$, $K_{S\text{-}session}(seq, END)$, PK_B, U_B]
[RREP, $\{T_O\}PK_S$, $T_O(seq, K'_{S\text{-}session})$]
[RREP, $\{T_{DN}\}PK_N$, $T_{DN}(seq, K'_{S\text{-}session})$]
N
O
B
[RREP, $\{T_B\}PK_O$, $T_B(seq, K'_{S\text{-}session})$]
[RREP, $\{T_N\}PK_B$, $T_N(seq, K'_{S\text{-}session})$]

Figure 8.1: Route Discovery Cycle in ASR Protocol.

The authors of the ASR assume that a shared secret exists between the source and the destination nodes. Let us consider on how anonymous multi-hop routes are established between the source $\mathcal{S}$ and the destination $\mathcal{D}$ through intermediate nodes. The following presents the notations used in ASR:

$[M]$	represents a packet
$K(M)$	represents the encryption of M by a secret key K
$\{M\}_{PK}$	represents the encryption of M by a public key PK

The format of the route request is listed below:

$$[RREQ, seq, K_{\mathcal{SD}}(\mathcal{D}, K_{\mathcal{S}-session}, U_{\mathcal{S}}), K_{\mathcal{S}-session}(seq, END), PK_{\mathcal{I}-1}, U_{\mathcal{I}-1}]$$

In the above, the *RREQ* denotes the route request, *seq* refers to the sequence number, $K_{\mathcal{SD}}$ is the shared secret between the source $\mathcal{S}$ and the destination $\mathcal{D}$, $K_{\mathcal{S}-session}$ is the session key to be used for the following data flow, *END* is included for a later confirmation that the destination has received the route request, $PK_{\mathcal{I}-1}$ is the one-time public key and $U_{\mathcal{I}-1}$ is a random number generated by the previous-hop node $node_{\mathcal{I}-1}$. Similarly, $PK_{\mathcal{S}}$ is the one-time public key and $U_{\mathcal{S}}$ is a random number chosen by source $\mathcal{S}$. As we will explain later, $PK_{\mathcal{S}}$ will be used by the next-hop to encrypt the route reply, so that $\mathcal{S}$ can decrypt the received route reply accordingly.

On receiving the route request, each intermediate node $\mathcal{I}$ checks for the *seq*. If the *seq* is already recorded, then the route request is dropped and no action is initiated. Alternatively, if the *seq* is the latest one in the sequence, then $\mathcal{I}$ records the *seq* and attempts to decrypt the third element of the route request, $K_{\mathcal{SD}}(\mathcal{D}, K_{\mathcal{S}-session}, U_{\mathcal{S}})$. Assuming that the decryption is successful, $\mathcal{I}$ then concludes that the route request is destined for it. The route request is further processed to extract the $U_{\mathcal{S}}$ and the validation of the maximum number of hops. On the unsuccessful decryption of the element $K_{\mathcal{SD}}(\mathcal{D}, K_{\mathcal{S}-session}, U_{\mathcal{S}})$, $\mathcal{I}$ records the *seq*, $PK_{\mathcal{I}-1}$ and $K_{\mathcal{S}-session}(seq, END)$ into its route table. $\mathcal{I}$ then generates $U_{\mathcal{I}}$ based on a standard function and replaces $PK_{\mathcal{I}-1}$ and $U_{\mathcal{I}-1}$ with $PK_{\mathcal{I}}$ and $U_{\mathcal{I}}$, and broadcasts the route request. The process is repeated until the route request reaches the destination $\mathcal{D}$.

A similar technique can be used to send the route reply. However the authors have proposed a novel approach to send the route reply to identify and eliminate the illegitimate route replies. In this approach, the route replies are forwarded, only if, (a) *the intermediate node has previously forwarded the route request* and, (b) *it can ensure that the destination has successfully received the route request.* The format of the route reply is listed below:

$$\left[RREP, T_{\mathcal{I}+1}(seq, K'_{\mathcal{S}-session}), \{T_{\mathcal{I}+1}\}_{PK_{\mathcal{I}}}\right]$$

$K'_{\mathcal{S}-session}$ is the proof that the destination $\mathcal{D}$ has recovered the session key from the third element of the route request. The intermediate nodes validate $K'_{\mathcal{S}-session}$

by verifying the element $K_{\mathcal{S}-session}(seq, END)$ stored in the route request. $T_{\mathcal{I}+1}$ is a random number chosen by $node_{\mathcal{I}+1}$ and is used as a secret between the intermediate nodes, $\mathcal{I}$ and $node_{\mathcal{I}+1}$. After receiving the route reply, each intermediate node $\mathcal{I}$ attempts to decrypt $\{T_{\mathcal{I}+1}\}_{PK_{\mathcal{I}}}$ in order to recover the final element of the route reply. Given $\{T_{\mathcal{I}+1}\}_{PK_{\mathcal{I}}}$ is encrypted with $PK_{\mathcal{I}}$, only $\mathcal{I}$ can decrypt the packet because $\mathcal{I}$ was the one that had sent $PK_{\mathcal{I}}$ to $node_{\mathcal{I}+1}$ together with the route request.

If the received route reply can be validated by the above process, then the node $\mathcal{I}$ chooses a random number $T_{\mathcal{I}}$, and adds $T_{\mathcal{I}}$ and $T_{\mathcal{I}+1}$ into its route table together with the other fields for the corresponding *seq*. In succession, it also computes $\{T_{\mathcal{I}}\}_{PK_{\mathcal{I}-1}}$ and $T_{\mathcal{I}}(seq, K'_{\mathcal{S}-session})$, and replaces the last two elements of the route reply with them before broadcasting the route reply. At the end of the route reply phase, each intermediate node $\mathcal{I}$ would have established a shared secret with the upstream previous-hop $node_{\mathcal{I}-1}$ and the downstream next-hop $node_{\mathcal{I}+1}$ for a communication flow between the source $\mathcal{S}$ and destination $\mathcal{D}$ nodes. Figure 8.1 details the sequence of both the route request and the route reply phases with an adequate explanation.

After the establishment of anonymous routes, the successive data transmission and route maintenance are achieved through the shared secrets. *Tags* are used to minimise the computation overheads at the nodes for validating and forwarding the data packets, where the intermediate nodes are required to validate only the tags instead of validating the complete data packets. In the following section, we will analyse how the attacker can perform flooding attacks with the ASR.

8.3 Analysis

As discussed in Chapter 2, the ASR provides a high degree of mutual anonymity compared with the other approaches. However, similar to the other approaches, there are some significant disadvantages with the ASR. Although the authors state that the initial validation of the route request is made through symmetric key and requires less computational resources, an attacker can feasibly flood the network with malicious route requests. Furthermore, those route requests can target a non-existing destination. Therefore, those route requests are likely to be forwarded by most of the nodes in the network, if only, because those route requests are maliciously embedded with a fresh sequence number. Since it is not feasible to differentiate the packets that originate from a particular source node or to identify the packets destined for a particular destination node, it is extremely difficult to deal with flooding attacks. For example, a node receiving a flood of packets from its upstream previous-hop node cannot determine whether it is flooded by, (a) *its previous-hop* or (b) *the upstream nodes that are prior to its previous-hop*. In the traditional MANET, where communications are non-anonymous, flooding attacks are detected based on the rate at which the packets are generated by the upstream source nodes or received by the destination nodes. Since it is not feasible to track back to either the source or the destination in an anonymous network, it is extremely hard to apply the existing approaches to defend against flooding attacks in the anonymous network.

Similarly, packet drop attacks are detected in the traditional MANET by relating the incoming and outgoing packets from a node. Since all nodes in an anonymous network act as a mix, it is impossible to relate an ingoing and outgoing packet from a mix. Thus, it becomes extremely difficult to determine whether the intermediate nodes that are acting as mixes are successfully forwarding or dropping the packets.

In summary, both the flooding and the packet drop attacks can severely degrade the performance of the MANET that supports anonymous communication. In the following section, we have adapted our fellowship model to deal with the flooding attacks in the MANET that supports anonymous communications. We will not consider packet drop attacks in this chapter for the above-stated reasons.

8.4 Adapted Fellowship Model

We inherit the assumption from the ASR that there is a shared secret between any two nodes, at least because the intermediate nodes can authenticate their neighbours and can therefore defend against the MAC or IP spoofing attacks. To meet the requirements of an anonymous communication, we have made the following changes to the operation of our fellowship model.

Note that, in the case of anonymous communications, a mobile node can only govern the rate of the packets that it can accept from a previous-hop, but cannot determine the rate at which a next-hop can transmit packets on its behalf. Therefore, the need for the enforcement component becomes obsolete in the adapted version of the fellowship model and, hence, is disabled during the operation.

The performance overhead and the criticality and sensitivity of a communication rule out the design option of enabling the intermediate nodes to negotiate bandwidth sharing with their upstream and downstream nodes. Hence, we believe in initialising mobile nodes that support the anonymous communications with pre-defined thresholds. Given that the design discourages the usage of varying thresholds and provided that the success of the adaptation rests on the minimisation of the performance overhead, the principle of employing a contribution share (η) is eliminated in our adapted model together with its offshoots, merit(σ) and demerit(λ) factors. As a result, the restoration component is retired in our adapted version.

To recapitulate, the above design adaptations fit well for the managed and tightly controlled anonymous communications, while the original fellowship model presented in Chapter 3 was primarily designed to accommodate non-anonymous communications in both managed and unmanaged (pure) MANETs. Therefore, our adapted fellowship model is reduced to only the rate-limitation component, whose functionality suitably meets the requirement of the restricting flooded packets without imposing an additional performance overhead on the MANET that supports anonymous communications.

As in Chapter 3, each incoming packet is passed on to the rate-limitation before being transmitted to the next-hop. Recall that the rate-limitation diminishes the flooding attacks based on the condition that each node is obligated to share the bandwidth with its neighbours. Therefore, at the initial stages of deployment, each mobile node is initialised with a pre-defined set of values for the thresholds (as explained later) for the other mobile nodes. Such an arrangement eliminates the role of an online CA for managing the thresholds. The reception-threshold (τ) dictates the maximum number of

Table 8.1: Threshold Parameters for Rate-limitation

Parameter Description	
Maximum number of packets permitted to transmit in an interval	τ
Maximum number of intervals the flooding node can exceed τ before being black-listed	ρ
Minimum number of consecutive intervals a flooding node has to abide τ for being white-listed	μ

packets that a node can accept from a previous-hop or transmit to a next-hop during an interval. In our adaptation, the negotiation threshold (ρ) permits the maximum number of times a flooder can exceed τ before being black-listed. Note that ρ is introduced here primarily to reduce the effect of false positives and we recommend a low value for the threshold. The redemption-threshold (μ) denotes the number of consecutive intervals for which the flooding node has to abide within τ, for it to be white-listed or redeemed. Note that μ is fixed to a higher value in comparison to ρ because a blacklisted flooder has to demonstrate more commitment for it to be white-listed.

8.4.1 System Operation

In this section, we explain the operation of the rate-limitation, in particular, at the receiving end of a fellowship-enabled node.

Let us now consider the operation of the rate-limitation component. Remember that the rate-limitation component at every mobile node monitors the packets transmitted by their immediate upstream nodes during an interval. In such a situation, if the received packets exceed the pre-determined τ within a given interval, then the subsequent packets are dropped as expected. If the same upstream or previous-hop exceeds τ by consecutive ρ intervals, then the receiving node deems its upstream neighbour as a flooder. As a result, the upstream neighbour is blacklisted as a *flooder* and all the packets received from the upstream neighbour are discarded until the flooder terminates its behaviour. We recommend ρ intervals before blacklisting a neighbour to prevent accidental blacklisting of the neighbour. However, the fellowship-enabled node continues to monitor the behaviour of the blacklisted node during the subsequent intervals. Once the flooder demonstrates a repenting behaviour by abiding its transmission within τ, then the flooder's repenting behaviour is taken into account. To be white-listed or redeemed, the blacklisted flooder is enforced to exhibit benign behaviour by abiding its transmissions within τ for μ intervals; this is known as the *redemption-threshold*. Given that the blacklisted neighbour is observed to be benign, the monitoring node then whitelists the neighbour and begins to forward packets for the neighbour. This not only provides an opportunity for the blacklisted neighbour to rejoin the network, but also forces the blacklisted neighbour to drain its energy to prove that it has been repenting. However, the redemption does not guarantee that the flooder will not be blacklisted for future flooding behaviours. Choosing a greater value for μ in comparison with ρ confirms that the blacklisted neighbour has to repent for its past flooding behaviour. In the case of a connection-oriented anonymous data flow, such blacklisting favours the source node choosing an alternative route.

Since anonymous communications are only featured in the managed MANET, a maintenance routine or notification alert can be triggered whenever a mix fails to comply with ρ. This is why the mix can either be compromised or faulty; in which case, an isolation and investigation is warranted. Note that the functionality of the rate-limitation is aggregated at all the monitoring neighbours against a flooder because of the broadcasts. This assures that a flooder's impact is confined to a single hop. In addition, it not only eliminates the need for techniques to trace back to the source of the flooder but also drains the resources of the flooder.

The fellowship-extended mobile nodes that receive multiple route requests from the upstream neighbours can deploy congestion control techniques such as the *drop trail*. Even if the route request packets are dropped at some nodes owing to congestions the routes can still be established through another node, provided a valid path exists. This is achievable because of the broadcast nature of the route request. Alternatively, if no paths are available, then the source node can detect the failure based on the non-reception of a route reply within a specified time. In this case, the source node can randomly wait for a time interval and then can send another route request.

The adapted model inherits several advantages from the original model that was discussed in Chapter 3. First, the adapted model is straight forward and complements the requirement of the MANETs that support anonymous communications. Second, the nodes have to maintain only the transmission rate for their upstream neighbours. Hence, the computation overhead is minimal. Furthermore, our adapted model deals with a flooding attack by targeting the *source of the attack*, *i.e.* containing flooding packets nearest to the flooder. This approach not only aids in saving considerable resources for the non-flooding mobile nodes, but also enables them to efficiently identify and isolate the flooders in the network. Even if the source and destination nodes or two intermediate nodes collude within an environment, the maximum number of packets that can be generated by each colluding node is limited by τ. As each of them exceeds τ by ρ intervals, the adapted approach restricts their packets from being propagated. In addition, our adapted model provides a mechanism that allows the repenting flooders to rejoin the network, if they exhibit benign behaviour for the specified time interval. However, flooders are notably penalised for their past flooding before they can rejoin the network.

8.5 Simulation

We have used the NS2 to investigate the performance of our adapted fellowship version against flooders in the MANET that supports anonymous communications. Given that the extensive simulation results were presented in Chapter 3 to demonstrate the efficiency and operation of the fellowship model, here, we present only the small-scale simulation results to substantiate our claim.

8.5.1 Simulation Setup

In this section, we present the parameters chosen for the adapted version of our fellowship model together with the simulation scenarios. Most of the general simulation parameters and the parameters that are attributed to the flooding behaviour are inherited from Chapter 3. All these simulation parameters are summarised in Table 8.2.

Table 8.2: Simulation Parameters for NS2, Fellowship and Adversary

NS2 Parameters	
Mobility	Random waypoint
Radio Transmission Range	250m
Traffic Type	CBR
Date Rate	11Mbps
Payload Size	1000 bytes
Average Maximum Velocity V_{max}	20m/s
Average Pause Time	10s
Average Simulation Area	$500 * 500m^2$
Total Number of Nodes	10
Fellowship Parameters	
Reception threshold (τ)	Variable
Negotiation threshold (ρ)	Variable
Redemption threshold (μ)	Variable
Malicious Parameters	
Probability of Malicious Action	Persistent / Selective
Distribution of Malicious Action	Random

The mobile nodes that do not enable our adapted fellowship model, but are incorporated with the ASR protocol, are called the *ASR-MAC nodes*. Alternatively, the mobile nodes that are activated with our adapted fellowship model and the ASR protocol are known as the *fellowship-extended nodes*. The nodes that perform the flooding attacks are called the *malicious nodes*.

We conducted simulations for the following compositions, namely, (a) *fellowship-extended and malicious nodes* and, (b) *the ASR-MAC and malicious nodes*. These compositions (*i.e.* the fellowship-extended and the ASR-MAC nodes) are independently simulated against the malicious nodes with identical parameters for every simulation scenario mentioned below. Since identical parameters were maintained, it is feasible to compare the performance of the fellowship-extended and the ASR-MAC nodes against the malicious nodes. Within the composition of the fellowship-extended and malicious nodes, the performance of the fellowship-extended nodes is evaluated for the different values of τ, ρ and μ. The simulation result for each of those compositions is derived from an average of 20 executions.

The performance metrics evaluated for all scenarios are mentioned below:

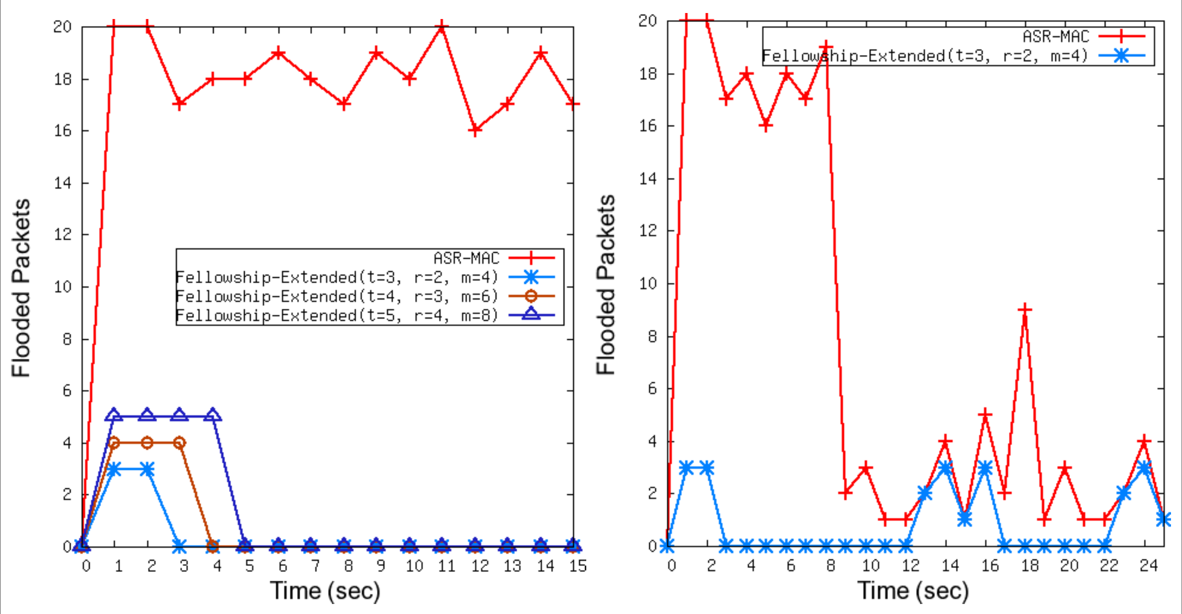
- **Responsiveness** is the time taken to detect, deem and inhibit a flooder from impacting the network.

- **Reactiveness** is the time taken to detect and redeem a repenting or corrected faulty node back into the networking.

The performance metrics such as the packet or byte overhead are not evaluated in our simulations given that the fellowship nodes do not generate additional packets or headers to exchange control messages. Furthermore, the performance metric (PDR) evaluated in Chapter 4 could not be considered here due to the anonymous nature of the MANET communications. However, it is feasible if a marker is explicitly added to the packets to trace the anonymous communications during the simulations and we leave it for future work. We considered reactiveness and responsiveness as the performance metrics to measure the time taken for the anonymous network to become functional in the presence of flooders. Recall that anonymous communications warrant broadcast of packets and therefore, those metrics present an insight into the time taken to resume a functional MANET. The simulation scenarios considered for the performance analysis are:

Scenario I. In this scenario, we evaluated the performance of the fellowship-extended and the ASR-MAC nodes against the malicious nodes that persistently flooded the packets with a pause time of 10 s, a V_{max} of 20 m/s, and a simulation area of $500 * 500m^2$. The objective of this scenario is to discover the responsiveness and reactiveness of both the fellowship-extended and the ASR-MAC nodes against the persistent flooders. Given that a maximum of 20 packets can be transmitted during an interval, the performance of the fellowship-extended nodes is then analysed for the different values of the thresholds.

Scenario II. This scenario presents the performance evaluation of the fellowship-extended and the ASR-MAC nodes against selectively flooding the malicious



(a) Performance against Persistent Flooders (b) Performance against Selective Flooders

FIGURE 8.2: Performance of ASR-MAC and fellowship-extended nodes.

nodes with a pause time of 10 s, a V_{max} of 20m/s, and a simulation area of $500*500m^2$. The responsiveness and reactiveness of both the fellowship-extended and the ASR-MAC nodes are discovered against the selective flooders. Here, the thresholds for the fellowship-extended nodes are determined, based on their performance against the persistent flooders.

8.5.2 Simulation Results

8.5.2.1 Scenario I

Figure 8.2(a) displays the performance comparisons between the fellowship-extended and the ASR-MAC nodes against the persistent flooders. As shown in Figure 8.2(a), the bandwidth of the ASR-MAC nodes is completely dominated by the flooded packets. When the rate of the flooded packets exceeds their bandwidth, they experience congestion and therefore discard the subsequent packets. The primary reason for the ASR-MAC nodes being vulnerable to flooders is that they fail to evaluate the rate of the received packets and therefore fall short in identifying the upstream flooders.

Note that the thresholds, τ, ρ, and μ are represented as *t, r,* and m in both Figures 8.2(a) and 8.2(b). The fellowship-extended nodes measure the rate of the received packet via τ and take ρ intervals to determine whether the upstream previous-hop is a flooder. During this interval, the upstream nodes are observed for their behaviour. Any upstream node that is continuously flooding is identified as a persistent flooder; their packets are dropped from then onwards until they end their flooding behaviour. Since μ signifies the *penalty time period* for which a repenting flooder is forced to demonstrate the change in behaviour, μ does not have any relevance in the case of a persistent flooder. The simulation was carried out with different values of τ, ρ, and μ. Although μ plays an insignificant role in the case of a persistent flooder, the values chosen for τ and ρ determine the rate of the accepted packets and the total count of intervals taken to determine the behaviour of the upstream nodes. The choice of values is left to the implementer to exercise their decision-making such as, whether to deploy an optimistic or a pessimistic design. In summary, it is evident that the fellowship-extended nodes effectively identify and isolate persistent flooders and restrict their flooded packets from being propagated beyond a hop.

8.5.2.2 Scenario II

As shown in Figure 8.2(b), the bandwidth usage of an ASR-MAC node varies in proportion to the rate of the packets received from an immediate upstream or previous-hop neighbour. Alternatively, a fellowship-extended node effectively detects and isolates its upstream selective flooder using the thresholds τ and ρ. The role of μ can be visualised in Figure 8.2(b) whenever the upstream selective flooder attempts to exhibit a repenting behaviour. Recall that μ does not come into play until the deemed selective flooder complies to τ for ρ intervals, which can be seen from 9 s to 12 s in this case. The fellowship-extended node then forwards packets for the selective flooder from 13 s onwards. However, the violation of τ at 14 s causes the fellowship-extended node to observe the selective flooder's behaviour for ρ intervals. Nevertheless, if the selective flooder demonstrates a repenting behaviour by complying to τ for ρ intervals after being blacklisted at 16 s (*i.e.* from 17 s to 21 s), then μ remains deactivated. From the analysis, it is evident that selective flooders are significantly penalised before they could even rejoin the network. However, they are forbidden from joining the network as long as they exhibit a repenting behaviour. Based on our study, we recommend the μ value to be at least twice the value of ρ.

8.6 Conclusion

In this chapter, we have successfully considered the ASR protocol among other available anonymous protocols for the MANET and have then analysed how an attacker can effectively perform both flooding and packet dropping attacks. We then performed our study and presented how it is feasible to mitigate the flooding attacks, but not the packet drop attacks because of the implicit requirement of the anonymous network to separate the ingoing and outgoing packets from a mix. We then adapted our fellowship model to mitigate the flooding attacks in the MANET that supports anonymous communications in which the flooders are efficiently identified and eliminated from the network. Most importantly, the adapted model complements the requirements of anonymous communications by not imposing a performance overhead and inherits the feature of the fellowship to defend against flooding attack nearest to the source of the attack. Furthermore, the adapted fellowship model does not incur any additional overhead in terms of the additional packets to identify the benign and malicious behaviour of the neighbouring nodes. The adapted model also provides a mechanism that enables the repenting nodes to rejoin the network and to isolate those rejoined nodes in the case of flooding behaviour. The simulation results confirm these characteristics and promises that the adapted fellowship model is efficient for counteracting flooding attacks in the MANETs supporting anonymity.

9

SL-SMRTI: Subjective Logic based SMRTI

9.1 Introduction

As we have discussed in Chapter 2, trust models that have been proposed for the MANET vary in their properties and also from the way they make trust-enhanced decisions for routing. Although they lack consensus on the definition of *trust* at a more fundamental level, most, if not all, these models often fail to represent the aspect of *ignorance* while establishing trust relationships between the mobile nodes. Hence, a trust relationship between two mobile nodes may not always reflect the actual relationship and, consequently, the executed decision may not always be accurate.

For instance, the existing nodes in a network may not have a record of past evidence to trust or distrust a newly-joining node. Assigning an arbitrary level of trust for the new node poses several issues. Related trust models resolve this issue either by pessimistically assigning a low [294] or a neutral level of trust [401] or by optimistically assigning a high level of trust [301] to the new node. The purpose of the pessimistic approach is to compel the new node to exhibit a consistent benign behaviour from the point of entry. However, in some of these models, it is not always clear how the less-trusted new node is selected for communications when nodes with high trust values exist. With the optimistic approach, the intent is to promptly identify whether the new node exhibits malicious behaviour from the point of entry. Prompt identification is feasible because the high level of trust assigned for a new node decreases rapidly as the malicious behaviour increases. However, the optimistic approach fails to discriminate a new node from the existing nodes, whose dynamically-changing selective behaviour has warranted the same level of trust. All these result because the existing nodes fail to explicitly represent their ignorance of a newly-joining node's behaviour. It also extends to nodes that have already established a trust relationship with one another. For example, when a node moves away from a neighbour (owing to mobility), it is unclear whether the neighbour should be considered with the same level of trust or distrust during the next interaction (when the neighbour returns). It may be that the neighbour has retained its behaviour that might have been either benign or malicious. Considering that had the neighbour been benign, there is an equal chance for the neighbour to be compromised prior to the next interaction. Alternatively, if the neighbour had been malicious, then it may be also repenting and expecting an interaction to improve its relationship. These issues are often solved by either increasing or decreasing the trust value of the nodes in proportion to the duration for which they are out of communication. The main problem with this approach is that a node's ignorance about the behaviour of the other nodes is represented by either increasing or decreasing its trust value for them, which actually should denote their benign or malicious behaviour respectively. Hence, failing to explicitly represent the notion of ignorance and thereby uncertainty has a fundamental impact on trust relationships.

In this chapter, we present how our novel ***Subjective Logic based SMRTI (SL-SMRTI)*** enables mobile nodes to explicitly represent and manage ignorance as *uncertainty* in their trust relationships with other nodes. Our approach not only enables the mobile nodes to distinguish the new nodes from the existing nodes, but also facilitates their resolution of the ignorance that results when they move away from the other nodes. Although some sections of this chapter are similar to Chapter 4, they are included for the sake completeness and to provide the background; however, those sections can be conveniently skipped for the sake of readability.

The rest of this chapter is organised as follows. Initially, we will recapitulate the SMRTI in the next section. We will then establish the context for the subjective logic in §9.3. In §9.4, we will not only present the architecture of the SL-SMRTI but also its tightly-coupled operation with the AOMDV that facilitates the AOMDV making trust-enhanced decisions for the various event-specific contexts. In addition, we will discuss how the SL-SMRTI enabled mobile nodes collect evidence and formulates such evidence into opinions using subjective logic. Section 9.5 describes an adversary model against which the performance of the AOMDV and the SL-SMRTI nodes are evaluated. The section then concludes with remarks on how the SL-SMRTI effectively defends against the threats posed by the adversary. The simulation setup and performance comparison between the SL-SMRTI and the AOMDV nodes in the presence of the adversary model will be demonstrated in §9.6. Finally, §9.7 explores the limitations of the SL-SMRTI and then §9.8 concludes the chapter.

9.2 SMRTI – Overview

This section recaps the operation of the SMRTI from Chapter 4 for the sake of completeness. Alternatively, this section can be skipped for ease of readability. As we have detailed in Chapter 4, the SMRTI-enabled mobile nodes formulate their *opinions* for other nodes based on the evidence of trustworthiness collected from the benign and malicious behaviours of those nodes. In the SMRTI, a *direct opinion* is formulated depending on the evidence captured from the one-to-one interactions with the neighbours. Similarly, an *observed opinion* is formulated depending on the evidence captured by observing the interactions of the neighbours. For both direct and observed opinions, the SMRTI relies on the IDS-based monitor mechanism so that the mobile nodes can capture evidence for the benign and misbehaviours of their neighbours. Lastly, the evidence captured from the derived recommendations is formulated into a *recommended opinion*. The direct opinion enables the mobile nodes to classify their neighbours as either malicious or benign. Capturing evidence from the interactions of the neighbours for an observed opinion enables the mobile nodes to identify the malicious neighbours even before interacting with them. The recommended opinion enables the mobile nodes to identify and establish trust relationships with other trustworthy nodes. As we have demonstrated in §4.6.3, the SMRTI prevents a mobile node's recommended opinion from being corrupted by a recommender's recommendation and this, in turn, enables the node to better resolve the issues concerned with the recommender's bias. Given that the recommenders do not disseminate their opinions as recommendations in our model, they are eventually prevented from exhibiting both honest-elicitation and free-riding behaviours. The mobile nodes then build a trust relationship with the other nodes using their opinions (direct, observed and recommended). The SMRTI assists the mobile nodes to make effective decisions for various event-specific contexts depending on the policies defined for those contexts and the trust relationship established with the nodes that are involved in those contexts.

9.3 Subjective Logic and MANET

9.3.1 Overview of Subjective Logic

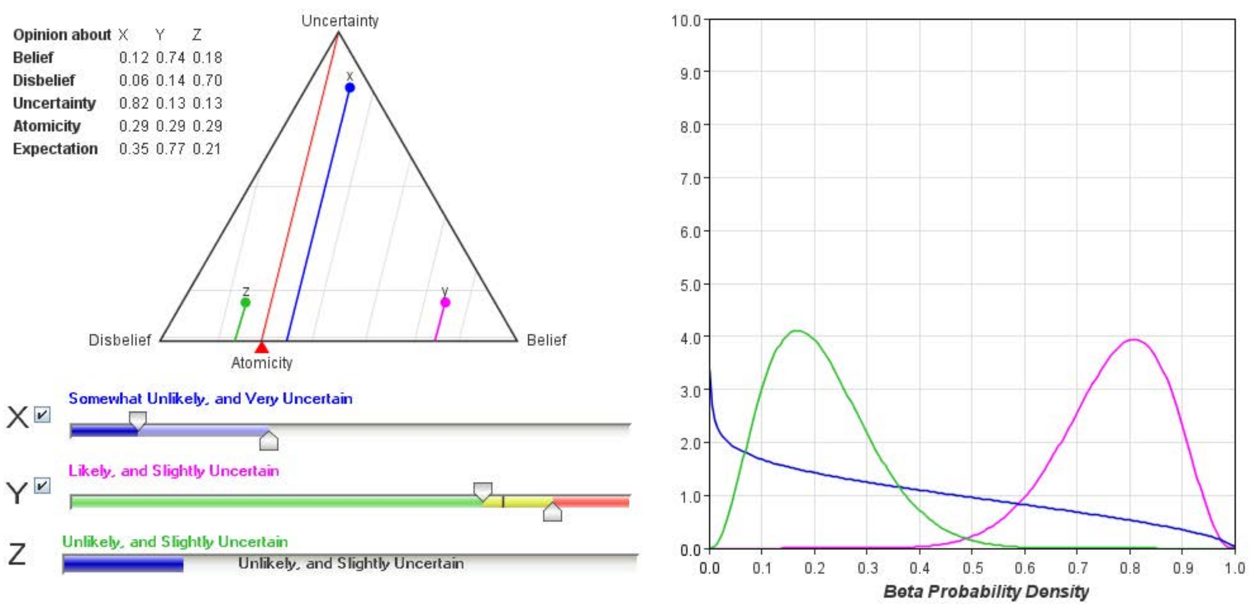
Subjective logic [183–185, 187, 188] is a probabilistic logic for uncertainty that is ideally suitable for analysing situations that are characterised by incomplete knowledge, especially for trust and Bayesian networks [312]. Therefore, subjective logic makes it possible to express uncertainty about the probability values themselves, meaning that it is possible to reason with argument models in presence of uncertain or partially incomplete evidence. Subjective logic mathematically formalises the fundamental aspect of human psychology, where individuals can never determine with absolute certainty whether a proposition about the universe is true or false. Furthermore, it includes the philosophical fact that the truth of a proposition is always expressed by an individual and hence, it can never be considered as a general and objective belief. For these reasons, we considered subjective logic instead of other models that represent uncertainty, such as, Bayesian and fuzzy randomness models.

The arguments in subjective logic are subjective opinions about propositions and these opinions can be of either binomial or multinomial. Note that a binomial opinion applies to a single proposition and can be represented as a *Beta distribution*. Alternatively, a multinomial opinion applies to a collection of properties and can be represented as a *Dirichlet distribution*. Subjective logic provides an algebra, based on the correspondence between the subjective opinions and the Beta or Dirichlet distributions. Rooted in the sound mathematical foundation of the *Dempster-Shafer belief theory of evidence* [350] and with the ability to explicitly represent and manage incomplete knowledge or ignorance in evidential beliefs, subjective logic therefore emerges as an attractive tool for us to handle trust relationships in inherently dynamic, open and uncertain networks such as the MANET. Since we are concerned with single proposition such as the trustworthiness of a mobile node, from here onwards we confine such discussions to *binomial subjective opinions*.

A binomial subjective opinion denotes the *subjective belief, disbelief* and *ignorance* about the truth of a proposition. An opinion is denoted as $\omega_x^{\mathcal{A}}$ or $\omega(\mathcal{A} : x)$, where $\mathcal{A}$ is the belief owner and x is the proportion to which the opinion applies. Although the belief ownership can be normally omitted, it can be included whenever required and the subject can have a semantic representation of the proportion. According to the Dempster-Shafer belief theory of evidence, the proposition (x) belongs to a frame of discernment (Θ) that contains the possible states of a given system. Hence, the attributes of an opinion are the subject, proposition and its frame.

9.3.1.1 Binomial Subjective Opinion

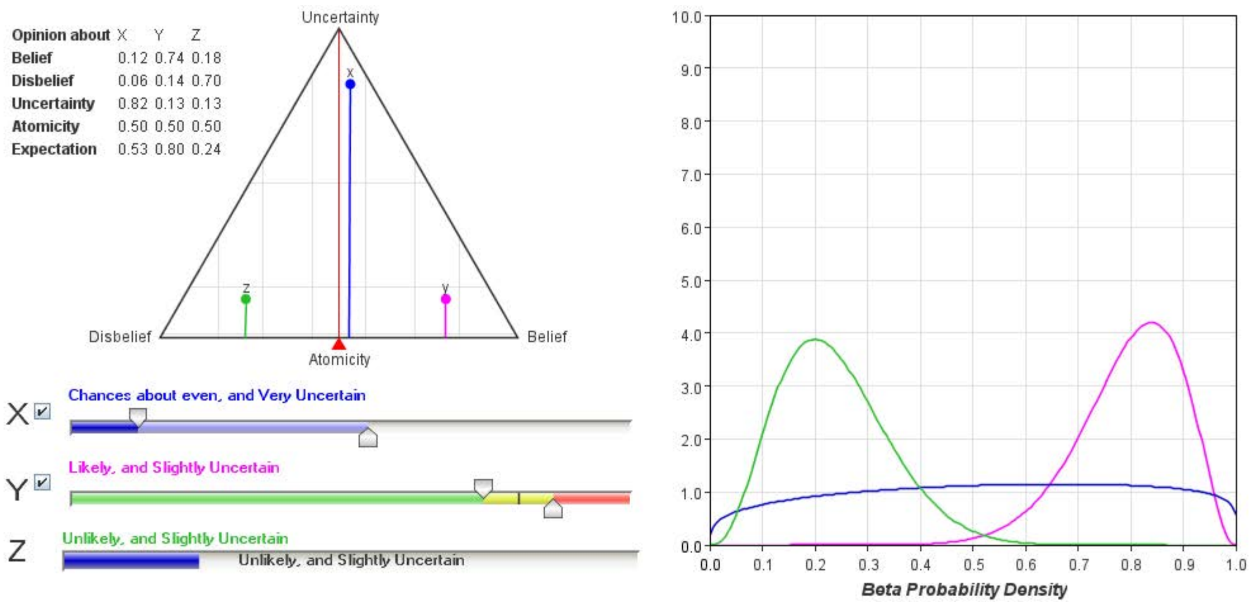
A binomial subjective opinion about the truth of x is an ordered quadruple, such that $\omega_x = (b_x, d_x, u_x, a_x)$. In the quadruple, b_x is the belief that the proposition x is true,



Opinion about	X	Y	Z
Belief	0.12	0.74	0.18
Disbelief	0.06	0.14	0.70
Uncertainty	0.82	0.13	0.13
Atomicity	0.29	0.29	0.29
Expectation	0.35	0.77	0.21

Uncertainty
Disbelief
Belief
Atomicity
X
Somewhat Unlikely, and Very Uncertain
Y
Likely, and Slightly Uncertain
Z
Unlikely, and Slightly Uncertain
Unlikely, and Slightly Uncertain
Beta Probability Density

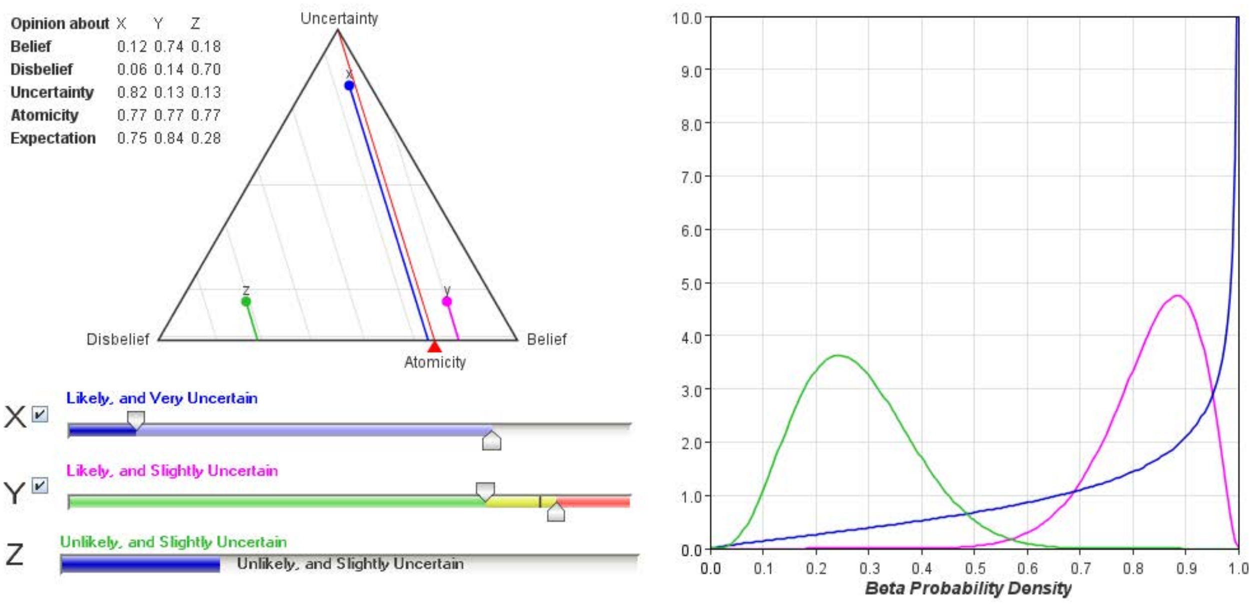
(a) Opinions with atomicity lower than 0.5



Opinion about	X	Y	Z
Belief	0.12	0.74	0.18
Disbelief	0.06	0.14	0.70
Uncertainty	0.82	0.13	0.13
Atomicity	0.50	0.50	0.50
Expectation	0.53	0.80	0.24

Uncertainty
Disbelief
Belief
Atomicity
X
Chances about even, and Very Uncertain
Y
Likely, and Slightly Uncertain
Z
Unlikely, and Slightly Uncertain
Unlikely, and Slightly Uncertain
Beta Probability Density

(b) Opinions with atomicity equal to 0.5



Opinion about	X	Y	Z
Belief	0.12	0.74	0.18
Disbelief	0.06	0.14	0.70
Uncertainty	0.82	0.13	0.13
Atomicity	0.77	0.77	0.77
Expectation	0.75	0.84	0.28

Uncertainty
Disbelief
Belief
Atomicity
X
Likely, and Very Uncertain
Y
Likely, and Slightly Uncertain
Z
Unlikely, and Slightly Uncertain
Unlikely, and Slightly Uncertain
Beta Probability Density

(c) Opinions with atomicity greater than 0.5

FIGURE 9.1: Visualisation of Subjective Opinions.

and similarly d_x is the disbelief that x is false. Consequently, *uncertainty* (u_x) is the proportion of belief that is neither committed to the truth nor falsehood of x, and the *base rate* (a_x) is a *priori* in the probability in the absence of evidence and the default value is the relative atomicity, *i.e.* 0.5 for binary state spaces. In other words, the base rate (a_x) determines how optimistically the uncertainty is viewed as a belief when the subjective opinion is used. All of $b_x, d_x, u_x, a_x \in [0.0, 1.0]$ and satisfy $(b_x + d_x + u_x) = 1$. This linear constraint restricts the possible points to a two-dimensional triangular space as shown in Figure 9.1. When the subjective opinion has $(b_x + d_x) = 1$, it then reflects the traditional probabilities. Instances $b_x = 1$ and $d_x = 1$ are equivalent to binary logic *true* and *false*, respectively. However, the opinion expresses degrees of ignorance when $0 < (b_x + d_x) < 1$, and complete ignorance when $(b_x + d_x) = 0$. Finally, the probability expectation value of an opinion is given as $E(\omega_x) = (b_x + u_x a_x)$.

9.3.1.2 Visualisation of Opinions

Binomial subjective opinions can be visualised on an equilateral triangle as shown in Figure 9.1. For instance, the opinions about three different propositions, *x, y,* and *z*, can be visualised as three different points within the triangle and represented as triples (b_x, d_x, u_x), (b_y, d_y, u_y), and (b_z, d_z, u_z) respectively. The vertices, *b, d,* and *u*, indicate the belief, disbelief and uncertainty, respectively. Hence, in Figures 9.1(a), 9.1(b) and 9.1(c), a strong positive opinion (ω_y) is represented close towards the bottom right belief vertex and a strong negative belief (ω_z) towards the bottom left disbelief vertex. The top middle region of the triangle is used to represent opinions (*e.g.* ω_x) that expresses a high degree of uncertainty.

As mentioned above, a base rate that is also known as the relative atomicity determines how uncertainty is viewed as belief, and is shown as a red pointer at the bottom along the base line or the probability axis that connects the belief and disbelief vertices. The *base rate projector line* connects the base rate to the uncertainty vertex. When an opinion is used in a decision, the opinion projects its probability expectation (E) onto the probability axis, parallel to the base rate projector line. A base rate of 0.0 causes uncertainty to be viewed as disbelief and a base rate of 1.0

causes uncertainty to be viewed as belief. However, a base rate of 0.5 causes uncertainty to be viewed as half as positively as the actual belief. As shown in Figure 9.1(c), the opinion $\omega_x = (0.5, 0.5, 0.5)$ unravels as half-believed and half-uncertain, and its expectation is $E(\omega_x) = (b_x + u_x a_x) = [0.5 + (0.5) \cdot (0.5)] = 0.75$. However, an entirely uncertain opinion $\omega_v = (0.0, 0.0, 1.0, a_v)$ will yield an expectation that equals the base rate, $E(\omega_v) = (b_v + u_v a_v) = [0.0 + (1.0) a_v] = a_v$. Hence, the base rate becomes the default opinion for unknown mobile nodes and therefore, as seen in Figure 9.1(c), we choose 0.5 as the default value for the binary state spaces. Figures 9.1(a) and 9.1(b) present the conditions where the base rate is either greater or lower than 0.5.

Alternative visualisations of the three opinion points (*x, y,* and *z*) displayed in the triangle can be seen as the three bars immediately under the triangle in Figure 9.1. These bars are coloured to quickly present the nature of an opinion with the marker above or within each of the bar displaying the element of the two-dimensional fuzzy set to which the opinion corresponds. The *expectation bar* is the first bar that indicates the expectation of the opinion (ω_x). The dark blue region on the bar represents the amount of expectation that is accounted to the belief, while the light blue region represents the amount of expectation accounted to uncertainty. In the case of the *Bayesian bar* (which is the second bar), the black line indicates the overall expectation of the opinion (ω_y) and the green region on the bar represents the amount of belief in the truth of proposition *y.* The red region represents the amount of disbelief in the truth of proposition *y.* The yellow region between the above-mentioned regions represents the amount of uncertainty regarding the truth of proposition *y.* The third and last bar is a *Fuzzy bar* that indicates the expectation of opinion (ω_z).

As shown in [185], the probability density over binary event spaces can be expressed as the *beta Probability Density Function (beta PDF)*, and denoted as $beta(\alpha, \beta)$. Therefore, the 'bijective mapping' between the opinion and the $beta(\alpha, \beta)$ parameters can be seen in [185]. Since belief functions are interpreted by $beta(\alpha, \beta)$ in classical statistical terms, the beta distributions are visualised on a plot to the right of triangle in Figure 9.1.

9.3.2 Subjective Logic tailored for MANET

Although subjective logic provides a framework for expressing opinions about any logical statement, the only logical statement that interests us is the trustworthiness of the mobile nodes. Hence, we omit the subscript x (the proposition) from the opinion tuple and its parameters. Given that we are only interested in the binomial subjective opinions and the base rate is kept constant at 0.5, an opinion would then be expressed as $\omega = (b, d, u)$. However, the decentralised design of the SMRTI enables mobile nodes to express their subjective opinion about every other mobile node based on the mode m (direct, observed, recommended and global[1]) through which the opinion is formalised, triggers us to denote an opinion as follows. A subjective opinion is written as ${}^{\mathcal{A}}\omega^{m}_{\mathcal{B}}$, where $\mathcal{A}$ points to the *trustor*, *i.e.* the subject belief owner, while $\mathcal{B}$ points to the *trustee* based on the mode m through which the opinion or trust relationship is established.

Here we present the revised representation of a subjective opinion in the context of the MANET, given as ${}^{\mathcal{A}}\omega^{m}_{\mathcal{B}}$ in (9.1), wherein the belief (${}^{\mathcal{A}}b^{m}_{\mathcal{B}}$) gives the probability of node $\mathcal{A}$'s trust in node $\mathcal{B}$, depending on $\mathcal{B}$'s benign behaviour captured through the mode m. Similarly, the disbelief (${}^{\mathcal{A}}d^{m}_{\mathcal{B}}$) presents the probability of $\mathcal{A}$'s distrust on $\mathcal{B}$ depending on $\mathcal{B}$'s malicious behaviour captured through the mode m. Finally, the uncertainty (${}^{\mathcal{A}}u^{m}_{\mathcal{B}}$) represents the probability of $\mathcal{A}$'s ignorance in $\mathcal{B}$ within the mode m. Belief, disbelief and uncertainty satisfy the conditions in (9.1) and this linear constraint confines the points to a two-dimensional triangular space, as shown in Figure 9.1.

$$\begin{aligned}
&{}^{\mathcal{A}}\omega^{m}_{\mathcal{B}} = \left({}^{\mathcal{A}}b^{m}_{\mathcal{B}},\, {}^{\mathcal{A}}d^{m}_{\mathcal{B}},\, {}^{\mathcal{A}}u^{m}_{\mathcal{B}}\right)\\
&\left({}^{\mathcal{A}}b^{m}_{\mathcal{B}} + {}^{\mathcal{A}}d^{m}_{\mathcal{B}} + {}^{\mathcal{A}}u^{m}_{\mathcal{B}}\right) = 1\\
&\left({}^{\mathcal{A}}b^{m}_{\mathcal{B}},\, {}^{\mathcal{A}}d^{m}_{\mathcal{B}},\, {}^{\mathcal{A}}u^{m}_{\mathcal{B}}\right) \in [0,1]
\end{aligned} \tag{9.1}$$

[1]It is the subjective opinion taken as a whole by combining the different types of opinions such as direct, observed and recommended opinions.

9.3.2.1 Subjective Logic Operators for MANET

A number of operators have been defined for subjective logic [185]; some represent generalisations of binary logic and probability calculus operators, while others are unique to the belief theory, because they depend on belief ownership. In this chapter, we are only interested in, (a) *mapping evidence to opinion (for direct and observed modes)*, (b) *deriving opinions based on recommendations* and, (c) *combining direct, observed, and recommended opinions into a global opinion (or global opinions into a single opinion to make event-based decisions).* Hence, we inherit those operators that are relevant to our objective and accordingly tailor them to meet the requirements of the MANET, if required. For the instances where there is no existing operator, we define a new operator to meet the requirement.

9.3.2.2 Evidence-to-opinion Mapping Operator

As we seen earlier, the probability density over the binary event spaces can be expressed as beta PDFs denoted by $beta(\alpha, \beta)$. If p and n express the total number of positive and negative past evidence, respectively, at a base rate a, then α and β can be determined as in (9.2).

$$\begin{aligned} \alpha &= (p + 2a) \\ \beta &= [n + 2(1 - a)] \end{aligned} \tag{9.2}$$

Since we maintain the base rate (a) at 0.5, α and β become *(p+1)* and *(n+1)*, respectively. As shown in [185], the opinion and the beta PDF parameters can analytically undergo the bijective mapping that is given in (9.3).

$$\begin{aligned} b &= \frac{p}{(p+n+2)} \\ d &= \frac{n}{(p+n+2)} \\ u &= \frac{2}{(p+n+2)}; \quad u \neq 0 \end{aligned} \tag{9.3}$$

Hence, a totally ignorant opinion $\omega = (0.0, 0.0, 1.0)$ with $u = 1$ is equivalent to the *uniform beta PDF (1, 1)*. Alternatively, a dogmatic opinion with $u = 0$ is equivalent to the spike PDF with infinitesimal width and infinite height expressed by $beta(b_\eta, d_\eta)$, where $\eta \to \infty$. Dogmatic opinions can thus be interpreted as being based on an infinite amount of evidence. However, such a situation is hard to achieve in the MANET for the reasons explained later. In summary, (9.3) allows a mobile node $\mathcal{A}$ to express its opinion ${}^{\mathcal{A}}\omega^{f}_{\mathcal{B}}$ or establish a trust relationship with another mobile node $\mathcal{B}$ depending on the mode f (direct or observed modes for the reasons detailed below) through which it had collected positive and (or) negative evidence for the latter.

Although $\mathcal{A}$ can assign $\mathcal{B}$'s initial opinion, ${}^{\mathcal{A}}\omega^{f}_{\mathcal{B}} = (0.0, 0.0, 1.0)$ as uncertainty, it would be able to follow (9.3) to build its opinion for $\mathcal{B}$ by monitoring the latter's behaviour. Nevertheless, (9.4) would not be able to assist $\mathcal{A}$ in representing the notion of ignorance that slips in its opinion for $\mathcal{B}$ when it is separated from $\mathcal{B}$ owing to mobility. For this purpose, we introduce a variable k such that the parameters *(b, d, u)* of ${}^{\mathcal{A}}\omega^{f}_{\mathcal{B}}$ still satisfies (9.2) and the new evidence-to-opinion mapping can also take into account the intermittent cycles for which there is lack of positive and negative evidence. In (9.4), k represents the count of the interval for which $\mathcal{A}$ and $\mathcal{B}$ are out of communication with each other. The interval is based on the average time taken for a communication flow by the node that maps the evidence-to-opinion, (*i.e.* $\mathcal{A}$ in this case). Furthermore, $\mathcal{A}$ applies (9.5) only to formulate direct and observed opinions because it collects positive and negative evidence only through the direct and observed modes respectively. In the case of a recommended opinion, $\mathcal{A}$ would derive the opinion from the inferred recommendations that are explained in detail in §9.3.2.4.

$$\begin{aligned}
{}^{\mathcal{A}}b^{f}_{\mathcal{B}} &= \frac{p}{(p+n+k)} \\
{}^{\mathcal{A}}d^{f}_{\mathcal{B}} &= \frac{n}{(p+n+k)} \\
{}^{\mathcal{A}}u^{f}_{\mathcal{B}} &= \frac{k}{(p+n+k)}; \quad u \neq 0
\end{aligned} \tag{9.4}$$

In summary, the evidence-to-opinion mapping operator (∇) holds true only for direct and observed opinions and also expresses the notion of ignorance that slips in when the nodes are separated by mobility. We will describe the process of capturing and evaluating the evidence prior to mapping it into an opinion later in §9.4.6.

9.3.2.3 Consensus Operator

It is natural for a mobile node to combine different types of opinion (direct, observed and recommended) into a single opinion known as a *global opinion*, so that it can make an objective judgment about another node's trustworthiness. Otherwise, a mobile node may be required to combine global opinions that are held for other mobile nodes (which are involved in an event), thus it can make an objective judgment about the context of that event. For this purpose, we inherit the consensus operator of subjective logic that is used to combine different types of opinion into a single opinion.

As given in [185], the consensus operator combines opinions ($\omega_x^{\mathcal{P}}$ and $\omega_x^{\mathcal{Q}}$) that belong to different subjects ($\mathcal{P}$ and $\mathcal{Q}$) respectively, provided those opinions refer to the same proposition (x) but are based on a different set of evidence. Let $\omega_x^{\mathcal{P}} = (b_x^{\mathcal{P}}, d_x^{\mathcal{P}}, u_x^{\mathcal{P}})$ and $\omega_x^{\mathcal{Q}} = (b_x^{\mathcal{Q}}, d_x^{\mathcal{Q}}, u_x^{\mathcal{Q}})$ be the trust in x by $\mathcal{P}$ and $\mathcal{Q}$ respectively. The opinion $\omega_x^{\mathcal{P}\diamond\mathcal{Q}} = (b_x^{\mathcal{P}\diamond\mathcal{Q}}, d_x^{\mathcal{P}\diamond\mathcal{Q}}, u_x^{\mathcal{P}\diamond\mathcal{Q}})$ is the consensus between $\omega_x^{\mathcal{P}}$ and $\omega_x^{\mathcal{Q}}$, which resembles the trust that an imaginary agent $|\mathcal{P}, \mathcal{Q}|$ would have in x, as if that agent represented both $\mathcal{P}$ and $\mathcal{Q}$. The symbol $\oplus$ is used to denote this operator, and hence we get $\omega_x^{\mathcal{P}\diamond\mathcal{Q}} = \omega_x^{\mathcal{P}} \oplus \omega_x^{\mathcal{Q}}$.

Case I: $\left(u_x^{\mathcal{P}} + u_x^{\mathcal{Q}} - u_x^{\mathcal{P}} u_x^{\mathcal{Q}}\right) \neq 0$

$$\begin{aligned} b_x^{\mathcal{P}\diamond\mathcal{Q}} &= \frac{\left(b_x^{\mathcal{P}}u_x^{\mathcal{Q}} + b_x^{\mathcal{Q}}u_x^{\mathcal{P}}\right)}{\left(u_x^{\mathcal{P}} + u_x^{\mathcal{Q}} - u_x^{\mathcal{P}}u_x^{\mathcal{Q}}\right)} \\ d_x^{\mathcal{P}\diamond\mathcal{Q}} &= \frac{\left(d_x^{\mathcal{P}}u_x^{\mathcal{Q}} + d_x^{\mathcal{Q}}u_x^{\mathcal{P}}\right)}{\left(u_x^{\mathcal{P}} + u_x^{\mathcal{Q}} - u_x^{\mathcal{P}}u_x^{\mathcal{Q}}\right)} \\ u_x^{\mathcal{P}\diamond\mathcal{Q}} &= \frac{\left(u_x^{\mathcal{P}}u_x^{\mathcal{Q}}\right)}{\left(u_x^{\mathcal{P}} + u_x^{\mathcal{Q}} - u_x^{\mathcal{P}}u_x^{\mathcal{Q}}\right)} \end{aligned} \tag{9.5}$$

Case II: $\left(u_x^{\mathcal{P}} + u_x^{\mathcal{Q}} - u_x^{\mathcal{P}}u_x^{\mathcal{Q}}\right) = 0$

$$\begin{aligned} b_x^{\mathcal{P}\diamond\mathcal{Q}} &= \frac{\left(\gamma^{\mathcal{P}/\mathcal{Q}} \cdot b_x^{\mathcal{P}} + b_x^{\mathcal{Q}}\right)}{\left(\gamma^{\mathcal{P}/\mathcal{Q}} + 1\right)} \\ d_x^{\mathcal{P}\diamond\mathcal{Q}} &= \frac{\left(\gamma^{\mathcal{P}/\mathcal{Q}} \cdot d_x^{\mathcal{P}} + d_x^{\mathcal{Q}}\right)}{\left(\gamma^{\mathcal{P}/\mathcal{Q}} + 1\right)} \\ u_x^{\mathcal{P}\diamond\mathcal{Q}} &= 0; \quad where,\ \gamma^{\mathcal{P}/\mathcal{Q}} = \lim \frac{u_x^{\mathcal{Q}}}{u_x^{\mathcal{P}}} \end{aligned} \tag{9.6}$$

To understand the meaning of the relative weight γ, we can consider some process with possible outcomes $\{x, \bar{x}\}$, such that the process produces γ times x as $\bar{x}$. For example, let the process be throwing a fair dice in which some mechanism makes sure that $\mathcal{P}$ only observes the outcome of *three* and $\mathcal{Q}$ only observes the outcome of *one*, *two*, *four*, *five*, and *six*. Therefore, $\mathcal{P}$ will think that the dice only produces *three* and $\mathcal{Q}$ will think that the dice never produces *three*. $\mathcal{P}$ and $\mathcal{Q}$ will have the conflicting dogmatic opinions, $\omega_{three}^{\mathcal{P}} = \left(1.0, 0.0, 0.0, \frac{1}{6}\right)$ and $\omega_{three}^{\mathcal{Q}} = \left(0.0, 1.0, 0.0, \frac{1}{6}\right)$ respectively, after infinite observations. Assuming that $\mathcal{Q}$ had noticed five times more events than $\mathcal{P}$; $\mathcal{Q}$ remains five times more dogmatic than $\mathcal{P}$ as $u_{three}^{\mathcal{P}}, u_{three}^{\mathcal{Q}} \to 0$, meaning that the *relative weight* γ between $\mathcal{P}$ and $\mathcal{Q}$ is $\frac{1}{5}$. According to (9.6), the combined opinion would then be $\omega_{three}^{\mathcal{P}\diamond\mathcal{Q}} = \left(\frac{1}{6}, \frac{5}{6}, 0, \frac{1}{6}\right)$ as expected.

Recollect that the SMRTI-enabled mobile nodes neither propagate their opinions nor collect the opinions of other mobile nodes. For that reason, we tailor consensus operator ($\oplus$) to assist a mobile node to combine its three types of opinion (direct, observed and recommended) that are held for another mobile node, where those opinions refer to the same proposition and are based on a distinct set of evidence. The combined opinion is referred as a *global opinion.* Since $\oplus$ effectively reduces uncertainty, it thus enables a mobile node $\mathcal{A}$ to increase its confidence on the expectation value, *i.e.* about the trustworthiness of another mobile node $\mathcal{B}$.

Let us assume that $\mathcal{A}$'s direct opinion for $\mathcal{B}$ is based on the latter's response to its requests and is denoted as ${}^{\mathcal{A}}\omega_{\mathcal{B}}^{dir} = \left({}^{\mathcal{A}}b_{\mathcal{B}}^{dir},\, {}^{\mathcal{A}}d_{\mathcal{B}}^{dir},\, {}^{\mathcal{A}}u_{\mathcal{B}}^{dir}\right)$. Similarly, its observed opinion for $\mathcal{B}$ is based on $\mathcal{B}$'s response to the requests received from its neighbours and is denoted as ${}^{\mathcal{A}}\omega_{\mathcal{B}}^{obs} = \left({}^{\mathcal{A}}b_{\mathcal{B}}^{obs},\, {}^{\mathcal{A}}d_{\mathcal{B}}^{obs},\, {}^{\mathcal{A}}u_{\mathcal{B}}^{obs}\right)$. The combined opinion, ${}^{\mathcal{A}}\omega_{\mathcal{B}}^{dir\Diamond obs} = \left({}^{\mathcal{A}}b_{\mathcal{B}}^{dir\Diamond obs},\, {}^{\mathcal{A}}d_{\mathcal{B}}^{dir\Diamond obs},\, {}^{\mathcal{A}}u_{\mathcal{B}}^{dir\Diamond obs}\right)$ is referred as the consensus between ${}^{\mathcal{A}}\omega_{\mathcal{B}}^{dir}$ and ${}^{\mathcal{A}}\omega_{\mathcal{B}}^{obs}$, and denoted as ${}^{\mathcal{A}}\omega_{\mathcal{B}}^{dir} \oplus {}^{\mathcal{A}}\omega_{\mathcal{B}}^{obs}$.

Equations (9.7) and (9.8) express ${}^{\mathcal{A}}b_{\mathcal{B}}^{dir\Diamond obs}$, ${}^{\mathcal{A}}d_{\mathcal{B}}^{dir\Diamond obs}$, and ${}^{\mathcal{A}}u_{\mathcal{B}}^{dir\Diamond obs}$. They are based on whether $\left({}^{\mathcal{A}}u_{\mathcal{B}}^{dir} + {}^{\mathcal{A}}u_{\mathcal{B}}^{obs} - {}^{\mathcal{A}}u_{\mathcal{B}}^{dir} \cdot {}^{\mathcal{A}}u_{\mathcal{B}}^{obs}\right)$ equals to zero or not. In (9.8), the weighted average of the probabilities is produced when subjective opinions, ${}^{\mathcal{A}}\omega_{\mathcal{B}}^{dir}$ and ${}^{\mathcal{A}}\omega_{\mathcal{B}}^{obs}$ are probability values ${}^{\mathcal{A}}u_{\mathcal{B}}^{dir}$, ${}^{\mathcal{A}}u_{\mathcal{B}}^{obs} \rightarrow 0$.

Finally, $\mathcal{A}$ formulates its global opinion ${}^{\mathcal{A}}\omega_{\mathcal{B}}^{glo}$ for $\mathcal{B}$ by combining ${}^{\mathcal{A}}\omega_{\mathcal{B}}^{dir\Diamond obs}$ with the recommended opinion ${}^{\mathcal{A}}\omega_{\mathcal{B}}^{rec}$ held for $\mathcal{B}$ using $\oplus$. This operation is summarised in (9.9) and the process of formulating the opinion from the inferred recommendations will be explained in detail in §9.3.2.4.

Case I: $\left[{}^{\mathcal{A}}u_{\mathcal{B}}^{dir} + {}^{\mathcal{A}}u_{\mathcal{B}}^{obs} - \left({}^{\mathcal{A}}u_{\mathcal{B}}^{dir} \cdot {}^{\mathcal{A}}u_{\mathcal{B}}^{obs}\right)\right] \neq 0$

$$\begin{aligned}
{}^{\mathcal{A}}b_{\mathcal{B}}^{dir\Diamond obs} &= \frac{\left[\left({}^{\mathcal{A}}b_{\mathcal{B}}^{dir} \cdot {}^{\mathcal{A}}u_{\mathcal{B}}^{obs}\right) + \left({}^{\mathcal{A}}b_{\mathcal{B}}^{obs} \cdot {}^{\mathcal{A}}u_{\mathcal{B}}^{dir}\right)\right]}{\left[{}^{\mathcal{A}}u_{\mathcal{B}}^{dir} + {}^{\mathcal{A}}u_{\mathcal{B}}^{obs} - \left({}^{\mathcal{A}}u_{\mathcal{B}}^{dir} \cdot {}^{\mathcal{A}}u_{\mathcal{B}}^{obs}\right)\right]} \\
{}^{\mathcal{A}}d_{\mathcal{B}}^{dir\Diamond obs} &= \frac{\left[\left({}^{\mathcal{A}}d_{\mathcal{B}}^{dir} \cdot {}^{\mathcal{A}}u_{\mathcal{B}}^{obs}\right) + \left({}^{\mathcal{A}}d_{\mathcal{B}}^{obs} \cdot {}^{\mathcal{A}}u_{\mathcal{B}}^{dir}\right)\right]}{\left[{}^{\mathcal{A}}u_{\mathcal{B}}^{dir} + {}^{\mathcal{A}}u_{\mathcal{B}}^{obs} - \left({}^{\mathcal{A}}u_{\mathcal{B}}^{dir} \cdot {}^{\mathcal{A}}u_{\mathcal{B}}^{obs}\right)\right]} \\
{}^{\mathcal{A}}u_{\mathcal{B}}^{dir\Diamond obs} &= \frac{\left({}^{\mathcal{A}}u_{\mathcal{B}}^{dir} \cdot {}^{\mathcal{A}}u_{\mathcal{B}}^{obs}\right)}{\left[{}^{\mathcal{A}}u_{\mathcal{B}}^{dir} + {}^{\mathcal{A}}u_{\mathcal{B}}^{obs} - \left({}^{\mathcal{A}}u_{\mathcal{B}}^{dir} \cdot {}^{\mathcal{A}}u_{\mathcal{B}}^{obs}\right)\right]}
\end{aligned} \tag{9.7}$$

Case I: $\left[{}^{\mathcal{A}}u_{\mathcal{B}}^{dir} + {}^{\mathcal{A}}u_{\mathcal{B}}^{obs} - \left({}^{\mathcal{A}}u_{\mathcal{B}}^{dir} \cdot {}^{\mathcal{A}}u_{\mathcal{B}}^{obs}\right)\right] = 0$

$$\begin{aligned}
{}^{\mathcal{A}}b_{\mathcal{B}}^{dir\Diamond obs} &= \frac{\left(\gamma^{dir/obs} \cdot {}^{\mathcal{A}}b_{\mathcal{B}}^{dir} + {}^{\mathcal{A}}b_{\mathcal{B}}^{obs}\right)}{\left(\gamma^{dir/obs} + 1\right)} \\
{}^{\mathcal{A}}d_{\mathcal{B}}^{dir\Diamond obs} &= \frac{\left(\gamma^{dir/obs} \cdot {}^{\mathcal{A}}d_{\mathcal{B}}^{dir} + {}^{\mathcal{A}}d_{\mathcal{B}}^{obs}\right)}{\left(\gamma^{dir/obs} + 1\right)} \\
{}^{\mathcal{A}}u_{\mathcal{B}}^{dir\Diamond obs} &= 0; \quad \gamma^{dir/obs} = \lim \frac{{}^{\mathcal{A}}u_{\mathcal{B}}^{obs}}{{}^{\mathcal{A}}u_{\mathcal{B}}^{dir}}
\end{aligned} \tag{9.8}$$

$${}^{\mathcal{A}}\omega_{\mathcal{B}}^{glo} = \left[\left({}^{\mathcal{A}}\omega_{\mathcal{B}}^{dir} \oplus {}^{\mathcal{A}}\omega_{\mathcal{B}}^{obs}\right) \oplus {}^{\mathcal{A}}\omega_{\mathcal{B}}^{rec}\right] \tag{9.9}$$

9.3.2.4 Discounting Operator

Given that mobile nodes are not only deficit in evidence in the dynamically changing environment but also required to, at least partially, converge their opinions with other like-minded mobile nodes to make better decisions and harvest the available network services. For these reasons, the related models [3, 39, 64, 66, 68–71, 82, 83, 107, 112, 116, 125, 134, 137, 140, 169, 178, 190, 204, 214, 220–222, 233–235, 239, 241, 242, 252, 261, 270–273, 292, 294, 301–305, 308–311, 327, 335, 347, 372, 373, 379, 380, 401, 404, 424, 425, 461, 468] have designed the mobile nodes to exchange opinions with other model nodes in the form of recommendations. It is important to note that such an exchange of opinions can be valid only when the recommendations are transitive, and can be applied in environments where the context invariance can be assumed. Such transitivity holds true for the MANET, because the proposition (*i.e.* the trustworthiness of the mobile nodes) is evaluated within the context of the packet forwarding at the routing layer, which attributes a closed environment. Therefore, we inherit the discounting operator of the subjective logic that derives a single opinion from two other opinions, where the first opinion is held by a mobile node for a recommender and the second opinion is the recommendation communicated by the recommender towards a recommended node. Note that we would be using the terms, (a) *recommended opinion*, (b) *derived opinion* and, (c) *indirect opinion* interchangeably, and that they all refer the same in our model.

As shown in [185], the discounting operator derives a single opinion from two other opinions ($\omega_{\mathcal{S}}^{\mathcal{R}}$ and $\omega_{x}^{\mathcal{S}}$) that belong to different subjects ($\mathcal{R}$ and $\mathcal{S}$), respectively, and refer to different propositions unlike the consensus operator ($\oplus$). Let $\omega_{\mathcal{S}}^{\mathcal{R}} = \left(b_{\mathcal{S}}^{\mathcal{R}}, d_{\mathcal{S}}^{\mathcal{R}}, u_{\mathcal{S}}^{\mathcal{R}}\right)$ be $\mathcal{R}$'s referral opinion about $\mathcal{S}$, where the proposition translates into $\mathcal{S}$ *is knowledgeable and will tell the requested truth.* Alternatively, let $\omega_{x}^{\mathcal{S}} = \left(b_{x}^{\mathcal{S}}, d_{x}^{\mathcal{S}}, u_{x}^{\mathcal{S}}\right)$ be the $\mathcal{S}$'s functional opinion about the proposition x. Now, $\mathcal{R}$ can form an indirect functional opinion about the proposition x by discounting $\mathcal{S}$'s functional opinion about the proposition x with its referral opinion about $\mathcal{S}$. The difficulty involves defining the effect of $\mathcal{R}$ disbelieving that $\mathcal{S}$ will tell the truth. This is interpreted as if $\mathcal{R}$ believes that $\mathcal{S}$ is uncertain about the trust value of x regardless of $\mathcal{S}$'s actual recommendation. The derived opinion is denoted by $\omega_{x}^{\mathcal{R}:\mathcal{S}} = \left(b_{x}^{\mathcal{R}:\mathcal{S}}, d_{x}^{\mathcal{R}:\mathcal{S}}, u_{x}^{\mathcal{R}:\mathcal{S}}\right)$ and the operation is written as $\omega_{x}^{\mathcal{R}:\mathcal{S}} = \omega_{\mathcal{S}}^{\mathcal{R}} \otimes \omega_{x}^{\mathcal{S}}$ using $\otimes$.

$$\begin{aligned} b_{x}^{\mathcal{R}:\mathcal{S}} &= \left(b_{\mathcal{S}}^{\mathcal{R}} \cdot b_{x}^{\mathcal{S}}\right) \\ d_{x}^{\mathcal{R}:\mathcal{S}} &= \left(b_{\mathcal{S}}^{\mathcal{R}} \cdot d_{x}^{\mathcal{S}}\right) \\ u_{x}^{\mathcal{R}:\mathcal{S}} &= \left[d_{\mathcal{S}}^{\mathcal{R}} + u_{\mathcal{S}}^{\mathcal{R}} + \left(b_{\mathcal{S}}^{\mathcal{R}} \cdot u_{x}^{\mathcal{S}}\right)\right] \end{aligned} \tag{9.10}$$

Recall from Chapter 4 that the SMRTI-enabled mobile nodes never disseminate their opinions as recommendations to defend against the recommendation-related issues. Rather, the SMRTI-enabled mobile nodes derive an indirect or recommended opinion for a recommended node based on the recommendation inferred from a recommender's trust relationship with the recommended node, and their trust relationship with the recommender. In other words, the SL-SMRTI enabled mobile nodes derive a recommended opinion for a recommended node by performing the following, where they deduce, (a) *the recommender's functional opinion about the recommended node based on the global opinion inferred from the recommender's trust relationship with the recommended node* and, (b) *the referral opinion for the recommender from the global opinion held for the recommender.* Hence, the proposition attribute for both of these global opinions resolves into *the trustworthiness held by one mobile node for another*, except that the trustor and trustee roles are determined by the positions taken by the mobile nodes along the transitive path. We defer further explanation on the approach adopted by the SMRTI for inferring recommendation until §9.4.6 for the sake of continuity. Since $\otimes$ increases uncertainty along the transitive path, it also effectively complements the uncertainty involved in the approach adopted by the SL-SMRTI for deriving recommended opinions from inferred recommendations.

Assume that $\mathcal{A}$'s global opinion for $\mathcal{B}$ is given by ${}^{\mathcal{A}}\omega_{\mathcal{B}}^{glo} = \left({}^{\mathcal{A}}b_{\mathcal{B}}^{glo},\, {}^{\mathcal{A}}d_{\mathcal{B}}^{glo},\, {}^{\mathcal{A}}u_{\mathcal{B}}^{glo}\right)$ and the inferred $\mathcal{B}$'s global opinion for $\mathcal{C}$ is given by ${}^{\mathcal{B}}\omega_{\mathcal{C}}^{i\text{-}glo} = \left({}^{\mathcal{B}}b_{\mathcal{C}}^{i\text{-}glo},\, {}^{\mathcal{B}}d_{\mathcal{C}}^{i\text{-}glo},\, {}^{\mathcal{B}}u_{\mathcal{C}}^{i\text{-}glo}\right)$. Here, $\mathcal{A}$ derives its recommended opinion for $\mathcal{C}$ by following its global opinion for $\mathcal{B}$ and inferred $\mathcal{B}$'s global opinion for $\mathcal{C}$. Consequently, $\mathcal{A}$'s recommended opinion for $\mathcal{C}$ is given by ${}^{\mathcal{A}}\omega_{\mathcal{C}}^{\mathcal{B}-rec} = \left({}^{\mathcal{A}}b_{\mathcal{C}}^{\mathcal{B}-rec},\, {}^{\mathcal{A}}d_{\mathcal{C}}^{\mathcal{B}-rec},\, {}^{\mathcal{A}}u_{\mathcal{C}}^{\mathcal{B}-rec}\right)$, and the operation is denoted as ${}^{\mathcal{A}}\omega_{\mathcal{C}}^{\mathcal{B}-rec} = {}^{\mathcal{A}}\omega_{\mathcal{B}}^{glo} \otimes^{\mathcal{B}} \omega_{\mathcal{C}}^{i-glo}$.

$$\begin{aligned}
{}^{\mathcal{A}}b_{\mathcal{C}}^{\mathcal{B}-rec} &= \left({}^{\mathcal{A}}b_{\mathcal{B}}^{glo} \cdot {}^{\mathcal{B}}b_{\mathcal{C}}^{i-glo}\right) \\
{}^{\mathcal{A}}d_{\mathcal{C}}^{\mathcal{B}-rec} &= \left({}^{\mathcal{A}}b_{\mathcal{B}}^{glo} \cdot {}^{\mathcal{B}}d_{\mathcal{C}}^{i-glo}\right) \\
{}^{\mathcal{A}}u_{\mathcal{C}}^{\mathcal{B}-rec} &= \left[{}^{\mathcal{A}}d_{\mathcal{B}}^{glo} + {}^{\mathcal{A}}u_{\mathcal{B}}^{glo} + \left({}^{\mathcal{A}}b_{\mathcal{B}}^{glo} \cdot {}^{\mathcal{B}}u_{\mathcal{C}}^{i-glo}\right)\right]
\end{aligned} \tag{9.11}$$

Equation (9.11) presents the case where $\mathcal{A}$ only infers $\mathcal{B}$'s recommendation for its recommended opinion ${}^{\mathcal{A}}\omega_{\mathcal{C}}^{\mathcal{B}-rec}$ towards $\mathcal{C}$. Therefore, it is apparent that $\mathcal{A}$ would also infer furthermore more recommendations for $\mathcal{C}$ from mobile nodes other than $\mathcal{B}$, which is not expressed in (9.11). In such a scenario, $\mathcal{A}$ would 'consensus' the recommended opinion that has been formulated from the latest inferred recommendation with the *overall recommended opinion* ${}^{\mathcal{A}}\omega_{\mathcal{C}}^{rec}$ held for $\mathcal{C}$. Here, the overall recommended opinion ${}^{\mathcal{A}}\omega_{\mathcal{C}}^{rec}$ expresses the consensus of all past recommended opinions derived for $\mathcal{C}$, which is initialised to uncertainty at the time of the deployment, *i.e.* ${}^{\mathcal{A}}\omega_{\mathcal{C}}^{rec(t_0)} = (0.0, 0.0, 1.0)$. Unless required, ${}^{\mathcal{A}}\omega_{\mathcal{C}}^{rec(t_i)}$ is denoted as ${}^{\mathcal{A}}\omega_{\mathcal{C}}^{rec}$ for simplicity.

If we assume the recommended opinion ${}^{\mathcal{A}}\omega_{\mathcal{C}}^{\mathcal{B}-rec}$ that was derived in (9.11) to be the foremost recommended opinion that has been derived for $\mathcal{C}$ since the deployment, then the previously-mentioned consensus operation would appear as follows,

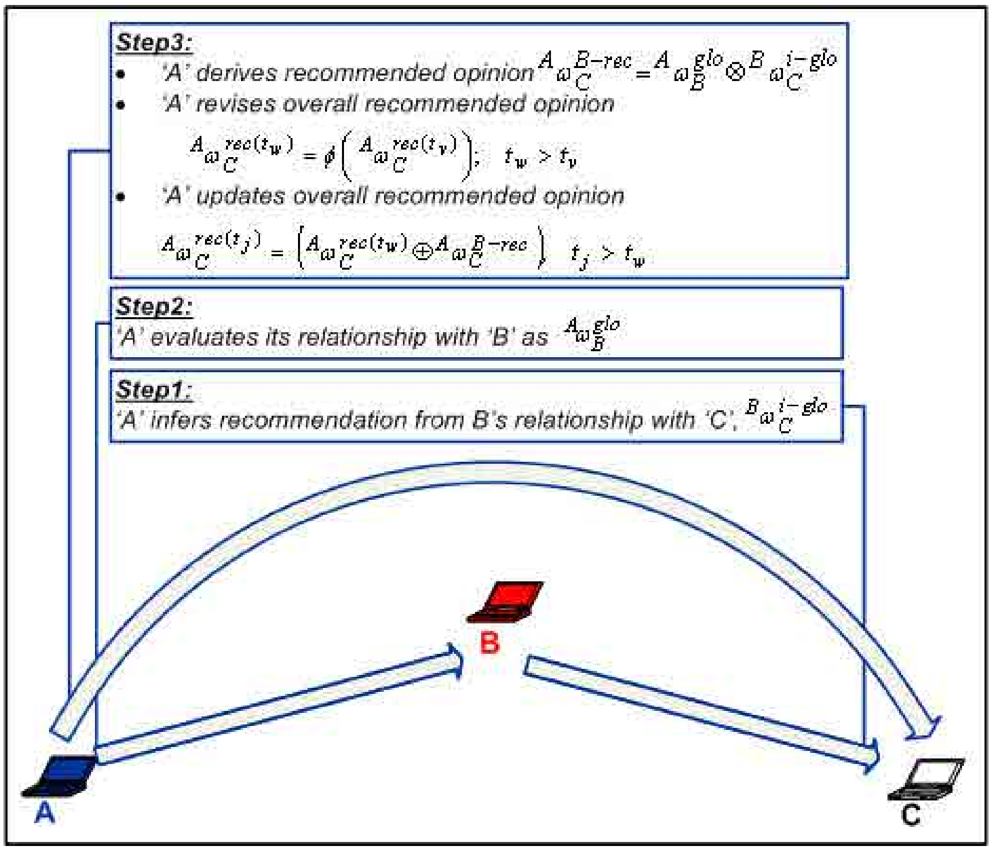

Step3:
'A' derives recommended opinion ${}^{A}\omega_{C}^{B-rec} = {}^{A}\omega_{B}^{glo} \otimes {}^{B}\omega_{C}^{i-glo}$
'A' revises overall recommended opinion
${}^{A}\omega_{C}^{rec(t_w)} = \phi\left({}^{A}\omega_{C}^{rec(t_v)}\right); \quad t_w > t_v$
'A' updates overall recommended opinion
${}^{A}\omega_{C}^{rec(t_j)} = \left({}^{A}\omega_{C}^{rec(t_w)} \oplus {}^{A}\omega_{C}^{B-rec}\right) \quad t_j > t_w$
Step2:
'A' evaluates its relationship with 'B' as ${}^{A}\omega_{B}^{glo}$
Step1:
'A' infers recommendation from B's relationship with 'C', ${}^{B}\omega_{C}^{i-glo}$
B
A
C

Figure 9.2: Formulation of Overall Recommended Opinion.

$$ {}^{\mathcal{A}}\omega_{\mathcal{C}}^{rec(t_j)} = \left({}^{\mathcal{A}}\omega_{\mathcal{C}}^{rec(t_0)} \oplus^{\mathcal{A}} \omega_{\mathcal{C}}^{\mathcal{B}-rec}\right); \quad t_j > t_0 \tag{9.12} $$

In summary, whenever $\mathcal{A}$ derives a recommend opinion for $\mathcal{C}$ by following (9.11), it is subsequently expected to execute (9.12). However, it never happens for the following reason. In comparison with the evidence-to-opinion mapping operator (∇) which is given in (9.4) and used to obtain direct and observed opinions, it can be noticed that the discounting operator ($\otimes$) fails to account for the ignorance introduced during the absence of recommendations. For this purpose, we propose a new operator known as the fading operator ($\oslash$), which is explained in detail below and expressed in (9.13). Therefore, for every inferred recommendation, $\mathcal{A}$ would execute the operations that are given by the following sequence of equations, (9.11), (9.13), and (9.12).

9.3.2.4.1 Fading Operator Similar to direct and observed opinions, mobile nodes must be capable of expressing their ignorance during a deficit of recommendations. It is necessary for mobile nodes to represent their trust relationship more precisely with other mobile nodes. The fading operator ($\oslash$) enables the mobile nodes to express ignorance in their overall recommended opinion for other nodes whenever there is a deficit of recommendation. It accomplishes this objective by exponentially increasing the uncertainty (${}^{\mathcal{A}}u_{\mathcal{C}}^{rec}$), and at the same time exponentially decreasing both the belief (${}^{\mathcal{A}}b_{\mathcal{C}}^{rec}$) and disbelief (${}^{\mathcal{A}}d_{\mathcal{C}}^{rec}$) components of the overall recommended opinion (${}^{\mathcal{A}}\omega_{\mathcal{C}}^{rec}$) in proportion to the duration for which there has been a deficit of recommendations. The operation still satisfies the conditions in (9.1) and can be confirmed in (9.13). This remains in-line with the reasoning that not only does belief reduce in the absence of positive recommendations, but also the disbelief when only positive recommendations are inferred.

Let us consider the previous scenario, where $\mathcal{A}$'s overall recommended opinion for $\mathcal{C}$ at time t_v is given by, ${}^{\mathcal{A}}\omega_{\mathcal{C}}^{rec(t_v)} = ({}^{\mathcal{A}}b_{\mathcal{C}}^{rec(t_v)}, {}^{\mathcal{A}}d_{\mathcal{C}}^{rec(t_v)}, {}^{\mathcal{A}}u_{\mathcal{C}}^{rec(t_v)})$. Let l represent the count of intervals for which recommendation is unavailable to update ${}^{\mathcal{A}}\omega_{\mathcal{C}}^{rec(t_v)}$, where an interval is equal to the average duration taken for a communication flow. Note that the interval is defined from the perspective of $\mathcal{A}$. Equation (9.13) denotes the operation of applying $\oslash$ to ${}^{\mathcal{A}}\omega_{\mathcal{C}}^{rec(t_v)}$ at time t_w to arrive at the *revised overall recommended opinion* ${}^{\mathcal{A}}\omega_{\mathcal{C}}^{rec(t_w)}$, and (9.14) presents the details of (9.13).

$$ {}^{\mathcal{A}}\omega_{\mathcal{C}}^{rec(t_w)} = \oslash \left({}^{\mathcal{A}}\omega_{\mathcal{C}}^{rec(t_v)} \right); \quad t_w > t_v \tag{9.13} $$

$$ \begin{aligned} {}^{\mathcal{A}}b_{\mathcal{C}}^{rec(t_w)} &= \left[{}^{\mathcal{A}}b_{\mathcal{C}}^{rec(t_v)} \cdot \left(1 - e^{-l}\right) \right] \\ {}^{\mathcal{A}}d_{\mathcal{C}}^{rec(t_w)} &= \left[{}^{\mathcal{A}}d_{\mathcal{C}}^{rec(t_v)} \cdot \left(1 - e^{-l}\right) \right] \\ {}^{\mathcal{A}}u_{\mathcal{C}}^{rec(t_w)} &= \left[{}^{\mathcal{A}}u_{\mathcal{C}}^{rec(t_v)} + \left| \begin{matrix} \left({}^{\mathcal{A}}b_{\mathcal{C}}^{rec(t_w)} + {}^{\mathcal{A}}d_{\mathcal{C}}^{rec(t_w)}\right) - \\ \left({}^{\mathcal{A}}b_{\mathcal{C}}^{rec(t_v)} + {}^{\mathcal{A}}d_{\mathcal{C}}^{rec(t_v)}\right) \cdot \left(1 - e^{-l}\right) \end{matrix} \right| \right] \end{aligned} \tag{9.14} $$

All the above details are summarised in Figure 9.2, where $\mathcal{A}$ is shown to derive its recommended opinion for $\mathcal{C}$, ${}^{\mathcal{A}}\omega_{\mathcal{C}}^{\mathcal{B}-rec}$ based on $\mathcal{B}$'s recommendation by using $\otimes$ as given in (9.11). Figure 9.2 then presents the operation, where $\mathcal{A}$ revises its overall recommended opinion for $\mathcal{C}$, ${}^{\mathcal{A}}\omega_{\mathcal{C}}^{rec}$ using $\oslash$ as given by (9.13). Finally, $\mathcal{A}$ is shown integrating the recommended opinion that was derived from $\mathcal{B}$'s recommendation with its revised overall recommended opinion for $\mathcal{C}$ using $\oplus$ as given by (9.12).

9.3.2.5 Comparison Operator

One of the main objectives of a trust model is to make decisions for a system using the defined policies and held opinions or established trust relationships. The related models [38, 64–71, 125, 220, 301–305, 308, 309, 311, 347, 373, 401] evaluate trust relationships by evaluating the opinions against a pre-defined threshold. We follow this conventional approach and, for this reason, we propose another new operator known as the *comparison operator* ($\geqslant$) to our adapted subjective logic. Therefore, $\geqslant$ aids mobile nodes to evaluate their trust relationship with other nodes or context-oriented events as trustworthy, uncertain or untrustworthy using the global opinion held for those nodes or the global opinions consented for those context-oriented events respectively.

Assume that $\mathcal{A}$'s global opinion for $\mathcal{B}$ is denoted by ${}^{\mathcal{A}}\omega_{\mathcal{B}}^{glo} = ({}^{\mathcal{A}}b_{\mathcal{B}}^{glo}, {}^{\mathcal{A}}d_{\mathcal{B}}^{glo}, {}^{\mathcal{A}}u_{\mathcal{B}}^{glo})$. On applying $\geqslant$ to ${}^{\mathcal{A}}\omega_{\mathcal{B}}^{glo}$, node $\mathcal{A}$ elucidates its trust relationship with $\mathcal{B}$ as trustworthy, uncertain or untrustworthy. Thus we get ${}^{\mathcal{A}}\omega_{\mathcal{B}}^{glo} \geqslant \mathit{Threshold}$ and the threshold values are:

$$
\begin{aligned}
&[\mathcal{A} \text{ trusts } \mathcal{B}] \Leftrightarrow \\
&\qquad \Big[\big({}^{\mathcal{A}}b_{\mathcal{B}}^{glo} \geqslant \mathit{Threshold}\big) \wedge \big({}^{\mathcal{A}}d_{\mathcal{B}}^{glo} < \mathit{Threshold}\big) \wedge (0.5 < \mathit{Threshold})\Big] \\
&[\mathcal{A} \text{ distrusts } \mathcal{B}] \Leftrightarrow \\
&\qquad \Big[\big({}^{\mathcal{A}}d_{\mathcal{B}}^{glo} \geqslant \mathit{Threshold}\big) \wedge \big({}^{\mathcal{A}}u_{\mathcal{B}}^{glo} < \mathit{Threshold}\big) \wedge (0.5 < \mathit{Threshold})\Big] \\
&[\mathcal{A} \text{ uncertain of } \mathcal{B}] \Leftrightarrow \Big[\text{for remaining values of } {}^{\mathcal{A}}b_{\mathcal{B}}^{glo},\ {}^{\mathcal{A}}d_{\mathcal{B}}^{glo},\ \mathit{and}\, {}^{\mathcal{A}}u_{\mathcal{B}}^{glo}\Big]
\end{aligned}
\tag{9.15}
$$

9.4 Architecture of SL-SMRTI

We begin this section with the formalisation of our SL-SMRTI. We then briefly introduce the working of the AODV and the AOMDV to introduce the context for the operation of the SL-SMRTI with the AOMDV. We then detail how the AOMDV is adapted and tightly coupled with the SL-SMRTI for enhancing the routing decisions, and also on how the SL-SMRTI assists the AOMDV in making those trust-enhanced routing decisions. Finally, we describe the process of capturing evidence for different types of opinion (direct, observed and recommended), which are evaluated by the SL-SMRTI to make the trust-enhanced decisions.

9.4.1 Formal Representation of SL-SMRTI

Recollect from Chapter 4 that the SMRTI is a decentralised independent component in every mobile node, where it operates along with an inbuilt IDS monitor mechanism to collect the evidence of trustworthiness (direct, observed and recommended) that is captured from the behaviour of other nodes. It then enables the mobile node to formulate an opinion (direct, observed and recommended) for those nodes for which the evidence was captured. Finally, the SMRTI assists the mobile node to make subjective routing decisions to enhance the security of the communication. On the other hand, recall from §9.3 that subjective logic provides a strong mathematical foundation for the SMRTI to represent the notion of ignorance in the established trust relationships. Therefore, the resulting SL-SMRTI at any mobile node $\mathcal{I}$ is represented using the following structure, $SL - SMRTI_{\mathcal{I}} = \{n_{\mathcal{I}}, \mathbb{R}_{\mathcal{I}}\}$. If n is the set of all nodes in the network, then $n_{\mathcal{I}}$ is the set of all nodes that exist in the network excluding $\mathcal{I}$ such that $n_{\mathcal{I}} = n - \{\mathcal{I}\}$. Finally, $\mathbb{R}_{\mathcal{I}}$ represents the set of trust relationships that are held between $\mathcal{I}$ and the list of nodes in $n_{\mathcal{I}}$. The following expression presents the trust relationship $\mathbb{R}_{\mathcal{I}\mathcal{J}}$ between the nodes $\mathcal{I}$ and $\mathcal{J}$, which is given as a tuple with eight attributes:

$$\mathbb{R}_{\mathcal{I}\mathcal{J}} = \{\mathsf{T}, \Omega, \mathsf{E}, \mathsf{P}, \mathsf{N}, \sigma, \tau, \delta\}$$

In the above expression, the node $\mathcal{I}$'s trust T for node $\mathcal{J}$ is given by its global opinion ${}^{\mathcal{I}}\omega_{\mathcal{J}}^{glo}$ for $\mathcal{J}$. Recall that ${}^{\mathcal{I}}\omega_{\mathcal{J}}^{glo}$ is derived from (9.9), and Ω contains the sub-opinions (direct, observed and recommended) held for $\mathcal{J}$ based on the evidence collected through the corresponding modes (direct, observed and recommended). Parameter E denotes the set of events (route request, route reply, route error and data flow) from which the evidence for benign and malicious behaviours is captured. The sets P and N contain the respective positive *(p)* and negative *(n)* evidence for both direct (${}^{\mathcal{I}}\omega_{\mathcal{J}}^{dir}$) and the observed (${}^{\mathcal{I}}\omega_{\mathcal{J}}^{obs}$) opinions. The timestamp set (σ) indicates the period when the opinions (direct, observed and recommended) were initialised or last revised; hence, σ has a bijective mapping with Ω. The average duration taken for a communication flow by $\mathcal{I}$ is given by τ. Recall that τ is used to calculate the number of intervals for which there has been no update for each type of opinion (direct, observed and recommended). Finally, δ refers to $\langle \delta_n, \delta_t \rangle$, where δ_n denotes the name of cluster, and δ_t represents the type of relationship in cluster-based MANET, *i.e.* whether it is an intra-cluster or inter-cluster relationship. In the MANETs where a cluster structure is not adopted, each node is treated as an individual cluster for simplicity.

9.4.2 System Operation

9.4.2.1 AODV

The AODV incorporates the notion of a destination sequence numbers as in the DSDV, and an on-demand route discovery as in the DSR, but relies on the hop-based routing for discovering routes between the source and destination mobile nodes. Whenever a source mobile node fails to find a valid route to the destination mobile node, it calls upon the AODV to initiate a route discovery. For simplicity, the source mobile node that initiates the route discovery is interchangeably referred to as the *originator* and the destination mobile node as the target. The controlled route request (RREQ) flooding is the first step in the route discovery process, in which the address and the sequence number of the originator together with the RREQ ID are used by the intermediate nodes to detect and discard duplicates. On identifying non-duplicates, the intermediate nodes establish a reverse route to the originator using the previous-hop (from which it had received the RREQ) as the next-hop to the originator along the reverse route. In addition, if any intermediate node has a route to the target, then the intermediate node generates a gratuitous route reply (RREP) on behalf of the target and unicasts it to the originator along the reverse route. In the situation where the intermediate nodes lack a route to the target, the hop-count field of the non-duplicate RREQ is incremented by them before re-broadcasting the RREQ.

Once the target receives the RREQ, it crafts and unicasts an RREP to the originator along the reverse route. Note that the previous-hop (from which the target had received the RREQ) would be the next-hop towards the originator along the reverse route. Since the RREP propagates towards the originator, all intermediate nodes establish the forward route to the target (*i.e.* to the source of RREP) by using the previous-hop (from which each of them had received the RREP) as the next-hop towards the target along the forward route. On receiving the RREP, the originator executes the same steps to establish the forward route to the target.

The AODV employs a soft state timer called the *route expiration timeout* to purge the expired routes. Alternatively, if a mobile node detects a link failure, it purges the routes leading to the various destinations through the invalid link, and performs the route maintenance operation using the route error (RERR) packet. The RERR contains the list of unreachable destinations and their corresponding sequence numbers propagated along the upstream routes via the previous-hop neighbours. On receiving the RERR, the upstream nodes invalidate the routes that correspond to the unreachable destinations and update the sequence number of those destinations for which they have outdated values.

The AODV has some significant advantages over the DSR by supporting multicast through the construction of the trees and the reduction of the network bandwidth overhead. In addition, it is scalable owing to the usage of the hop count; however it suffers in the presence of asymmetric links and fails to use multi-paths. Several extensions and optimisations have been proposed to fix those limitations and to enhance the AODV, including the discovery of a bidirectional path in the presence of unidirectional links [43], detecting multi-paths over bidirectional links [244], containing the scope of flooding [85] and reducing the redundant broadcasts [275]. In the following section, we will be presenting one such extension of the AODV, known as the AOMDV [244], that is capable of computing multiple loop-free paths per route discovery. We focus on only the AOMDV, because the other extensions and optimisations are orthogonal to our objective.

9.4.2.2 AOMDV

The AOMDV adds value to the AODV by allowing the mobile nodes to switch over to alternate routes during the failure of the primary route and hence, enables those mobile nodes to avoid a new route discovery. The AOMDV discovers the link disjoint paths so that a route discovery can be initiated only if all paths independently fail each other. Note that the link disjoint paths are different from the node disjoint paths that are used for load balancing, while the former is adopted for reducing the routing overheads in multi-path routing. Therefore, the link disjoint paths may contain common nodes among the discovered multiple routes.

The AOMDV incorporates several changes to the AODV to deliver multiple routes; one among those changes is to enable the intermediate nodes to examine the duplicate RREQs that are supposed to be discarded in accordance with the specification of the AODV. They examine the duplicate RREQs to determine whether those duplicates are received via the disjoint paths. For this reason, the AOMDV adds the address of the *first-hop* taken by the RREQ as an additional field to the RREQ. This is because all trajectories of the RREQs between any pair of nodes with unique first-hops are guaranteed to be disjoint [244]. The intermediate nodes establish reverse paths for those RREQs that have a unique first-hop field. Note that the reverse routes established at the intermediate nodes are node disjoint paths and are not yet link disjoint paths. Consequently, the intermediate nodes propagate only the first copy of the RREQ as in the AODV, thereby eliminating the additional overhead.

For example, in Figure 9.3, although node $\mathcal{G}$ receives two duplicate RREQs from the originator $\mathcal{S}$ along the routes, $\mathcal{S} \mapsto \mathcal{B} \mapsto \mathcal{E} \mapsto \mathcal{G}$ and $\mathcal{S} \mapsto \mathcal{B} \mapsto \mathcal{F} \mapsto \mathcal{G}$, it only

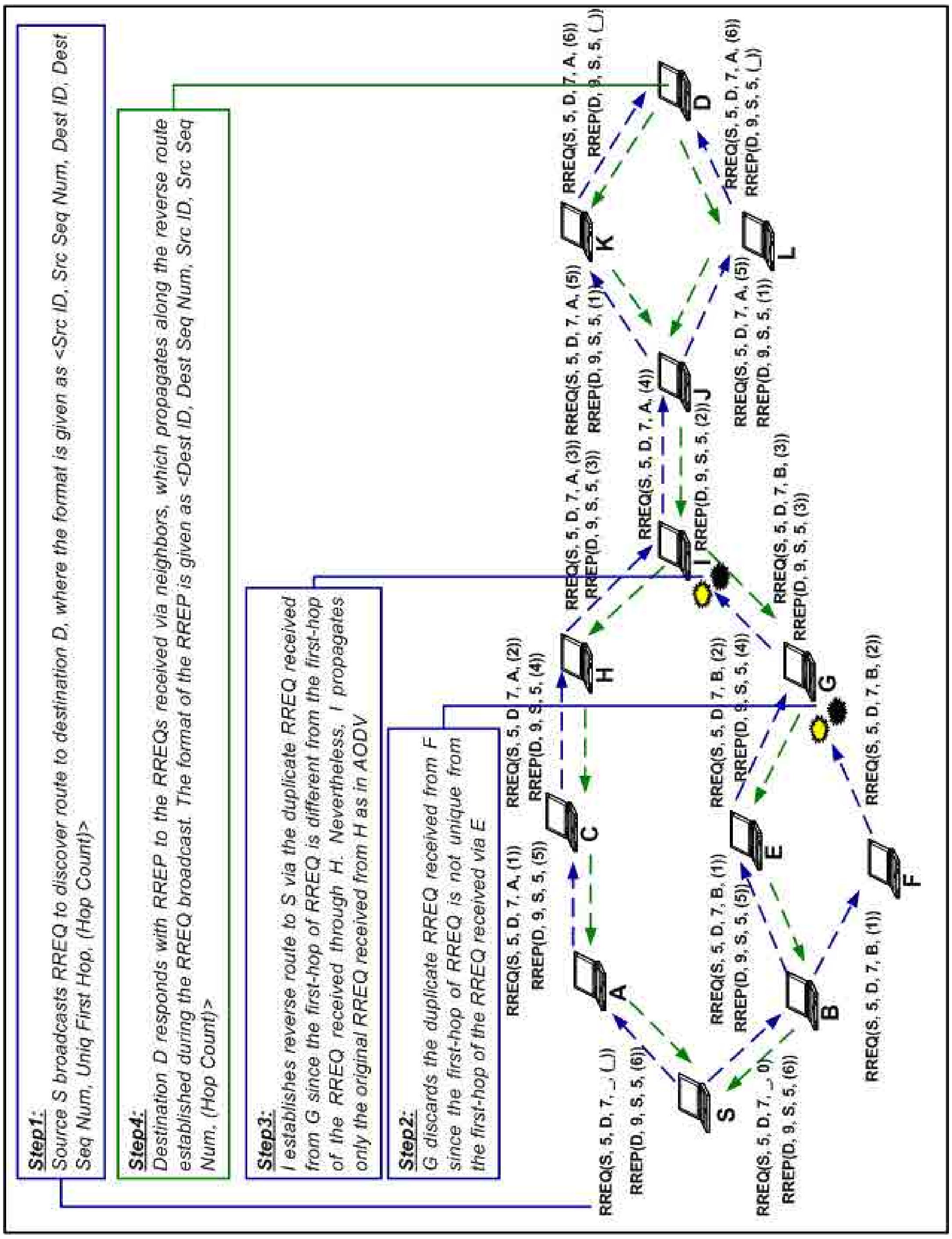

Step1:
Source S broadcasts RREQ to discover route to destination D, where the format is given as <Src ID, Src Seq Num, Dest ID, Dest Seq Num, Uniq First Hop, (Hop Count)>
Step4:
Destination D responds with RREP to the RREQs received via neighbors, which propagates along the reverse route established during the RREQ broadcast. The format of the RREP is given as <Dest ID, Dest Seq Num, Src ID, Src Seq Num, (Hop Count)>
Step3:
I establishes reverse route to S via the duplicate RREQ received from G since the first-hop of RREQ is different from the first-hop of the RREQ received through H. Nevertheless, I propagates only the original RREQ received from H as in AODV
Step2:
G discards the duplicate RREQ received from F since the first-hop of RREQ is not unique from the first-hop of the RREQ received via E
S
A
B
C
E
F
G
H
I
J
K
L
D
RREQ(S, 5, D, 7, _ (_))
RREP(D, 9, S, 5, (6))
RREQ(S, 5, D, 7, _ 0)
RREP(D, 9, S, 5, (6))
RREQ(S, 5, D, 7, A, (1))
RREP(D, 9, S, 5, (5))
RREQ(S, 5, D, 7, A, (2))
RREP(D, 9, S, 5, (4))
RREQ(S, 5, D, 7, B, (1))
RREP(D, 9, S, 5, (5))
RREQ(S, 5, D, 7, B, (2))
RREP(D, 9, S, 5, (4))
RREQ(S, 5, D, 7, B, (1))
RREQ(S, 5, D, 7, B, (2))
RREQ(S, 5, D, 7, A, (3))
RREP(D, 9, S, 5, (3))
RREQ(S, 5, D, 7, B, (3))
RREP(D, 9, S, 5, (3))
RREQ(S, 5, D, 7, A, (4))
RREP(D, 9, S, 5, (2))
RREQ(S, 5, D, 7, A, (5))
RREP(D, 9, S, 5, (1))
RREQ(S, 5, D, 7, A, (5))
RREP(D, 9, S, 5, (1))
RREQ(S, 5, D, 7, A, (6))
RREP(D, 9, S, 5, (_))
RREQ(S, 5, D, 7, A, (6))
RREP(D, 9, S, 5, (_))

Figure 9.3: Route Discovery in AOMDV.

propagates the first seen RREQ (*i.e.* from $\mathcal{E}$). This is because the duplicates contain the same first-hop field. Alternatively, node $\mathcal{I}$ propagates duplicate RREQs that were received from $\mathcal{S}$ via the routes, $\mathcal{S} \mapsto \mathcal{A} \mapsto \mathcal{C} \mapsto \mathcal{H} \mapsto \mathcal{I}$ and $\mathcal{S} \mapsto \mathcal{B} \mapsto \mathcal{E} \mapsto \mathcal{G} \mapsto \mathcal{I}$ because of the unique first-hop field. It then establishes the reverse routes to $\mathcal{S}$ via the previous-hops $\mathcal{H}$ and $\mathcal{G}$, through which the duplicate RREQs were received.

The target introduces link disjoint paths by forwarding a copy of the RREP to each of its unique neighbours (from which the RREQ was received) regardless of the count of the received duplicate RREQs. This confirms that the first-hop field of the RREQ is irrelevant from the perspective of the target. On the other hand, the dispatched RREPs follow the node disjoint reverse routes (that were established at the intermediate nodes on the receipt of RREQs) after leaving the target's neighbours. Even though the design explained thus far assures a loop-free AOMDV, the optional feature to allow the intermediate nodes to respond with a gratuitous RREP would destroy such a loop-free assurance in the AOMDV. This is because the intermediate nodes may have multiple routes to respond to the RREQs. For this reason, the AOMDV replaces the, (a) *next-hop field of the AODV with a route list that contains multiple routes, where each of those routes is given by the next-hop and hop-count* and, (b) *hop-count field of the AODV with an advertised hop-count field that can hold the maximum of the hop-counts from the route list.* Finally, the AOMDV calls for the route maintenance (RERR) only when all the routes to a destination become invalid.

Figure 9.3 displays the scenario where node $\mathcal{D}$ dispatches the RREPs via its unique neighbours $\mathcal{K}$ and $\mathcal{L}$. The RREPs next follow the reverse routes ($\mathcal{D} \mapsto \mathcal{K} \mapsto \mathcal{J} \mapsto \mathcal{I} \mapsto \mathcal{H} \mapsto \mathcal{C} \mapsto \mathcal{A} \mapsto \mathcal{S}$, and $\mathcal{D} \mapsto \mathcal{L} \mapsto \mathcal{J} \mapsto \mathcal{I} \mapsto \mathcal{G} \mapsto \mathcal{E} \mapsto \mathcal{B} \mapsto \mathcal{S}$) that were recorded earlier during the propagation of the RREQs. In the following, we first detail the interface between the SL-SMRTI and the AOMDV, and then the approach adopted to enhance the routing decisions.

9.4.2.3 AOMDV and SL-SMRTI

The SL-SMRTI assists the AOMDV protocol in making decisions in the following cases, (a) *whether to record or discard a route that is given by a backward-link (previous-hop)* *towards the originator*, (b) *whether to accept or reject a route that is given by a forward-link (next-hop) towards the target*, (c) *which route to choose from available routes for a communication*, (d) *whether to forward packets on the behalf of an originator* and, (e) *whether to participate as an intermediate node for the packets flowing towards the target.* The decision for each of the above cases rests on the result of their corresponding trust evaluation. In turn, the trust evaluation relies on the opinions (direct, observed and recommended) held for one or more nodes that are involved in the evaluated case.

Let us consider the scenario shown in Figure 9.3, where $\mathcal{S}$ is the originator and $\mathcal{D}$ is the target for the communication flow. Nodes $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, $\mathcal{E}$, $\mathcal{G}$, $\mathcal{H}$, $\mathcal{I}$, $\mathcal{J}$, $\mathcal{K}$, and $\mathcal{L}$ are the intermediary nodes that form the multiple routes between $\mathcal{S}$ and $\mathcal{D}$. Whenever $\mathcal{S}$ wishes to send data packets to $\mathcal{D}$, it initially checks for a route or a forward-link (next-hop) to $\mathcal{D}$. The instance of the SL-SMRTI at $\mathcal{S}$ evaluates the trustworthiness of the route, provided $\mathcal{S}$'s AOMDV finds a route to $\mathcal{D}$ from the routing cache. The SL-SMRTI performs such evaluation prior to the deployment of every route, regardless that the trustworthiness of all routes was evaluated before allowing the AOMDV to record those routes. The reasoning is because the trust is not monotonic; and henceforth the trustworthiness of a route may change anytime between its point of entry and deployment. Given that there is more than one route to $\mathcal{D}$ and most of them are trustworthy, the route with the highest trust value is chosen for the communication flow. If two routes exist with the same trust value, then the route with smallest hop-count is selected. For instances where the hop-counts are the same, one of the routes is arbitrarily chosen for the communication. If the trust evaluation asserts that all the available routes attribute to an uncertain value, then the route or forward-link (next-hop) with the least uncertainty is chosen for the communication with $\mathcal{D}$. Note that such a scenario would arise during the initial stages of the network deployment, because nodes are neither trusted nor distrusted. Alternatively, if only untrustworthy routes exist or there is no route available to $\mathcal{D}$, then $\mathcal{S}$ would initiate a new route discovery to $\mathcal{D}$. Prior to the initiation, $\mathcal{S}$ would also purge all the untrustworthy routes from its route cache.

In our trust model, the intermediary nodes are designed to perform the following trust evaluations. Let us consider the operation at node $\mathcal{I}$ in Figure 9.3. The node evaluates the trustworthiness of the previous-hop neighbours before setting them as the backward-link along the reverse routes to $\mathcal{S}$. This is in accordance with the expectation that the route is likely to be free from modification, only if the previous-hop neighbour is trustworthy. Also, $\mathcal{I}$ evaluates the trustworthiness of the next-hop neighbours before setting them as the forward-link to $\mathcal{D}$. This is in accordance with the expectation that the route will probably be free from modification and the corresponding data packets will likely reach $\mathcal{D}$, only if the next-hop neighbour is trustworthy. Finally, $\mathcal{I}$ evaluates the trustworthiness of a packet by evaluating the trustworthiness of $\mathcal{S}$ (the originator) and $\mathcal{D}$ (the target). The reasoning is that the intermediate nodes forward packets only for the sake of the source and destination mobile nodes. The trust evaluations of a route (forward-link or backward-link) and packet are described in the following §9.4.3. The intermediate node $\mathcal{I}$ exercises an exception when the above set of evaluations has to be adopted for the route request event, for which the next-hop neighbour is unknown because of the broadcast nature. Node $\mathcal{I}$ only evaluates the trustworthiness of the previous-hop and packet (*i.e.* source and destination) and excludes the unknown next-hop neighbour from the evaluation. The trust evaluations are deemed successful, only if the evaluated opinion is asserted as either trustworthy or uncertain.

In the case of $\mathcal{D}$, the previous-hops are accepted as backward-links to $\mathcal{S}$, only after evaluating their trustworthiness. Otherwise, $\mathcal{D}$ initiates a new route discovery to $\mathcal{S}$.

9.4.3 Trust Evaluation

Let us assume that every node had set its opinion of type m (direct, observed and recommended) for all other nodes to the default uncertain value, ${}^{\mathcal{I}}\omega^{m}_{\mathcal{J}}$, during the initial stages of network deployment.

Let us now consider the trust evaluation at some node $\mathcal{I}$ for another node $\mathcal{J}$, where $\mathcal{J}$ acts as $\mathcal{I}$'s forward-link or backward-link (route) towards the target or originator respectively. Node $\mathcal{I}$ evaluates the trustworthiness of $\mathcal{J}$ in two steps. As shown in (9.9), the $\oplus$ operator is used by $\mathcal{I}$ to combine its opinions (direct, observed and recommended) held for $\mathcal{J}$ into a global opinion (${}^{\mathcal{I}}\omega^{glo}_{\mathcal{J}}$). Prior to the derivation, the direct (${}^{\mathcal{I}}\omega_{\mathcal{J}}^{dir}$) and observed (${}^{\mathcal{I}}\omega_{\mathcal{J}}^{obs}$), and recommended (${}^{\mathcal{I}}\omega_{\mathcal{J}}^{rec}$) opinions are updated using ∇ (9.4) and $\oslash$ (9.13) operators respectively, in proportion to the duration for which there was no evidence. Operator $\geqslant$ from (9.15) is then applied to $\mathcal{I}$'s global opinion ${}^{\mathcal{I}}\omega_{\mathcal{J}}^{glo}$ to evaluate the type of trust relationship (trustworthy, untrustworthy and uncertainty) it holds with $\mathcal{J}$. If the evaluation advocates to either a trustworthy or uncertain relationship, then $\mathcal{J}$ is recorded as the forward or backward-link (route) to the target or originator, respectively. Otherwise, the route is discarded together with the packet. As mentioned earlier, a route pertaining to an uncertain relationship is recorded to enable the establishment of a trust relationship during the initial stages of deployment and to meet the conditions when trustworthy routes are unavailable.

Similarly, $\mathcal{I}$ evaluates the trustworthiness of a packet *pkt* in two steps. Initially, $\mathcal{I}$ follows the $\oplus$ operator as given in (9.9) to compute its global opinions, ${}^{\mathcal{I}}\omega_{\mathcal{S}}^{glo}$ and ${}^{\mathcal{I}}\omega_{\mathcal{D}}^{glo}$ for the source and destination of the packet, respectively. Recall that $\mathcal{I}$ updates its direct and observed opinions using the ∇ (9.4) and recommended opinion using the $\oslash$ (9.13), for both $\mathcal{S}$ and $\mathcal{D}$, in proportion to the duration for which there has been no evidence. Node $\mathcal{I}$ then computes its opinion ${}^{\mathcal{I}}\omega_{pkt}^{glo}$ for *pkt* by applying the consensus operator ($\oplus$) to the global opinions ${}^{\mathcal{I}}\omega_{\mathcal{S}}^{glo}$ and ${}^{\mathcal{I}}\omega_{\mathcal{D}}^{glo}$ held for $\mathcal{S}$ and $\mathcal{D}$, respectively; the operation is given below in (9.16). Finally, $\mathcal{I}$ evaluates the trustworthiness of *pkt* by applying the $\geqslant$ operator to ${}^{\mathcal{I}}\omega_{pkt}^{glo}$. Node $\mathcal{I}$ forwards *pkt* only if the evaluation results in either a trustworthy or an uncertain relationship.

$$ {}^{\mathcal{I}}\omega_{pkt}^{glo} = \left({}^{\mathcal{I}}\omega_{\mathcal{S}}^{glo} \oplus^{\mathcal{I}} \omega_{\mathcal{D}}^{glo} \right) \tag{9.16} $$

9.4.4 Direct Opinion

We define node $\mathcal{I}$'s direct opinion (${}^{\mathcal{I}}\omega_{\mathcal{J}}^{dir}$) towards node $\mathcal{J}$, as its trust on $\mathcal{J}$ depending on the evidence collected from the one-to-one interactions with $\mathcal{J}$. The evidence is collected by forwarding a packet to the next-hop neighbour $\mathcal{J}$ and then monitoring the latter's action in forwarding the same packet. The evidence captured for $\mathcal{J}$ is considered as benign behaviour, only if the packet has been forwarded without any modification. Alternatively, if the packet has been modified to disrupt the data flow, then the evidence is considered as a malicious behaviour. Furthermore, the malicious $\mathcal{J}$ is excluded from the corresponding communication flow until the completion of the flow.

After capturing the recent evidence, which may attribute to either a benign or malicious behaviour, node $\mathcal{I}$ accordingly revises the parameter p or n for its next-hop neighbour $\mathcal{J}$ at the sets P or N, respectively, as was detailed in §9.4.1. Also, node $\mathcal{I}$ revises k, the duration for which there has been no interaction between $\mathcal{I}$ and $\mathcal{J}$. Finally, it updates the direct opinion $({}^{\mathcal{I}}\omega_{\mathcal{J}}^{dir})$ for $\mathcal{J}$ by following the ∇ operator defined in (9.4).

9.4.5 Observed-Opinion

The notion of capturing evidence from the interactions of the neighbours is inspired by social psychology, where an individual's behaviour in a society is observed whenever the individual deviates from the normal behaviour. This, in turn, explains the psychology of the observers, who are inherent in remembering the individuals known for misbehaviours. Their point of view is to take advantage of their observations so that they can be cautious, whenever they interact in the future with the observed individuals who were known for misbehaving. Note that the definition of a normal behaviour may be subjective from the perspective of an observing individual, although a generic definition may exist in terms of social laws. In addition, the observers would fail to consider the individual's normal behaviours unless it was of direct benefit to them. However, it is noted that the observers do consider an extraordinary behaviour of the individual as a benign behaviour.

We consider node $\mathcal{G}$ in Figure 9.3 to study the process of $\mathcal{F}$ capturing evidence and updating its observed opinion $({}^{\mathcal{F}}\omega_{\mathcal{G}}^{obs})$ for $\mathcal{G}$. First, $\mathcal{F}$ overhears the packet forwarded by node $\mathcal{E}$ to $\mathcal{G}$ and then the packet forwarded by $\mathcal{G}$ on behalf of $\mathcal{E}$. Node $\mathcal{F}$ does not perform any further operations, if $\mathcal{G}$ has forwarded the packet without any modification. From the perspective of $\mathcal{F}$, forwarding a packet by following the specification of the AOMDV protocol is a normal behaviour. Further disregard of the normal behaviour assists in counteracting colluding attacks. Otherwise, $\mathcal{E}$ and $\mathcal{G}$ may be exchanging dummy packets between them to increase their observed opinions at $\mathcal{F}$. Alternatively, $\mathcal{F}$ revises the value of n for $\mathcal{G}$ in its negative set N (as mentioned in §9.4.1), only if $\mathcal{G}$ had modified the packet to disrupt the data flow. Node $\mathcal{G}$ is then excluded from the corresponding communication flow until the completion of the flow. Interestingly, $\mathcal{F}$ revises the value of p for $\mathcal{G}$ in its positive set P (as mentioned in § 9.4.1), only if $\mathcal{G}$ reports a genuine broken link to $\mathcal{E}$. This is considered to be an extraordinary action in the resource-constrained MANET. Note that any falsified report in a link would be captured as an evidence for $\mathcal{G}$'s malicious behaviour at the opposite neighbour (node $\mathcal{I}$, which was maliciously reported as unreachable), and also at other observing neighbours.

Node $\mathcal{F}$ then revises k, the duration for which there has been no observation for $\mathcal{G}$. It finally updates the observed opinion (${}^{\mathcal{F}}\omega_{\mathcal{G}}^{obs}$) for $\mathcal{G}$ by following the ∇ operator defined in (9.4). Note that a malicious behaviour would not only cause $\mathcal{G}$ to lose its direct opinion (${}^{\mathcal{E}}\omega_{\mathcal{G}}^{dir}$) at its previous-hop $\mathcal{E}$, but also the observed opinion (${}^{\mathcal{I}}\omega_{\mathcal{G}}^{obs}$) at all observing neighbours $\mathcal{J}$.

9.4.6 Recommended Opinion

For ease of explanation, the node that provides a recommendation is referred to as the *recommender* and the recommended node is referred as the *recommendee*. In [64–72] consider the evidence resulting from system failure as malicious behaviour in order to realise a reliable and trustworthy MANET. Most of these systems [3, 39, 64, 66, 68–71, 82, 83, 107, 112, 116, 125, 134, 137, 140, 169, 178, 190, 204, 214, 220–222, 233–235, 239, 241, 242, 252, 261, 270–273, 292, 294, 301–305, 308–311, 327, 335, 347, 372, 373, 379, 380, 401, 404, 424, 425, 461, 468], recommendations are communicated among nodes by disseminating explicit data packets or additional headers. In these models, the notion of disseminating recommendations corrupts the trust decisions for the following reasons. First, they lack well-analysed approaches to determine a recommender's bias embedded in a recommendation. Second, they fail to investigate whether the recommender exhibits free-riding and honest-elicitation behaviours. Even when these models do attempt to resolve these problems, they are unable to defend against those behaviours completely. In addition, the dissemination of the recommendations increases the overhead and hence, degrades the performance of the network.

In general, a recommender's opinion for a recommendee can be *deduced* from the disseminated recommendation. Given that there has been no change in the *deduced opinion*, it is then possible to determine whether the recommender will forward or discard a subsequent packet received from the recommendee. The reasoning holds as long as the context for both the disseminated recommendation and the deduced opinion is the same. In our model, we reverse the above reasoning to derive recommendations for a recommendee, rather than requesting the recommenders to disseminate recommendations as explicit messages or additional headers. In other words, a mobile node deduces its previous-hop's intention to forward an upstream packet on the behalf of the originator as the previous-hop's opinion for the originator. The node then derives the deduced opinion as the previous-hop's recommendation for the originator. Note that here the previous-hop is the recommender and the originator is the recommendee. The mobile node derives such recommendations from the route (forward-link or backward-link), only if the received packet is considered as trustworthy for forwarding. Recall from §9.4.3 that a mobile node forwards a packet only if it trusts its previous-hop (backward-link), the next-hop (forward-link), the source (originator) and the destination (target) of the packet. The same conditions would have been applied by the previous neighbour to its upstream previous-hop, and therefore by all the upstream intermediate nodes towards their previous-hop until the originator of packet. Also in our model, recommendations are derived only once for a communication flow, especially during the data flow. This enables our model to derive recommendations only from trustworthy routes because both the direct and observed opinions would have captured the evidence for malicious behaviours during the route discovery and accordingly, prevented the establishment of routes containing malicious intermediate nodes for the data flow.

Let us again consider the scenario shown in Figure 9.3, where node $\mathcal{I}$ receives a data packet from node $\mathcal{G}$, which was originated by the source $\mathcal{S}$. As mentioned earlier,

$\mathcal{I}$ prepares to forward the received data packet, only if the packet satisfies all the trust evaluations mentioned in §9.4.3. Subsequent to successful evaluations, $\mathcal{I}$ deduces $\mathcal{G}$'s (backward-link) intention to forward the packet on the behalf of $\mathcal{S}$, as $\mathcal{G}$'s opinion for $\mathcal{S}$. As discussed earlier, $\mathcal{I}$ then derives the deduced opinion as $\mathcal{G}$'s recommendation for $\mathcal{S}$. This is based on $\mathcal{I}$'s inference that $\mathcal{G}$ should have either a trustworthy or uncertain relationship with $\mathcal{S}$ to forward the packet originated by $\mathcal{S}$. From this inference, $\mathcal{I}$ postulates $\mathcal{G}$'s global opinion for $\mathcal{S}$ as, ${}^{\mathcal{G}}\omega_{\mathcal{S}}^{i-glo} = (Threshold, 0, (1 - Threshold))$, where $\mathcal{G}$'s belief (${}^{\mathcal{G}}b_{\mathcal{S}}^{i-glo}$) and uncertainty (${}^{\mathcal{G}}u_{\mathcal{S}}^{i-glo}$) for $\mathcal{S}$ is set to *Threshold* and *(1-Threshold)* respectively. Given that $\mathcal{I}$'s global opinion (${}^{\mathcal{I}}\omega_{\mathcal{G}}^{glo}$) for G can be retrieved from (9.9), $\mathcal{I}$ then derives its recommended opinion (${}^{\mathcal{I}}\omega_{\mathcal{S}}^{\mathcal{G}-rec}$) for $\mathcal{S}$ using the other component given in (9.11), *i.e.* recommendation (${}^{\mathcal{G}}u_{\mathcal{S}}^{i-glo}$) inferred from $\mathcal{G}$ for $\mathcal{S}$. Finally, $\mathcal{I}$ updates its overall recommended opinion (${}^{\mathcal{I}}\omega_{\mathcal{S}}^{rec}$) for $\mathcal{S}$ by applying $\oslash$ operator according to (9.13), and then integrating the result with the recommended opinion (${}^{\mathcal{I}}\omega_{\mathcal{S}}^{\mathcal{G}-rec}$) recently computed for $\mathcal{S}$ using (9.12). The same set of operations is then carried out by, (a) $\mathcal{J}$ *to derive* $\mathcal{I}$*'s recommendation for* $\mathcal{S}$, (b) $\mathcal{K}$ *to derive* $\mathcal{J}$*'s recommendation for* $\mathcal{S}$, (c) $\mathcal{L}$ *to derive* $\mathcal{J}$*'s recommendation for* $\mathcal{S}$ and, (d) $\mathcal{D}$ *to derive the recommendation from both* $\mathcal{K}$ *and* $\mathcal{L}$ *for* $\mathcal{S}$.

In summary, the proposed approach prevents a mobile node's opinion from being corrupted by the recommender and this, in turn, facilitates the node only to believe in its decisions. Hence, the node better resolves the issues concerned with the recommender's bias accompanied with the recommendation. Since the recommenders do not forward their opinions explicitly, they are also prevented from exhibiting both honest-elicitation and free-riding behaviours.

9.5 Adversary Model

We have modelled the adversary based on the case study presented in [277], where the feasibility of both *atomic* and *compound* level attacks against the AODV is discussed. However, we concentrate only on the atomic attacks that can enable the adversary to build compound attacks and therefore enable the adversary to embed itself in a communication flow for traffic analysis.

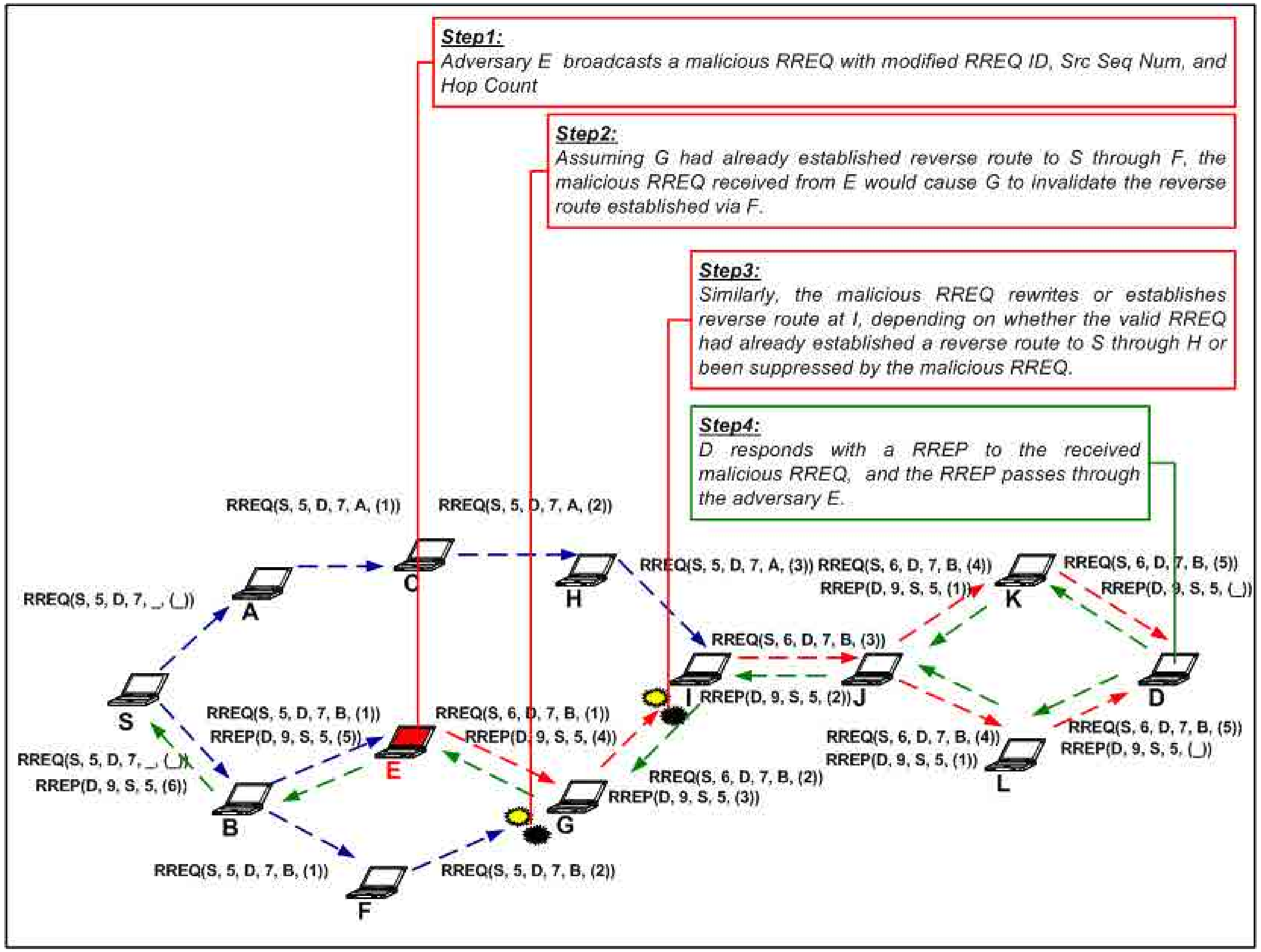
Unlike the adversary model discussed against the DSR in Chapter 5, the hop-based routing in the AOMDV opens up a different set of parameters for an adversary to operate in. In the case of the RREQ, the adversary can modify the following fields, (a) *the RREQ Identifier (ID)*, (b) *the hop count*, (c) *the source address*, (d) *the source sequence number*, (e) *the destination address*, (f) *the destination sequence number* and, (g) *the packet type*. In the case of the RREP, the adversary can either modify the destination sequence number or forge an RREP in response to an RREQ or arbitrarily insert a new RREP. Similarly, the broken links reported in an RERR can be modified or a new RERR can be forged by the adversary.

However, we believe those modifications that can disrupt the communication flow thereby leading to packet drops, would be recorded at the direct and observed opinions. Furthermore, our integrated fellowship approach discussed in Chapter 6 can be brought in as a defence against such packet drops. Hence in the following, we focus only on the composite of the atomic attacks that would allow the adversary to be a part of the communication flow to investigate the traffic flow and packet payload.

9.5.1 Modification of Route Request

An adversary can embed into a communication flow by propagating a maliciously-modified RREQ that can trigger all the intermediate nodes to establish a reverse route to the originator through it. Therefore, the reverse route via the adversary would not only transport the RREP to the originator, but also assures the data packets flow through to the adversary.

In the case of the AODV, an adversary can successfully increment the RREQ ID to propagate the maliciously-modified RREQ to overwrite the reverse route established by the valid original RREQ. In such a situation, all intermediate nodes up to and including the target are forced to re-establish the reverse route to the originator via the adversary. If the malicious RREQ happens to travel through a shorter path, then it can also suppress the propagation of the valid duplicate RREQs. Although, in the AOMDV


Step1:
Adversary E broadcasts a malicious RREQ with modified RREQ ID, Src Seq Num, and Hop Count

Step2:
Assuming G had already established reverse route to S through F, the malicious RREQ received from E would cause G to invalidate the reverse route established via F.

Step3:
Similarly, the malicious RREQ rewrites or establishes reverse route at I, depending on whether the valid RREQ had already established a reverse route to S through H or been suppressed by the malicious RREQ.

Step4:
D responds with a RREP to the received malicious RREQ, and the RREP passes through the adversary E.

Figure 9.4: Insertion of Modified RREQ.

an adversary can propagate the malicious RREQ (that contains an incremented RREQ ID) either to the target or to the intermediate node that has a forward route to the target; the adversary cannot overwrite the reverse route established by the valid original RREQ. When the malicious RREQ was propagated through a shorter path, the adversary could not prevent the valid duplicate RREQs (that have unique first-hop) from establishing reverse routes at the intermediate nodes, irrespective of suppressing their propagation. Given that there is more than one route in the AOMDV, the increment of the RREQ ID alone cannot assure that the adversary can embed in the communication flow. Therefore, the adversary is modelled to device a compound attack, where it takes advantage of the AODV/AOMDVs feature that updates the reverse route to the originator, provided that the originator's sequence number in the RREQ is greater than the sequence number held at the intermediate nodes. In the situation where the sequence numbers are equal, the RREQ with a smaller hop-count establishes the reverse route at the intermediate nodes.

For the reason stated above, the adversary increments both the RREQ ID and the source sequence number, and also decrements the hop-count of the RREQ. As explained earlier, the incremented RREQ ID assures the propagation of a malicious RREQ, while the incremented source sequence number invalidates the reverse routes corresponding to the outdated source sequence number. The decremented hop-count confirms that the data packets flow through the adversary.

For instance, in Figure 9.4, the malicious RREQ propagated by the adversary $\mathcal{E}$ (with the modified RREQ ID, source sequence number and hop-count) not only reaches $\mathcal{D}$, but also effectively overwrites the reverse route established by the valid original RREQ at $\mathcal{I}$ (*i.e.* $\mathcal{H} \mapsto \mathcal{C} \mapsto \mathcal{A} \mapsto \mathcal{S}$) and thereafter.

Otherwise, an adversary can perform one or a combination of the following, (a) *the decrement of the RREQ ID*, (b) *the reduction of the source sequence number*, (c) *the decrease in the destination sequence number*, (d) *the increment of the hop-count*, (e) *the alteration of the source (or destination) address* and, (f) *the modification of the packet type*. However, the consequence of such actions is synonymous with the malicious or selfish packet drop behaviour and, henceforth, we refer to our fellowship as the defence against such actions.

9.5.2 Modification of Route Reply and Route Error

The adversary can avoid the efforts involved in modifying an RREQ, by simply inserting a forged gratuitous RREP with better parameters on behalf of the target. This can trigger the data packets that are streamed to the target along the valid route to follow the re-directed route given by the adversary. Alternatively, the adversary can forge an RERR that can possibly induce route maintenance at the source node. Consequently, the adversary can then forge an RREP on behalf of the destination as the response to the forthcoming new RREQ.

The limitation of the RERR forgery is that the source node will not declare a target as unreachable until all available routes to the target are declared to be broken, and therefore the forged RERR may fail to induce a new route discovery at the initiator. The forged gratuitous RREP also suffers from the same issue when the route discovery reverts back to the originator with multiple routes to the destination including the forged RREP. As explained in the previous §9.5.1, the fields of the RERR and RREP can be altered to produce results equivalent to the malicious or selfish packet drop attacks, for which we refer back to the integrated approach detailed in Chapter 6.

As shown in Figure 9.5, the adversary $\mathcal{F}$ forges an RERR following the route establishment between the originator and the target via $\mathcal{E}$. Node $\mathcal{F}$ then forwards the forged RERR to $\mathcal{S}$ with $\mathcal{E}$ as the source of the RERR and the payload stating that $\mathcal{D}$ is unreachable. Given that an alternate route exists at $\mathcal{S}$ for $\mathcal{D}$ through the next-hop $\mathcal{A}$, the source node $\mathcal{S}$ fails to initiate a new route discovery. Assuming that the alternate route via $\mathcal{A}$ has been reported to be broken, $\mathcal{F}$ can then revert back to $\mathcal{S}$ with a forged gratuitous RREP as a response to the newly-initiated route discovery. In such a situation, $\mathcal{F}$ is expected to set a minimum route metric (*i.e.* the hop-count to the target) in the forged RREP so that the probability of being embedded in the communication flow is increased. The same approach is adopted in Figure 9.6, where

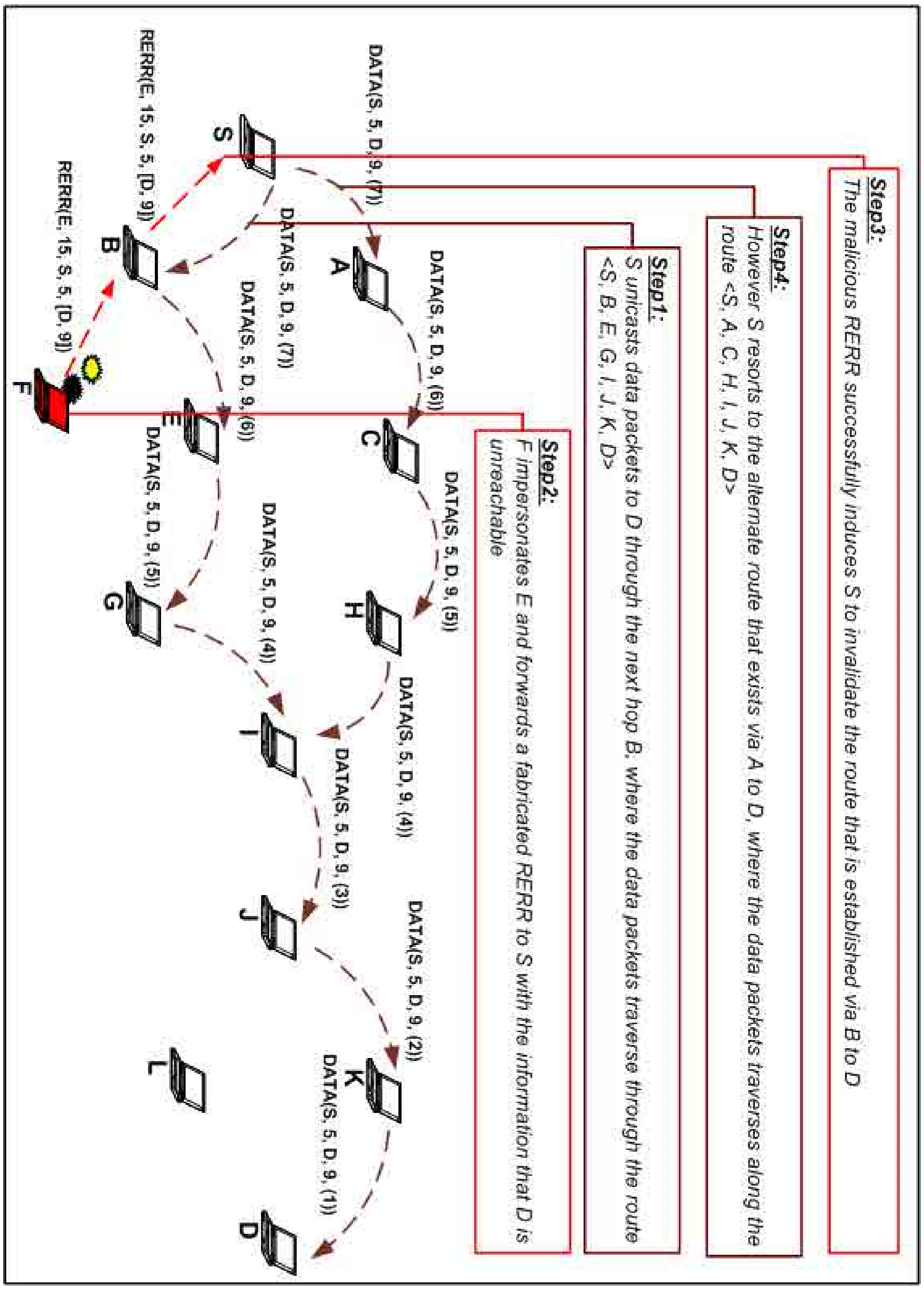

Step3:
The malicious RERR successfully induces S to invalidate the route that is established via B to D
Step4:
However S resorts to the alternate route that exists via A to D, where the data packets traverses along the route <S, A, C, H, I, J, K, D>
Step1:
S unicasts data packets to D through the next hop B, where the data packets traverse through the route <S, B, E, G, I, J, K, D>
Step2:
F impersonates E and forwards a fabricated RERR to S with the information that D is unreachable
S
A
B
C
E
F
G
H
I
J
K
L
D
DATA(S, 5, D, 9, (7))
DATA(S, 5, D, 9, (6))
DATA(S, 5, D, 9, (5))
DATA(S, 5, D, 9, (4))
DATA(S, 5, D, 9, (3))
DATA(S, 5, D, 9, (2))
DATA(S, 5, D, 9, (1))
DATA(S, 5, D, 9, (7))
DATA(S, 5, D, 9, (6))
DATA(S, 5, D, 9, (5))
DATA(S, 5, D, 9, (4))
RERR(E, 15, S, 5, [D, 9])
RERR(E, 15, S, 5, [D, 9])

Figure 9.5: Insertion of Fabricated RERR.

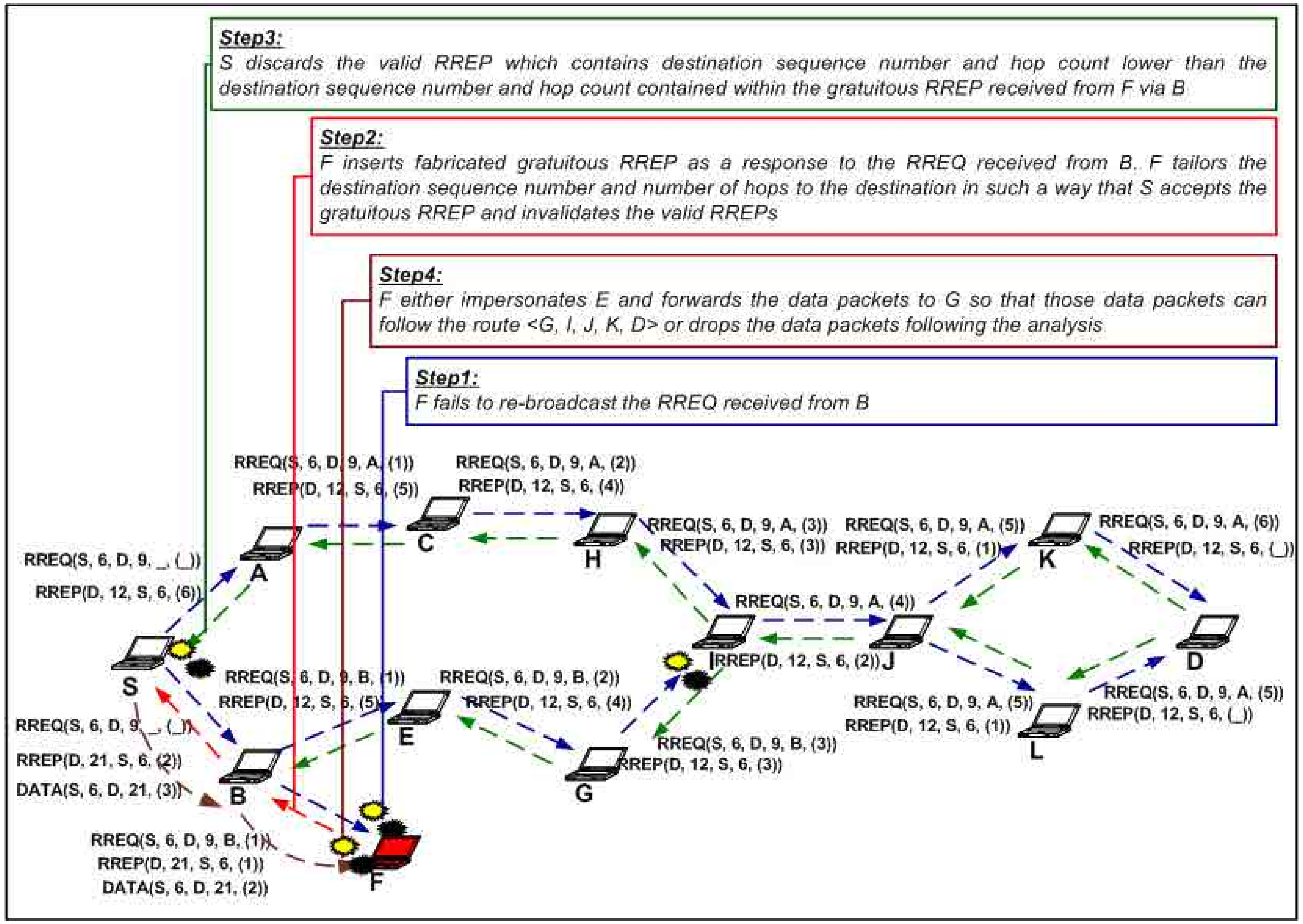


Figure 9.6: Insertion of Fabricated Gratuitous RREP.

$\mathcal{F}$ responds with a forged gratuitous RREP to the initial RREQ received from $\mathcal{S}$. The difference between the two scenarios is that $\mathcal{F}$ is effective in the former scenario because of its better position to set the minimum hop-count in the forged RREP (after witnessing the route establishment between $\mathcal{S}$ and $\mathcal{D}$ via $\mathcal{E}$).

9.5.3 SL-SMRTI's Effect on Adversary Model

In the case of the RREQ modification, the exposed behaviour of the adversary can be captured by the adversary's previous-hop and also by their common neighbours. Therefore, it is hard for the adversary to forward the corresponding RREP back to the originator, given that the adversary's previous-hop would discard the RREP. Any alternate route presented by the route discovery prevents the originator from propagating another RREQ. When there is no available route to target, the originator would wait for the interval defined by the AOMDV's specification prior to initiating another route discovery, rather than using the route discovered through the adversary. Even in an extreme condition when there is no available route to the target, the initiator would suspend the communication flow rather than to consider the adversary-embedded route. A possible improvement to the AOMDV in such a situation is to exponentially increase the interval between the route discoveries, similar to the technique employed for the contention window. Similar to the extension discussed for the DSR in Chapter 5, the SL-SMRTI enabled nodes are tailored to buffer the RREQ ID, source and destination sequence numbers and alternate routes reported by RREQs, and delay the recording of such information into the route cache until the data packets flow through the valid route.

The neighbour-detection feature via the HELLO packets in the AODV/AOMDV facilitates SL-SMRTI enabled mobile nodes to detect adversaries that report falsified broken links. Whenever an adversary injects a forged RERR, the node that was reported to be unreachable announces a HELLO packet to notify its presence in the neighbourhood, and also captures the adversary's misbehaviour in its direct opinion. Consequently, their common neighbours collect the evidence to reduce the observed opinion for the adversary. However, if the forged RERR induces a new route discovery, then both the common neighbours and the node reported as unreachable will be unable to prevent the propagation of the RREP to the originator via the upstream nodes. Note that the source node would initiate a subsequent route discovery only if all of its available routes to the target are invalidated and are beyond the control of the adversary.

Nevertheless, it is difficult for the SL-SMRTI enabled nodes to defend against the insertion of a malicious RREP by the adversaries as a response to the broadcasted RREQ. This is synonymous with the malicious RREP inserted by the adversary responding back to the new route discovery initiated as a result of its falsified RERR. The incapability results from the SL-SMRTI enabled nodes lacking a reference to verify the forged RREP, as in the case of modified RREQ (with the valid original RREQ) and the falsified RERR (with the HELLO packet). However, the probability of data packets being delivered through the adversary is low in a multi-routed setup. An effective defence against the insertion of such a forged RREP is to employ secure routing in conjunction with the SMRTI as explained in Chapter 7.

9.6 Simulation

We have used the NS2 to simulate the scenarios that involve the interaction between the adversaries and the SL-SMRTI enabled AOMDV nodes (or the basic AOMDV nodes). The SMRTI extensions of the DSR are inherited to implement the SL-SMRTI and are incorporated as an extension to the AOMDV modules. Similarly, the adversary model is implemented as an extension to the AOMDV. A mobile node retains all of its constituents as stated in Chapter 5, except the routing protocol (the AOMDV) and the latter's required constituents. The tools that were mentioned in Chapter 5 are used here for the same purpose. Mobile nodes that do not have the SL-SMRTI enabled are called because the *AOMDV nodes* and mobile nodes with the SL-SMRTI are known as *TME (Trust Model Enabled) nodes*. Nodes that exhibit malicious behaviours that are not only characterised by the adversary model stated in §9.5, but are also imbibed by the other modifications on the AOMDV that are synonymous with packet drops are called as *malicious nodes*. The feedback from the fellowship component that is incorporated as an extension above the MAC layer is used by the TME nodes to resolve the packet drop behaviours as explained in Chapter 6. The impact of the malicious nodes on the performance of the AOMDV nodes and the effect of the TME nodes against the malicious nodes are analysed in the absence of a secure routing protocol to study the performance of the SL-SMRTI.

9.6.1 Simulation Parameters

In this section, we present the NS2's parameters and the parameters chosen for the AOMDV and the TME nodes. All these simulation parameters and their values are summarised in Table 9.1, and the values chosen for the NS2's parameters apply to all types of mobile nodes (AOMDV, TME and malicious nodes). For the sake of consistency, we have mirrored most of the values for the NS2's parameters from Chapter 5. A significant change in the NS2's parameters can be seen in terms of the total number of nodes considered for each simulation run and the maximum number of malicious nodes that can be defined in a simulation scenario. This is to facilitate the comparison of the SL-SMRTI's performance with Pirzada's model [308, 310] because the latter has the closest resemblance to our simulation setup.

The NS2's version of the AOMDV relies on the IEEE 802.11's acknowledgement feature to establish multiple routes above the bidirectional links. Henceforth the destination propagates an RREP along the reverse route established by the RREQ, rather than invoking a new route discovery to the source. In addition, the promiscuous mode of operation is enabled to assist the TME nodes to listen to the transmissions of the other nodes. In our simulations, the AOMDV allows the intermediate nodes to respond with a gratuitous RREP to the incoming RREQs.

We have chosen 0.60 as the uniform threshold (Δ) value for all evaluations. The TME nodes initialise their positive *(p)* and negative *(n)* evidence counts with zero, where p and n counts the trackback to the direct and observed behaviours of the other

Table 9.1: Simulation Parameters for NS2, AOMDV, SL-SMRTI and Adversary

NS2 Parameters	
Radio Transmission Range	250m
Mobility	Random waypoint
Traffic Type	CBR
Date Rate	4packets/second
Payload Size	512 bytes
Total Simulation Time	500s
Total Number of CBR Connections	10
Average Pause Time	10s
Average Simulation Area	$1200 * 1200m^2$
Average Maximum Velocity V_{max}	20m/s
Total Number of Nodes	50
Maximum Number of Malicious Nodes	25
AOMDV Parameters	
Promiscuous Monitoring	True
Bidirectional Links	True
Gratuitous Route Reply	True
SL-SMRTI Parameters	
$\mathcal{I}$*'s Initial Positive Count* (p) *for* $\mathcal{J}$*'s Direct and Observed Behaviour*	0
$\mathcal{I}$*'s Initial Negative Count* (n) *for* $\mathcal{J}$*'s Direct and Observed Behaviour*	0
Initial Separation Interval (k) *between* $\mathcal{I}$ *and* $\mathcal{J}$	1
$\mathcal{I}$*'s Initial Direct Opinion* $\left({}^{\mathcal{I}}\omega_{\mathcal{J}}^{dir}\right)$ *for* $\mathcal{J}$	$(0.0, 0.0, 1.0)$
$\mathcal{I}$*'s Initial Observed Opinion* $\left({}^{\mathcal{I}}\omega_{\mathcal{J}}^{obs}\right)$ *for* $\mathcal{J}$	$(0.0, 0.0, 1.0)$
$\mathcal{I}$*'s Initial Recommended Opinion* $\left({}^{\mathcal{I}}\omega_{\mathcal{J}}^{rec}\right)$ *for* $\mathcal{J}$	$(0.0, 0.0, 1.0)$
$\mathcal{I}$*'s Initial Trust Relationship or Global Opinion* $\left({}^{\mathcal{I}}\omega_{\mathcal{J}}^{glo}\right)$ *for* $\mathcal{J}$	$(0.0, 0.0, 1.0)$
Cluster Name (δ_n)	Node ID
Cluster Type (δ_t)	Inter-cluster
Events	RREQ, RREP, RERR
Threshold (Δ)	0.60
Adversary Parameters	
Probability of Malicious Action	100%
Distribution of Malicious Action	Random

nodes. Furthermore, the TME nodes assign a unit value to the interval of separation *(k)* that influences both the direct and observed opinions held for the other nodes. The assigned value resembles the ignorance of the TME nodes about the behaviours of the other nodes before deployment. Therefore, the TME nodes establish their opening direct and observed opinions for the other nodes in the uncertainty zone based on the initialised parameters, *p, n,* and *k.* Similarly, they attribute their recommended opinion for other nodes in the same zone, and therefore classify their trust relationship with every other node in the network as an uncertain relationship, which is given by their initial global opinion in Table 9.1. The TME nodes collect evidence from all routing events and adopt an inter-cluster communication with every node that acts as an independent cluster. As shown in Table 9.1, the node identity is used as the cluster identifier.

9.6.2 Simulation Scenarios

The following two compositions, (a) *the TME and malicious nodes* or (b) *the AOMDV and malicious nodes*, are simulated with identical parameters for each of the simulation scenarios mentioned below. The total number of nodes for each of the above compositions is fixed at 50, although the proposition between malicious and the TME (or AOMDV) nodes may vary depending on the simulation scenario. The performance comparison between the TME and the AOMDV compositions for each of the following scenarios is derived from an average of 20 executions. The simulation scenarios considered for the performance analysis are given below:

Scenario I. In this scenario, we evaluated the performance of the TME and AOMDV nodes against an increasing proportion of malicious nodes from 0 to 25 in increments of 5, with a pause time of 10 s, a V_{max} of 20 m/s, and a simulation area of $1200 * 1200m^2$. Note that the proportion of the TME (or AOMDV) nodes decreases with the increasing proportion of malicious nodes to maintain the total count of nodes at 50. The objective of this scenario is to discover the proportion of the malicious nodes after which the performance of the TME and the AOMDV nodes fall noticeably.

Scenario II. This scenario presents the impact of mobility on the performance of the TME or AOMDV nodes against the malicious nodes. We have maintained the proportion of malicious nodes to TME or AOMDV nodes to 40% of the total nodes in the network. The performance of the TME or AOMDV nodes is then analysed by varying the V_{max} from 0 m/s to 20 m/s in increments of 5 with a pause time of 10 s, and a simulation area of $1200 * 1200m^2$.

We have evaluated the PDR and the latency together with the *Packet Loss (PL)* as the metrics for each scenario and the PL is defined as follows. However, the performance metrics, such as packet or byte overheads, are not evaluated in our simulations because the TME nodes do not generate additional packets or headers to communicate recommendations similar to the related models.

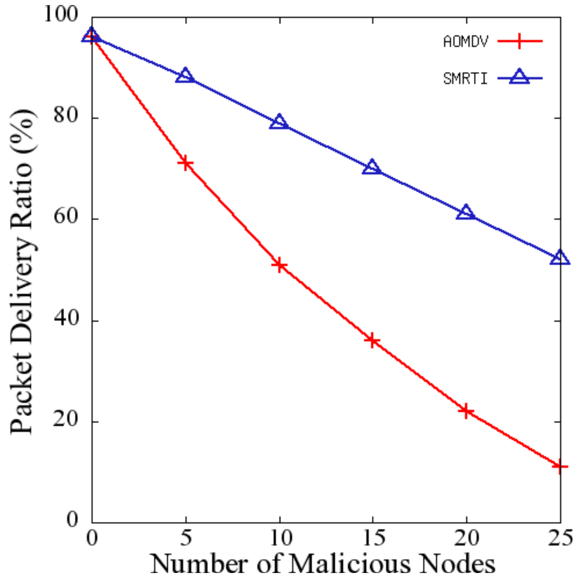
The ***Packet Loss (PL)*** is the average ratio of the total number of CBR data packets lost owing to misbehaviours without any notification to the total number of the CBR packets sent by the source.

9.6.3 Simulation Results

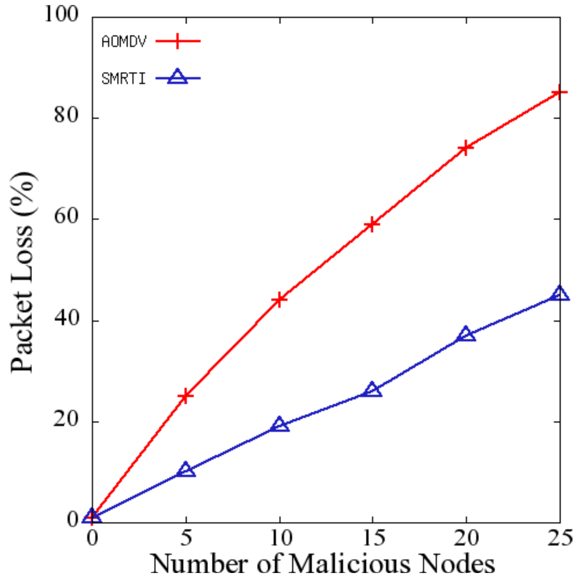
Here we present the performance metrics obtained for each of the scenarios and consequently analyse the performance of the TME nodes in comparison with the AOMDV nodes.

9.6.3.1 Scenario I

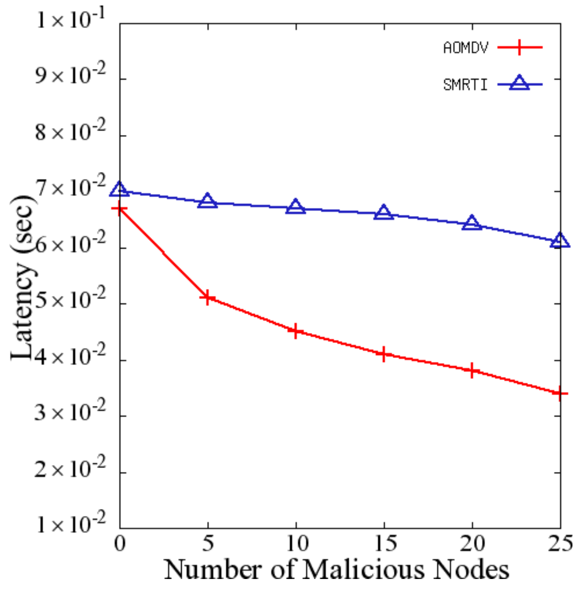
As shown in Figure 9.7(a), the PDR for the AOMDV nodes fall steeply with the increasing number of the malicious nodes. The steep fall results from the characteristic of the AOMDV nodes not being able to distinguish the benign behaviours from the malicious behaviours. As a result, they are prone to select routes that are subject to either a malicious modification of the RREQ or the insertion of an RREP. Such routes not only host malicious nodes as intermediaries, but also disrupt the data flow to bring down the PDR.



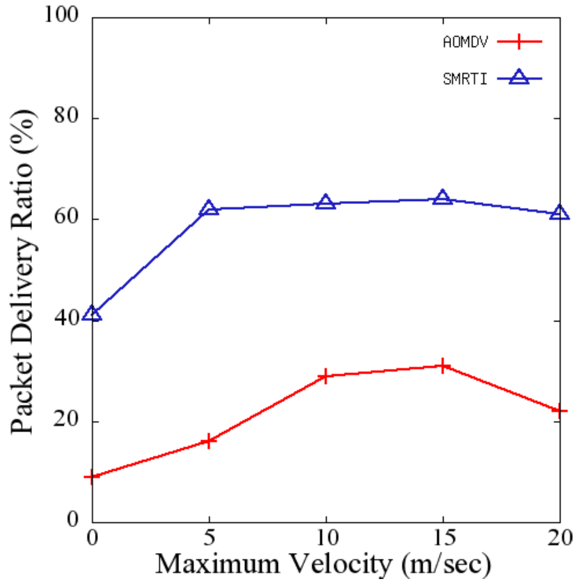
(a) Scenario I – PDR Vs Malicious Nodes



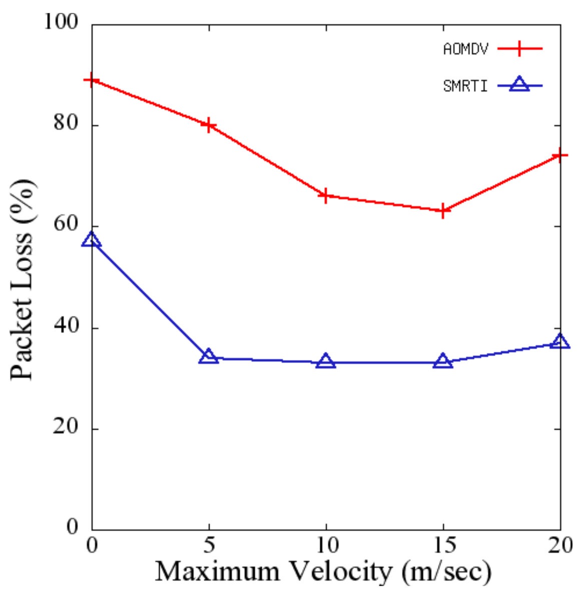
(b) Scenario I – Packet Loss Vs Malicious Nodes



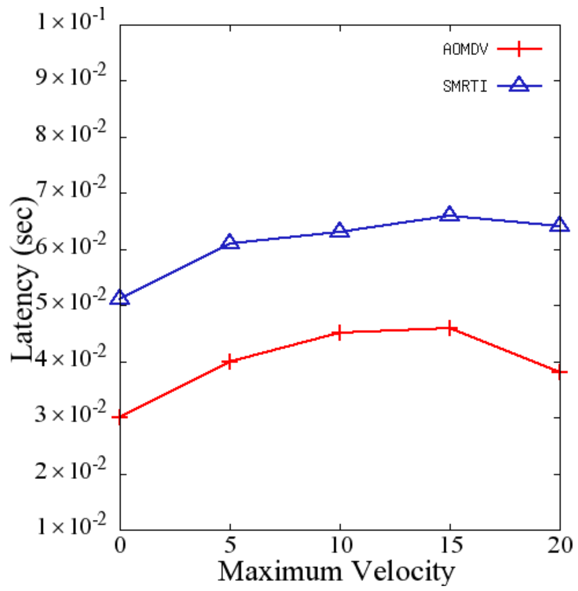
(c) Scenario I – Latency Vs Malicious Nodes



(d) Scenario II – PDR Vs V_{max}



(e) Scenario II – Packet Loss Vs V_{max}



(f) Scenario II – Latency Vs V_{max}

FIGURE 9.7: Performance of TME and AOMDV Nodes against Adversaries.

Alternatively, the TME nodes perform better because of their ability to make the improved routing decisions as mentioned in §9.4.2. Their better decision-making rests on the following facts, their capability to, (a) *represent their ignorance about the behaviour of newly-joining and existing nodes as uncertainty, while establishing or maintaining a trust relationship with those nodes*, (b) *differentiate benign behaviours from malicious behaviours with the help of evidence captured for direct and observed opinions and therefore to detect and isolate the malicious nodes from benign nodes* and, (c) *identify other TME nodes and to establish a trustworthy relationship with them.* Note that the capability of the TME nodes to short-list malicious neighbours even before interacting with them gives them leading edge in comparison with the AOMDV nodes. Therefore, the proportion of the nodes that are required to be classified by the TME nodes via direct interactions (as either benign or malicious) are reduced considerably, which is not the case for the AOMDV nodes. Also recall that the recommended opinion indirectly communicates the existence of the TME nodes to other TME nodes that are not seen among the AOMDV nodes. Finally Figure 9.7(a) confirms that the TME nodes do not incur overhead, because the PDR for both the TME and AOMDV nodes is the same in the absence of malicious nodes. This results from the design of the SL-SMRTI to collect evidence within the constraints of the MANET without incurring additional overheads.

Figure 9.7(b) again confirms that the TME nodes are successful in establishing valid routes despite the increasing proportion of the malicious nodes. However, while the proportion of malicious nodes increases, the PL for the TME nodes also increases. This is due to the decision to propagate only trusted packets, and that only through trustworthy previous and the next-hops.

As expected, the latency for the TME nodes is marginally greater than the latency of the AOMDV nodes in Figure 9.7(c) for the following reasons, (a) *the initiation of the route discoveries to find trustworthy routes when a previously-discovered route contains malicious node(s)*, (b) *the likelihood of the trustworthy routes being longer in hop-length than the optimal hop-length* and, (c) *the time taken for making subjective decisions at every hop.* Observe in Figures 9.7(a) and 9.7(c) that the latency values are paired with the corresponding PDR values, where the high values attribute to routes with variable lengths, while the low values predominantly attribute to routes with a shorter hop-length.

9.6.3.2 Scenario II

In Figure 9.7(d), the PDR for both the TME and the AOMDV nodes is low at 0 m/s. The decreased performance is due to the nodes not being able to establish a valid route when they are positioned in a malicious environment. In the case of the TME nodes, the observed opinion prevents the routes from being established through malicious neighbours. On the other hand, the AOMDV nodes discover routes through the malicious environment; however, the presence of the malicious nodes in the routes is sufficient to deteriorate the AOMDV's PDR. Nonetheless, the PDR increases for both the TME and AOMDV nodes as they become mobile, which can be confirmed by the reduced packet loss in Figure 9.7(e). The PDR then decreases as the velocity increases beyond the optimal value of 15 m/s. The reduction in the PDR results from broken links, which are the root causes for the loss of the data packets and the increased route discoveries.

Similar to Figure 9.7(c), the latency for the TME nodes is higher than the latency of the AOMDV nodes (Figure 9.7(f). At low velocities, the latency is caused by the time taken to make subjective decisions at every hop; while at high velocities, it results from the increased route discoveries and trusted routes being longer in hop-length than the optimal hop-length.

9.6.4 Discussion

As pointed out in Chapter 5, it is difficult to compare the performance of the SL-SMRTI model with the related models, because the related models have been analysed using other simulators or, at least, with different simulation parameters. We can closely relate our simulation setup to the Pirzada *et al.* [308, 310] model with the limited available information. Although the simulations exhibit a resemblance in terms of the

NS2 parameters, a close insight reveals that the Pirzada *et al.* model targets only blackhole and greyhole attacks that are induced through the route modifications. On the other hand, the SL-SMRTI encompasses a wide-range of attacks that includes the insertion of forged RREPs and RERRs. Furthermore, a significant difference can be noted between the simulation parameters, such as the simulation area, simulation time, and the maximum count of malicious nodes that are deployed in the simulated network. Our simulation setup uses an area of $1200 * 1200m^2$ compared with the Pirzada *et al.* simulation area of $1000 * 1000m^2$, and a simulation time of 500 s in comparison with 900 s. Our simulation setup evaluates the performance of the TME and AOMDV nodes against 50% of malicious nodes while the Pirzada *et al.* setup studies the performance up to 40% of malicious nodes.

The PDR's slope for the Pirzada *et al.* model closely resembles the PDR obtained for the TME nodes in Figure 9.7(a). However, the TME nodes that are enabled with the SL-SMRTI model outperform the Pirzada *et al.* model consistently on average by 10% and eliminate the routing packet overhead as generated in their model. Furthermore, from the comparison, it can be affirmed that the integration of the fellowship at the interface of the MAC and the routing layers has enabled the TME nodes to defend against the packet drop attacks. Otherwise, the TME nodes would have performed poorer than the Pirzada *et al.* model that primarily targets the packet drop attacks. We also believe that the performance of the Pirzada *et al.* model would suffer if their model was exposed to a composite of atomic attacks, such as the insertion of forged RERRs and RREPs. The belief results from the reality that the Pirzada *et al.* model is not tailored to detect such attacks, let alone having a defence. However, it is important to note that the Pirzada *et al.* trust model is one of the initial proposals that iterate the importance of an overlaid trust layer with the routing layer to make better routing decisions.

In summary, our simulation results confirm that the TME nodes make better routing decisions because of their ability to differentiate the benign nodes from the malicious nodes, and to better represent their trust relationship with the other nodes by managing ignorance as uncertainty. In addition, they also do not incur overhead and perform better by eliminating the issues related to the recommendations (such as recommender's bias, honest-elicitation, and free-riding) and operating within the constraints of the MANET. However, the TME nodes enabled with the SL-SMRTI will fail to exhibit a similar performance in the presence of spoofing attacks, in which the efficient key management mechanism and the secure routing protocols are required to authenticate the identity of the nodes and to protect the integrity of the routing information, as stated in Chapter 7.

9.7 Limitations

The SL-SMRTI inherits most of the limitations that have been associated with the SMRTI in Chapter 4. It also contrasts with the SMRTI in couple of limitations that are more to do with the change in the routing protocol, *i.e.* from the DSR to the AOMDV. Finally, the SL-SMRTI opens up a few more new limitations that are specific to subjective logic. All these limitations are discussed next.

9.7.1 Synonymous with SMRTI

Similar to the SMRTI, the SL-SMRTI steps into pure MANETs to establish a relationship between the mobile nodes using their behavioural evidence. Since the SL-SMRTI belongs to the trust management category, it is open to questions, such as, (a) *how authentic is the collected behavioural evidence* and, (b) *how well does the established relationship reflects the trust situation.* As we noted in Chapter 4 and detailed in Chapter 7, the presence of an offline-CA based key management and the development of a secure routing protocol based on such secret associations in the presence of the SL-SMRTI can counter the above questions.

The simplest form of the IDS in the MANET that relies on promiscuous monitoring is restricted by the, (a) *receiver collisions*, (b) *ambiguous collisions*, (c) *variations in transmission power* and, (d) *directional antenna.* Although the RTS/CTS handshake covers receiver and ambiguous collisions, it suffers from broadcasts such as route requests. The promiscuous monitoring based IDS do not have any answer to mobile nodes that vary the transmission power and use directional antenna. However, the capability of the SL-SMRTI to integrate into any IDS as long as it can capitalise the presented evidence into an opinion prevents it from being restricted by those restrictions that inhibit the promiscuous monitoring based IDS.

Following on the similarity with the SMRTI, the SL-SMRTI keeps out of DoS attacks that are launched at the MAC and the physical layers, and refers to the security solutions [1, 84, 319, 419, 459] that have been specifically proposed to defend against those DoS attacks — although, those proposals have not proved to be completely effective. Note that the enhancement or development of new solutions to counter those DoS attacks warrants a separate research, which is beyond the scope of this thesis.

9.7.2 Contrast with SMRTI

The SL-SMRTI contrasts with the SMRTI by handling fabricated gratuitous RREPs, collecting recommended evidence and setting up an initial reputation for other mobile nodes. The SL-SMRTI effectively defends against the insertion of a modified RREQ and a fabricated RERR by taking advantage of the original valid RREQ and the HELLO message received from the supposedly reported unreachable node, respectively. However, in the case of the gratuitous RREP, the SL-SMRTI lacks a reference for cross-checking the validity of the proclaimed gratuitous RREP and therefore profoundly struggles to differentiate the valid from the fabricated gratuitous RREP. Even in the presence of a key management based authentication mechanism, the problem persists, because it is hard to conclude whether an authenticated mobile node delivers a valid gratuitous RREP or the one with the modified destination sequence number and hop-count. Note that this problem is universal in all trust models. However, the SL-SMRTI differs from the related models by mitigating the problem, *i.e.* by evaluating the trustworthiness of the originator or the previous-hop before accepting any message from them. Although the use of a gratuitous RREP is available in the DSR, it is only an option in the DSR and does not impact the performance as much as it does in the AOMDV.

As we noted in Chapter 4, observed and recommended evidence comes low in count in comparison with direct evidence. In contrast to the SMRTI, the SL-SMRTI collects considerably less evidence for a recommended opinion that is to do with the change in routing protocol (*i.e.* from DSR to AOMDV). Unlike the SMRTI's role in the source-routed DSR, the SL-SMRTI collects the evidence for a recommended opinion only between a previous-hop and the source node pair in the hop-based AOMDV, where the previous-hop is the recommender and the source node is the recommendee. In other words, a mobile node can derive a maximum of one recommendation from a communication flow's route, which is of the order *(h-1)* lower than the recommendations derived in the DSR. Here, h denotes the total number of hops a mobile node is away from the source node, and *(h-1)* is the total number of recommendations that a mobile node can derive, based on its previous-hop's relationship with the upstream nodes by using the source route of the DSR. Regardless of the slow-rate resulting from the hop-based design of the AOMDV, the simulation results have shown that the SL-SMRTI nodes are capable of identifying other SL-SMRTI nodes.

In the case of choosing an initial opinion for other nodes, the SL-SMRTI performs better than the SMRTI with the help of subjective logic; where the SL-SMRTI enabled mobile nodes assign an *uncertain* opinion to all other mobile nodes in the network. The use of subjective logic not only facilitates an SL-SMRTI enabled node to resolve the difficulty of expressing ignorance regarding the behaviour of the other nodes, but also allows it to avoid the assignment of arbitrary opinions to them. Therefore, the SL-SMRTI stands out from the related trust models by not resorting to either a pessimistic or an optimistic value for the initial opinion, rather a value that reflects the true status of the relationship.

9.7.3 Limitations of Subjective Logic

Irrespective of the assurance rendered by subjective logic that it is a suitable component for the MANET based trust models in formulating evidence for an opinion and expressing ignorance in trust relationships, subjective logic does fall behind in two notable aspects, as discussed below. Recall subjective logic provides $\oplus$ operator to merge two opinions that are formulated from a different set of evidence so that the uncertainty is reduced in the resulting opinion. However, subjective logic fails to provide an operator that would allow the union of two opinions in such a way that the proportion contributed by one opinion over another opinion in the resulting opinion can be controlled at ease.

9.8 Conclusion

Although the SL-SMRTI is subject to the above limitations, those limitations are inherent to subjective logic, and their impact on the advantages rendered by subjective logic is yet to be studied, is unknown, and warrants further examination that is beyond the scope of this thesis. However, the results from our initial study show promise that subjective logic can be coupled with the SMRTI and possibly other MANET-based trust models to assist the mobile nodes in expressing and managing their ignorance as uncertainty in their trust relationships with the other nodes.

Subjective logic that is a probabilistic logic and has its foundation in the Dempster-Shafer belief theory of evidence has been successfully used by the SMRTI-enabled mobile nodes to analyse the trust relationships held with other mobile nodes. The binomial subjective opinion that represents the trust relationship expresses the truth of the proposition (for example, the trustworthiness of the mobile nodes) in terms of the subjective belief, disbelief and ignorance. As shown in this chapter, an opinion's base rate (a_x) is fixed at 0.5 in the context of the MANET to view uncertainty as half as optimistically as the actual belief. An opinion can also be visualised in an equilateral triangle using the three vertices (belief, disbelief and uncertainty), with the other visualisations being expectation, bayesian, and fuzzy bars. Furthermore, the mapping between an opinion and the $beta(\alpha, \beta)$ parameters enables the opinion to be visualised as beta distributions on a plot.

We then detailed how subjective logic is tailored to meet the requirements of the MANET and to tightly-couple with the SMRTI. Among the operators defined for subjective logic, we short-listed the consensus ($\oplus$) and discounting ($\otimes$) operators, improvised an evidence-to-opinion mapping (∇) operator, and finally proposed fading ($\oslash$)

and comparison ($\geqslant$) operators.

Although ∇ can successfully map the opinion parameters to the beta PDF parameters, it fails to express the notion of ignorance in the mapping when the nodes are separated by mobility. Hence, we adapted ∇ to resolve the notion of ignorance that slips between the separated mobile nodes by assigning the count of the separation interval to the variable k of the beta PDF parameters. The separation interval is determined by the average time taken for the communication flow by the mobile node that performs the mapping of the evidence to an opinion. The operator ∇ is heavily used for the mapping of the direct and observed evidence to the direct and observed opinions, respectively. Recall that the SMRTI operates as a decentralised and independent module at every node and collects direct and observed evidence from the behaviours of neighbouring nodes. The mapping of the direct evidence to the direct opinion (which results from direct interactions) enables the SL-SMRTI nodes to identify the malicious nodes from benign nodes. Alternatively, the observed opinion that results from the observed evidence enables the SL-SMRTI nodes to shortlist the malicious nodes even before interacting with them. The inherited $\oplus$ is used by the SL-SMRTI nodes to combine their different types of opinions (direct, observed and recommended) into a single global opinion. Alternatively, $\oplus$ is also used to combine global opinions that are held for the other mobile nodes to reach a single global opinion for a context, where these mobile nodes are involved in that context. We proposed $\geqslant$ to evaluate the global opinion and hence, to elucidate whether the trust relationship is trustworthy, uncertain or untrustworthy.

The SL-SMRTI nodes exclusively inherited $\otimes$ to formulate their recommended opinions, for which they rely on the global opinion held for the recommender and recommender's global opinion for the recommendee. The latter is derived from the relationship deduced between the recommender and the recommendee, specifically from the willingness of the recommender to forward packets on behalf of the recommendee. The SL-SMRTI nodes capture the recommended evidence whenever a previous-hop participates in a communication flow and forwards packets on the behalf of originator. The formulated recommended opinion is merged with the overall recommended opinion using $\oplus$. Whenever the SL-SMRTI nodes fail to see recommendations, it uses $\oslash$ to reflect the duration in which there were no recommendations before merging the latest recommended opinion with the overall recommended opinion. The outcome of a new operator $\oslash$ is because $\otimes$ is not designed to solve the notion of ignorance when the mobile nodes are separated by mobility. Remember the unique approach adopted for deducing recommendations eliminates, (a) *honest-elicitation*, (b) *free-riding*, (c) *recommender's bias* and, (d) *additional message dissemination*. As stated earlier in this thesis, the decoupled design of the evidence collection (direct and observed) and the opinion formulation further facilitates the SL-SMRTI enabled nodes upgrading the IDS because of the availability of a comprehensive lightweight IDS or whenever there is the necessity to collect evidence for emergent attacks.

We duly introduced the definition of trust together with the architecture of the SL-SMRTI and presented it with the formal expression of an SL-SMRTI enabled node's trust relationship with another node. The expression is a tuple with eight attributes, including the contexts from which the evidence is captured and for which the decisions are made, and extends the applicability of the SL-SMRTI to the cluster-based MANETs. We explained the operation of both the AODV and the AOMDV for the sake of completeness and then demonstrated the working of the SL-SMRTI and the AOMDV. The discussion included the AOMDV's routing contexts for which the SL-SMRTI assists the mobile nodes in making trust-enhanced subjective decisions. In our discussion, we detailed the policies defined for each of those contexts and how the SL-SMRTI nodes use their trust relationships to evaluate the trustworthiness in those contexts. In summary, those policies either assist the SL-SMRTI nodes to reactively isolate the malicious nodes or to pair with the benign nodes. As noted earlier in the thesis, the separation of the policies from the evaluation of the trust relationships enables the SL-SMRTI nodes to accommodate new contexts (from a higher layer, such as the application layer) and accordingly define efficient policies to counteract emerging attacks. Recall the SL-SMRTI enabled nodes do not share their opinions with other nodes and hence, their decisions are never corrupted by the opinions of the other nodes.

Later in the chapter, we presented an adversary model in which we discussed the possible modifications and the forged insertions that an adversary could employ against the AOMDV protocol. Those modifications and forged insertions opened wide the door for the adversaries to perform one of the following attacks, (a) *a route discovery disruption*, (b) *a detour*, (c) *a data flow disruption*, (d) *a blackhole*, (e) *a grayhole*, (f) *a route error disruption* and, (g) *a gratuitous detour*. Subsequently, we demonstrated the effect of the SL-SMRTI's role against such adversaries. We then established the context required for the simulations by detailing the simulation parameters chosen for the NS2, AOMDV, SL-SMRTI and adversaries. Given that there is no benchmark for setting up simulation parameters, the choice for the simulation parameters was based on the same reasons given earlier in the thesis. Finally, we have also described the simulation scenarios and performance metrics that are considered for each of the two scenarios.

The SL-SMRTI enabled nodes, known as the TME nodes, demonstrated a superior PDR and a lower PL in comparison with the AOMDV in both scenarios. The TME nodes exhibited a better performance against the varying proportions of the malicious nodes for the following reasons, (a) *the detection of both the benign and malicious neighbours using direct evidence and also from the observed evidence, even without interacting with those neighbours*, (b) *the identification of the multi-hop benign nodes using the recommended evidence such that the collected evidence is free from the issues related to the recommendations*, (c) *the reactive decision policies to enhance the security of routing* and, (d) *the feature representing the notion of ignorance as uncertainty in the trust relationships using subjective logic*. We also discovered that the TME nodes struggle to deliver a higher performance at immobility because of being embedded in a malicious environment. Since the latency is proportional to the delivered performance, the TME nodes exhibit a higher latency in comparison to the AOMDV nodes for the following reasons, (a) *a longer duration to discover the trustworthy routes in a high proportion of malicious nodes* and, (b) *the use of longer routes when packets are routed around malicious nodes*. Finally, we have also compared and contrasted our SL-SMRTI model with one of the closely-related trust models in literature, and shown how the SL-SMRTI outperforms the related model.

In summary, the SL-SMRTI renders a promising solution for representing the notion of ignorance in trust relationships and hence, closely represents the state of the trust relationships. Therefore, subjective logic coupled with the SMRTI confirms better trust-enhanced decisions and therefore an improvised defence against adversaries in the absence of a CA or tamper-proof hardware. In our future work, we foresee to the integration of the SL-SMRTI with the secure routing protocol discussed in Chapter 7.

10 Conclusion

To the best of our knowledge, we believe that our work is the fore-runner for a two-layered[1] trust-enhanced security architecture for the MANET. In this thesis, we have successfully established, (a) *the rationale for a two-layered approach to secure the MANET based on the analyses of different categories of related models and thereof highlighting on the root-causes of their shortcomings*, (b) *the functional design for such a two-layered approach that complements the inherent issues of the MANET* and, (c) *the system architecture that incorporates the two-layers to achieve trustworthy, secure and operational communications, whilst at the same time meeting the traditional security requirements.* The results of our research work have been published as research papers at *ten* different IEEE and ACM conference proceedings [23–25, 27, 28, 31–35] and *two* book chapters as a part of edited books by the Springer and Wiley [29, 30]

[1]A layer of prevention systems and another layer of detection-reaction systems.

publications. Furthermore, we have *another* book chapter [26] accepted for publication.

During the systematic development of our trust-enhanced security architecture, we initially focused on the development of a detection-reaction layer and, in particular the operational facet of the MANET. Therefore, we developed the obligation-based fellowship model in the Chapter 3 that targeted on mitigating the packet drop and flooding attacks and, thereby enforcing the co-operation among the mobile nodes. Interestingly, fellowship is a composition of both prevention and detection-reaction characteristics, in which the prevention characteristic pro-actively defends against the flooders based on the thresholds pre-defined for the packet-reception rate. On the other hand, the detection-reaction characteristic warrants a learning curve to defend against the packet droppers, *i.e.* to detect them based on the count of dropped packets and, then to react to them by excluding them in the future communications and also by ignoring their future requests. Although flooders are pro-actively prevented from hijacking the bandwidth beyond a hop, the one-hop fellowship nodes are also required to react to those flooders. As stated in Chapter 3, recollect that appropriate thresholds govern these reaction decisions. Another distinguished feature of fellowship is the elimination of additional control information for communicating or learning the presence of packet droppers or flooders. Recall that the downside of such an approach is the delay in communicating or learning the presence of such attackers. In addition, we have presented the operation of fellowship against various attack scenarios and also successfully demonstrated its better performance using comprehensive NS2-based simulations. In particular, the simulation scenarios not only varied the proportion of attackers and attack scenarios to study the performance of fellowship model, but also other parameters that impacted the operation of the MANET. Those parameters included the mobility and pause time of the nodes and, the node density in the MANET. Notably, these parameters facilitated the understanding of how the setup of the MANET can impact the performances of the fellowship model. We have published *three* research papers based on our fellowship model at conference proceedings, (a) *articulating the impact of packet droppers on the operation of the MANET* [25], (b) *the conceptual design of fellowship model* [24] and, (c) *the working of the fellowship model* [27].

Given the maturity of prevention mechanisms (such as secure routing and key management), we then focused on the development of the concluding part of the detection-reaction layer, *i.e.* a trust model for the MANET in the Chapter 4. Our trust model, the SMRTI, primarily focussed on resolving the limitations harboured by the related trust models and also on capitalising all evidence that are available within the limitations of the MANET for making better decisions. The SMRTI-enabled mobile nodes are designed to collect evidence based on their direct interactions with the neighbours and also from the recommendations derived for the other nodes. Unlike the related models, the SMRTI-enabled mobile nodes also collected evidence based on the interactions observed between two neighbours. Recall that the evidence collected from the direct interactions took higher precedence, while the observed and recommended evidence followed the sequence in ascending order. Interestingly, the observed evidence became the fore-teller that which neighbours will misbehave even before interacting with them, whilst the recommended evidence pointed out the trustworthy nodes in the network. As noted in the Chapter 4, the approach developed for communicating recommendations overcame the issues such as, (a) *honest-elicitation*, (b) *free-riding*, (c) *recommender's bias* and (d) *dissemination of additional control information.* Remember that the notion of capturing the observed evidence and deriving the recommended evidence are the novel approaches deployed in the SMRTI. Furthermore, the SMRTI handled the reputation-saturation problem by limiting the reputation ratings for persistently benign and malicious nodes. The design of the SMRTI also resolved the uncertainty that would result in any trust relationship as a result of nodal mobility. Recall that the SMRTI-enabled nodes made better decisions depending on the contexts and then the policies defined for those contexts. Similar to the fellowship model, we demonstrated the operation of SMRTI-enabled nodes against various attack scenarios and also their better performance against the malicious nodes using comprehensive NS2-based simulations in the Chapter 5. In those simulations, the proportion of attackers were varied to test the strength of SMRTI-enabled nodes. Also other parameters such as mobility and pause time of the nodes and, the nodal density of the MANET were varied to study the impact of MANET's setup on the performances of the SMRTI-enabled nodes. We have successfully published *two* research papers and *two* book chapters based on our SMRTI model, (a) *focusing on the novel approach deployed for deriving recommendations* [32] and, (b) *the design and performance analysis of the SMRTI model* [33].

Having built the appropriate detection-reaction based mechanisms (fellowship and SMRTI), we then pursued to integrate those mechanisms to realise the functional detection-reaction layer in the Chapter 6. Recall that during the integration, the fellowship model was adapted to feed the evidence into the SMRTI's detection engine, whilst the SMRTI's reaction engine was modified to govern the fellowship's decisions. This work has been published as a research paper in a conference proceedings [34].

Although the detection-reaction layer has been achieved, a functional defence-in-depth architecture can be realised only in the presence of a prevention layer. Hence, we resolved the design issues related to the coupling of prevention and detection-reaction layers and ensured that such a design allowed both the layers to complement each other in Chapter 7. Recall that any key management mechanism chosen for the MANET has to satisfy the requirements of the MANET (*i.e.* managed or pure) and, a secure routing mechanism only capitalised the secure associations established by the key management to secure the communications. In particular, offline-CA based key management mechanism was recommended for the pure MANETs and then a secure routing protocol such as, BISS, was referred to build on top of it to secure the communications. Alternatively, SMG was adapted for the managed MANETs and a new secure routing protocol known as Scasec was proposed to secure the communications. Recall that any secure routing protocol and key management mechanism that can complement the characteristics of the MANET, can be deployed as suitable candidates for the prevention layer. In the TEAM, both the prevention and detection-reaction layers have been integrated so that the evidence for benign and misbehaviours is always forwarded to the SMRTI by the secure routing protocol. In return, SMRTI assists the secure routing protocol to make trust-enhanced decisions. Similarly, the key management mechanism is designed to take advantage of the SMRTI's knowledge to make decisions related to the key renewals and revocations. Note that the TEAM has not only been designed to integrate the detection-reaction based fellowship and SMRTI models with the prevention-based key management and secure routing mechanisms, but also to facilitate its capability to incorporate futuristic prevention and/or detection-reaction based extensions. We have published this work as a research paper in a conference proceedings [31].

We then adapted the notion of fellowship to the MANETs that support anonymous communication to demonstrate the extensibility of our models. Given that it is infeasible to correlate between the received and transmitted packets in any anonymous communication, the fellowship model was tailored to mitigate only the flooders. In Chapter 8, we demonstrated the effectiveness of our fellowship model against the flooders that disrupt anonymous communication using small-scale NS2 simulations. Similar to the fellowship version presented in the Chapter 3, the anonymous communication-based fellowship model eliminated additional overhead and defended against flooding attack nearest to the source of attack. Also, this work has been published as a research paper in a conference proceedings [35].

In Chapter 9, we coupled subjective-logic with SMRTI so that mobile nodes can express and manage their ignorance as uncertainty in their trust relationships with the other nodes. Recall that an opinion in subjective logic is expressed in three-dimensions, *belief*, *disbelief* and *uncertainty*. During the coupling process, we also proposed new operators for subjective logic to meet the requirements of the MANET. We then demonstrated the effectiveness of subjective-logic based SMRTI (*i.e.* SL-SMRTI) and fellowship models against multi-routed version of AODV, *i.e.* AOMDV. Finally, we compared and contrasted the performance of SL-SMRTI with one of the closely related trust model [308, 310] and shown how SL-SMRTI outperformed the related model. This work has also been published as a research paper in a conference proceedings [28].

10.1 Further Work

In this section, we identify some possible directions for further research work based on our models and proposed system and architecture.

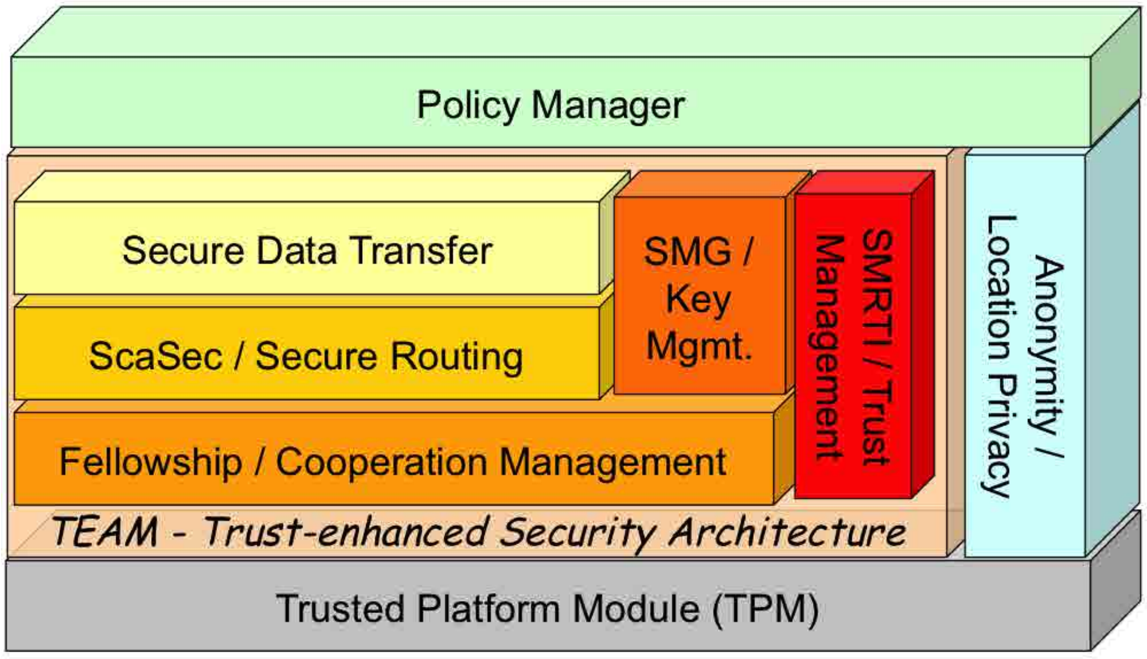

Policy Manager
Secure Data Transfer
ScaSec / Secure Routing
Fellowship / Cooperation Management
SMG / Key Mgmt.
SMRTI / Trust Management
Anonymity / Location Privacy
TEAM - Trust-enhanced Security Architecture
Trusted Platform Module (TPM)

FIGURE 10.1: TEAM and Related Security Areas in the MANET.

First we believe there is scope for doing further work on conducting more simulations especially in the context of the integrated model of the fellowship and SMRTI. These additional simulations should primarily focus on studying the performance of the integrated model by employing different reactive, proactive and hybrid protocols. Given that the decisions of the SMRTI are influenced by the evidence provided by the fellowship, in these additional simulations, one needs to vary the parameters of fellowship and SMRTI to determine the optimal values required for those parameters.

As shown in 10.1, we believe an area of further research to be in anonymity and location privacy in the MANETs. Anonymous communications may be employed in the MANETs to conceal one or more of the following information: (a) *source-destination pair*, (b) *number of neighbours located in the environment or around the route*, (c) *node locations*, (d) *topology changes*, (e) *traffic load in the network* and (f) *key connectors in the network*. Although the requirements for anonymous communications are different from the requirements of non-anonymous communications in MANETs, we have successfully adapted fellowship to defend against flooders in the MANETs that support anonymous communications. Therefore, we anticipate to explore the possibility of expanding the SMRTI to MANETs that support anonymous communications. Factors that will challenge such an adaptation are, (a) *anonymous routing protocols nature to broadcast unlike to unicast as in non-anonymous communications* and (b) *design questions in relation to whether or not direct, observed and recommended evidence can be captured.* However, on inheriting the key management mechanism and anonymous security protocol as the prevention layer and the anonymous communication-based fellowship with an adapted modular version of SMRTI as the detection-reaction layer, we believe that our architecture TEAM can be successfully applied to support anonymous communications. Regardless of the adaption, further work is required to address issues in capturing the evidence either through an IDS-based promiscuous monitoring for direct and observed reputations or via derived recommendations. For instance, the possible source of evidence for the SMRTI appears to be from the anonymous secure routing protocol. In particular, such evidence can be expected to be confined only to the misbehaviours exhibited by the previous-hop neighbours. Nevertheless, the adapted version of the SMRTI can render its service by facilitating the sub-models of the TEAM to make trust-enhanced decisions.

Another area that we foresee further work is in the policy set for the TEAM. Given that the focus has been within the network layer, the policies have remained much simpler and static than anticipated. However, as the application layer is coupled with the SMRTI to feed the evidence from a variety of applications then the decision-making process will become harder. Nevertheless, such an integration of the TEAM with a policy manager at the top and a tamper-proof hardware module, such as *Trusted Platform Module (TPM)*, at the bottom can be a possible solution for achieving a seamless trusted interaction between the end-to-end applications in MANETs. The presence of such a tamper-proof hardware can also add more flexibility and resolve strong-identity related issues during the design of key management mechanism and secure routing protocols.

Finally, another area of further work involves the study of the impact of different initial values of the subjective-logic based opinions (direct, observed and recommended) and threshold values. We envisage that the architecture and the simulation setups that have been described in our thesis can be used for this purpose, and the performance results of the SL-SMRTI with those of related trust models to be compared. It will also be interesting to explore the performance characteristics of the TEAM with the SMRTI with that of SL-SMRTI enabled in the context of selected reactive, proactive and hybrid routing protocols.

A
UML Design Diagrams

Here, we present the UML class and sequence diagrams of the fellowship and SMRTI models. The class diagrams describe the classes and their relationships (such as inheritance, aggregation and association) and, their operations and attributes. Alternatively, the sequence diagrams present the visual flow of logic in our models. These facilitate the documentation, analysis, and validation of our logic. In particular, we confine to the UML class and sequence diagrams of the fellowship and SMRTI models since the integrated or extended models are built upon them.

A.1 Fellowship: Class Diagrams

The following list of figures presents the object-oriented design of the fellowship model,

- **Figures A.1 and A.2:** Presents the conceptual design of the fellowship model in terms of the class diagrams and their relationships.

- **Figure A.3:** Interfaces and the derived classes of the data structures used in the fellowship model.
- **Figure A.4:** Structure of the fellowship's enforcement component and the interfaces through which it access the buffered packets and the model's parameters.
- **Figure A.5:** Structure of the fellowship's rate-limitation component and the interfaces through which it access the model's parameters.

A.2 Fellowship: Sequence Diagrams

The following list of figures presents the logical flow of controls within the fellowship model,

- **Figure A.6:** Sequence that takes place whenever a new packet is received by the mobile node and passed on to the fellowship's rate-limitation component.
- **Figures A.7 and A.8:** Calls initiated by the fellowship's rate-limitation component for forwarding an incoming packet to the enforcement component.
- **Figure A.9:** Controls invoked by the fellowship's rate-limitation component to allocate the reception channel for the requesting neighbours.
- **Figure A.10:** Sequence generation at the restoration component whenever a mobile node detects that the transmission channel is under contention or the transmission buffer is congested.
- **Figures A.11 and A.12:** Inter-modular communications that are generated at the enforcement component whenever a mobile node receives an acknowledgement or a duplicate.
- **Figure A.13:** Modular calls for updating the contributions for other nodes whenever those nodes forward packets or exhibit misbehaviours such as flooding or dropping packets.

A.3 SMRTI: Class Diagrams

The following list of figures presents the object-oriented design of the SMRTI model,

- **Figures A.14 and A.15:** Conceptual design of the SMRTI model in terms of the class diagrams and their relationships.
- **Figure A.16:** Trust and reputation structures of the SMRTI and, their interactions and associations.
- **Figure A.17:** Interfaces and class diagrams used for data structure traversals, where the data structures may either contain packets or reputation values for other nodes.
- **Figure A.18:** Class structure for storing packets (A.18(b)), reputation ratings (A.18(b)) and, the weights (A.18(c)) that would be used during the trust evaluation of a context.

A.4 SMRTI: Sequence Diagrams

The following list of figures presents the logical flow of controls within the SMRTI model,

- **Figures A.19 and A.20:** Flow of function calls for evaluating the trust of a context by the SMRTI.
- **Figure A.21:** Control sequence carried out by the SMRTI to determine whether or not a packet is trusted.
- **Figure A.22:** Call sequence for evaluating the trust of a mobile node using the reputations held by the SMRTI.
- **Figures A.23 and A.24:** Modular calls that are initiated by the SMRTI's direct reputation module on overhearing a packet transmitted by the next-hop neighbour. The examination allows the direct reputation module to determine the behaviour exhibited by the next-hop neighbour towards the forwarded packet.

- **Figures A.25 and A.26:** Control sequence initiated by the SMRTI's observed reputation module on overhearing a packet transmitted by a neighbour. It determines whether or not the captured packet relates to an interaction with one of its common neighbour. Depending on the sequence, the examination would allow the observed reputation module to determine the misbehaviour exhibited by the neighbour even before interacting with it.
- **Figures A.27 and A.28:** Flow sequence for traversing the observed reputation related packet buffer in search of a duplicate packet.
- **Figures A.29 and A.30:** Inter-modular call sequences that are initiated by the SMRTI's recommended reputation module to derive the indirect reputation from the received packet.
- **Figures A.31 and A.32:** Inter-modular communication sequence for matching a duplicate packet with one of the packet located in the direct reputation related packet buffer.
- **Figure A.33:** Control sequence carried out by the SMRTI to locate the reputation (direct, observed or recommended) of a mobile node.
- **Figure A.34:** Sequence of calls made by the SMRTI to revise the reputation (direct, observed or recommended) of a node in proportion to the duration for which there has been no interaction.
- **Figure A.35:** Flow of function calls for determining the total count of duplicate packets in a buffer.
- **Figure A.36:** Inter-modular sequence for dumping the contents of a packet table for debugging.
- **Figure A.37:** Modular calls made for dumping the contents of a reputation table (direct, observed or recommended) for debugging.

- **Figure A.38:** Control sequence carried out by the SMRTI to purge expired packets for which a corresponding match was not detected in the wireless channel.
- **Figures A.39 and A.40:** Flow sequence for storing a packet in a buffer before it can be traversed and cross-checked with promiscuously captured packets in the future.
- **Figure A.41:** Sequence of function calls made for dynamically assigning weights to various participants in a context at the time of trust evaluation.

A.5 DSR and CBR Generator

In the following, the figures present the class integration between the SMRTI and DSR and also the design of our custom-made CBR generator to meet our requirements.

- **Figure A.42:** The class diagram presents the association between the SMRTI and the DSR.
- **Figure A.43:** Presents the object-oriented design of our CBR generator and its associations, aggregations and relationships with other structures.
- **Figure A.44:** Sequence of calls indicating the process carried out for generating CBR traffic.
- **Figure A.45:** A detailed flow revealing the internal call structures of CBR traffic generator.

A.6 Class and Sequence Diagrams

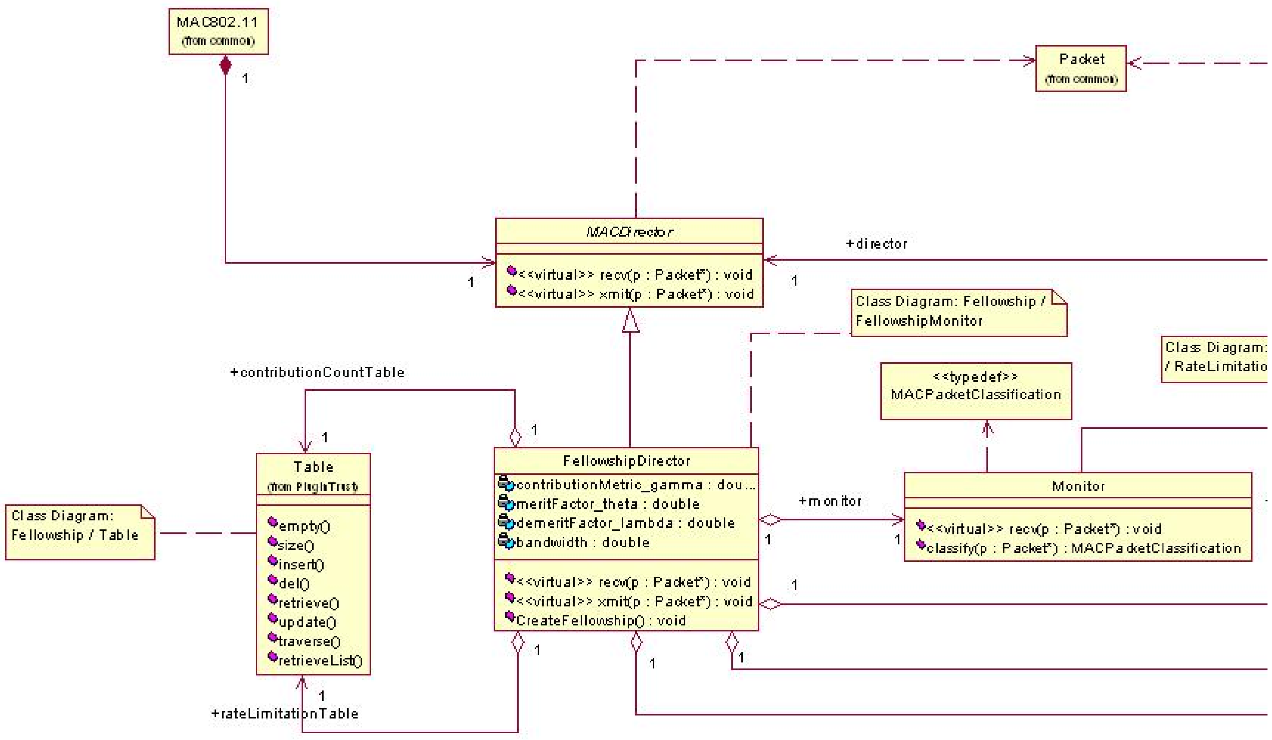

MAC802.11
(from common)
Packet
(from common)
MACDirector
<<virtual>> recv(p : Packet*) : void
<<virtual>> xmit(p : Packet*) : void
+director
Class Diagram: Fellowship / FellowshipMonitor
Class Diagram: / RateLimitatio
<<typedef>> MACPacketClassification
+contributionCountTable
Table
(from PluginTrust)
empty()
size()
insert()
del()
retrieve()
update()
traverse()
retrieveList()
Class Diagram: Fellowship / Table
FellowshipDirector
contributionMetric_gamma : dou...
meritFactor_theta : double
demeritFactor_lambda : double
bandwidth : double
<<virtual>> recv(p : Packet*) : void
<<virtual>> xmit(p : Packet*) : void
CreateFellowship() : void
+monitor
Monitor
<<virtual>> recv(p : Packet*) : void
classify(p : Packet*) : MACPacketClassification
+rateLimitationTable
1

FIGURE A.1: Class Diagram of the Fellowship Model (*Left*).

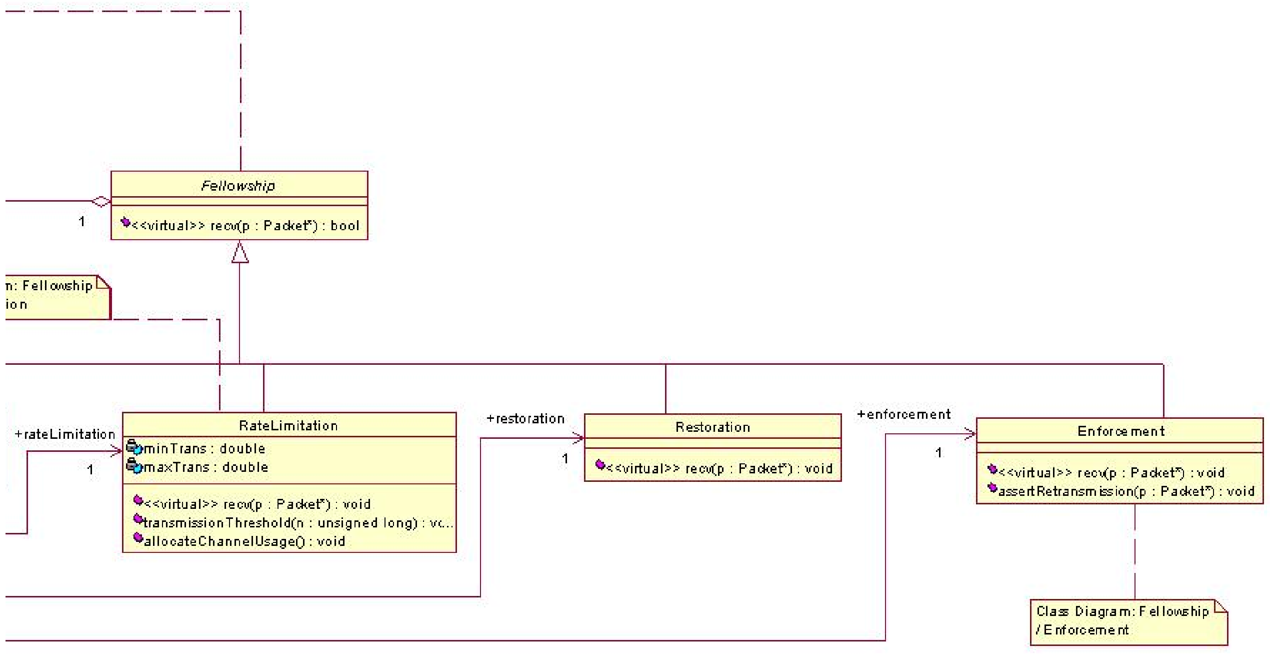

Fellowship
<<virtual>> recv(p : Packet*) : bool
1
n: Fellowship
ion
+rateLimitation
1
RateLimitation
minTrans : double
maxTrans : double
<<virtual>> recv(p : Packet*) : void
transmissionThreshold(n : unsigned long) : vo...
allocateChannelUsage() : void
+restoration
1
Restoration
<<virtual>> recv(p : Packet*) : void
+enforcement
1
Enforcement
<<virtual>> recv(p : Packet*) : void
assertRetransmission(p : Packet*) : void
Class Diagram: Fellowship / Enforcement

Figure A.2: Class Diagram of the Fellowship Model (*Right*).

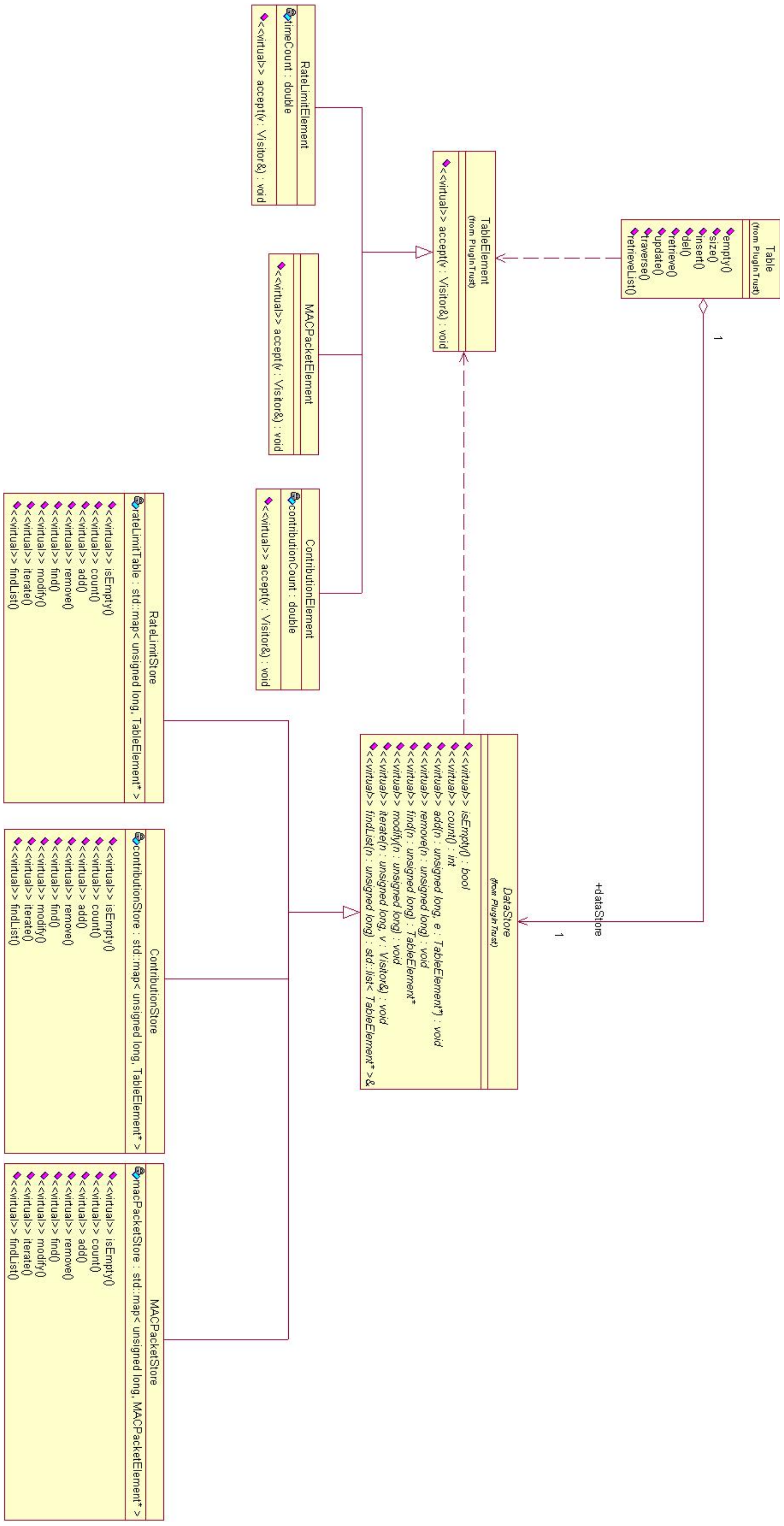


Figure A.3: Class Diagram of the Fellowship's Data Structures.

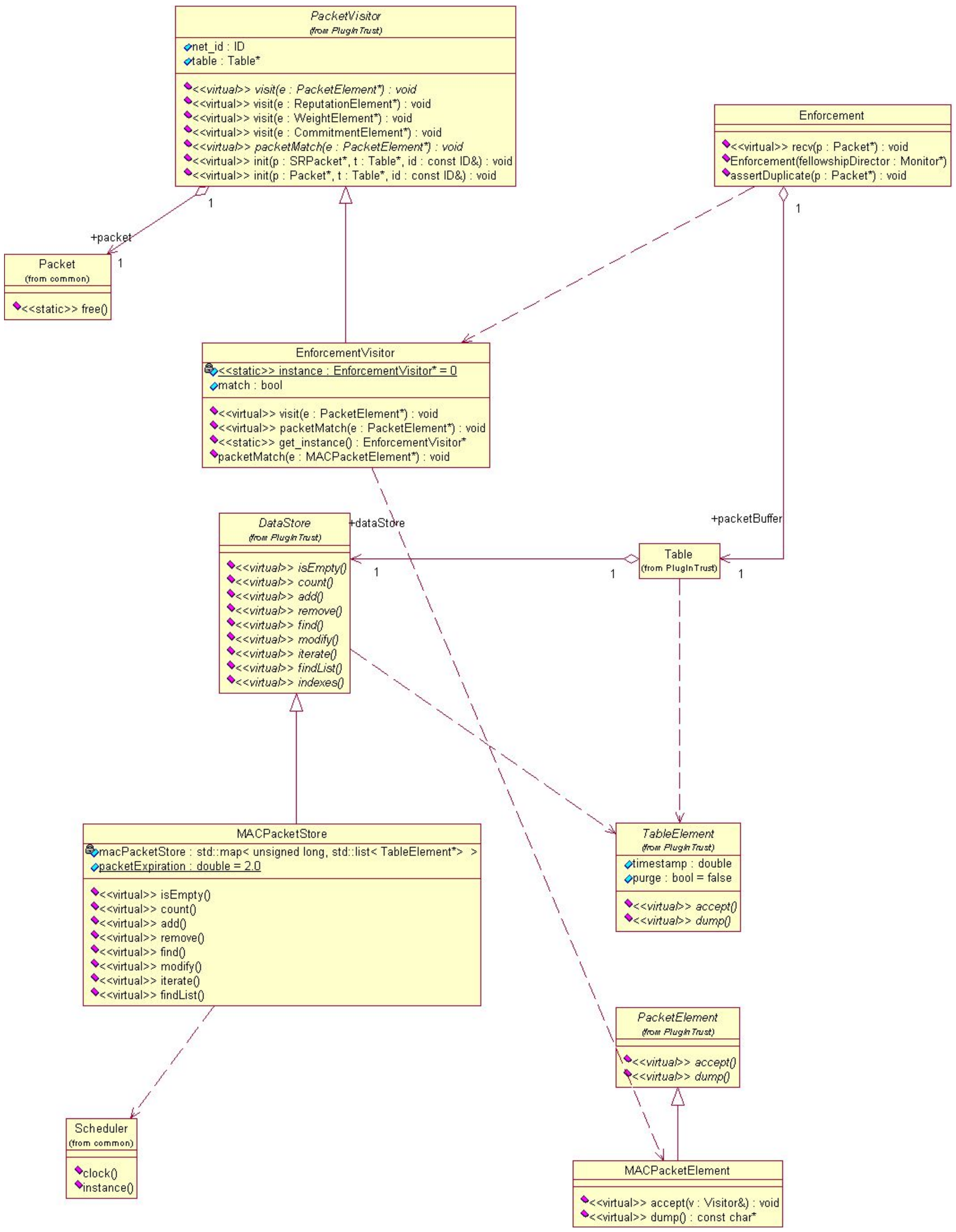

PacketVisitor
(from PlugIn Trust)
net_id : ID
table : Table*
<<virtual>> visit(e : PacketElement*) : void
<<virtual>> visit(e : ReputationElement*) : void
<<virtual>> visit(e : WeightElement*) : void
<<virtual>> visit(e : CommitmentElement*) : void
<<virtual>> packetMatch(e : PacketElement*) : void
<<virtual>> init(p : SRPacket*, t : Table*, id : const ID&) : void
<<virtual>> init(p : Packet*, t : Table*, id : const ID&) : void
Enforcement
<<virtual>> recv(p : Packet*) : void
Enforcement(fellowshipDirector : Monitor*)
assertDuplicate(p : Packet*) : void
1
+packet
1
Packet
(from common)
<<static>> free()
1
EnforcementVisitor
<<static>> instance : EnforcementVisitor* = 0
match : bool
<<virtual>> visit(e : PacketElement*) : void
<<virtual>> packetMatch(e : PacketElement*) : void
<<static>> get_instance() : EnforcementVisitor*
packetMatch(e : MACPacketElement*) : void
DataStore
(from PlugIn Trust)
+dataStore
+packetBuffer
<<virtual>> isEmpty()
<<virtual>> count()
<<virtual>> add()
<<virtual>> remove()
<<virtual>> find()
<<virtual>> modify()
<<virtual>> iterate()
<<virtual>> findList()
<<virtual>> indexes()
1
1
Table
(from PlugIn Trust)
1
MACPacketStore
macPacketStore : std::map< unsigned long, std::list< TableElement*> >
packetExpiration : double = 2.0
<<virtual>> isEmpty()
<<virtual>> count()
<<virtual>> add()
<<virtual>> remove()
<<virtual>> find()
<<virtual>> modify()
<<virtual>> iterate()
<<virtual>> findList()
TableElement
(from PlugIn Trust)
timestamp : double
purge : bool = false
<<virtual>> accept()
<<virtual>> dump()
PacketElement
(from PlugIn Trust)
<<virtual>> accept()
<<virtual>> dump()
Scheduler
(from common)
clock()
instance()
MACPacketElement
<<virtual>> accept(v : Visitor&) : void
<<virtual>> dump() : const char*

Figure A.4: Class Diagram of the Fellowship's Enforcement Component.

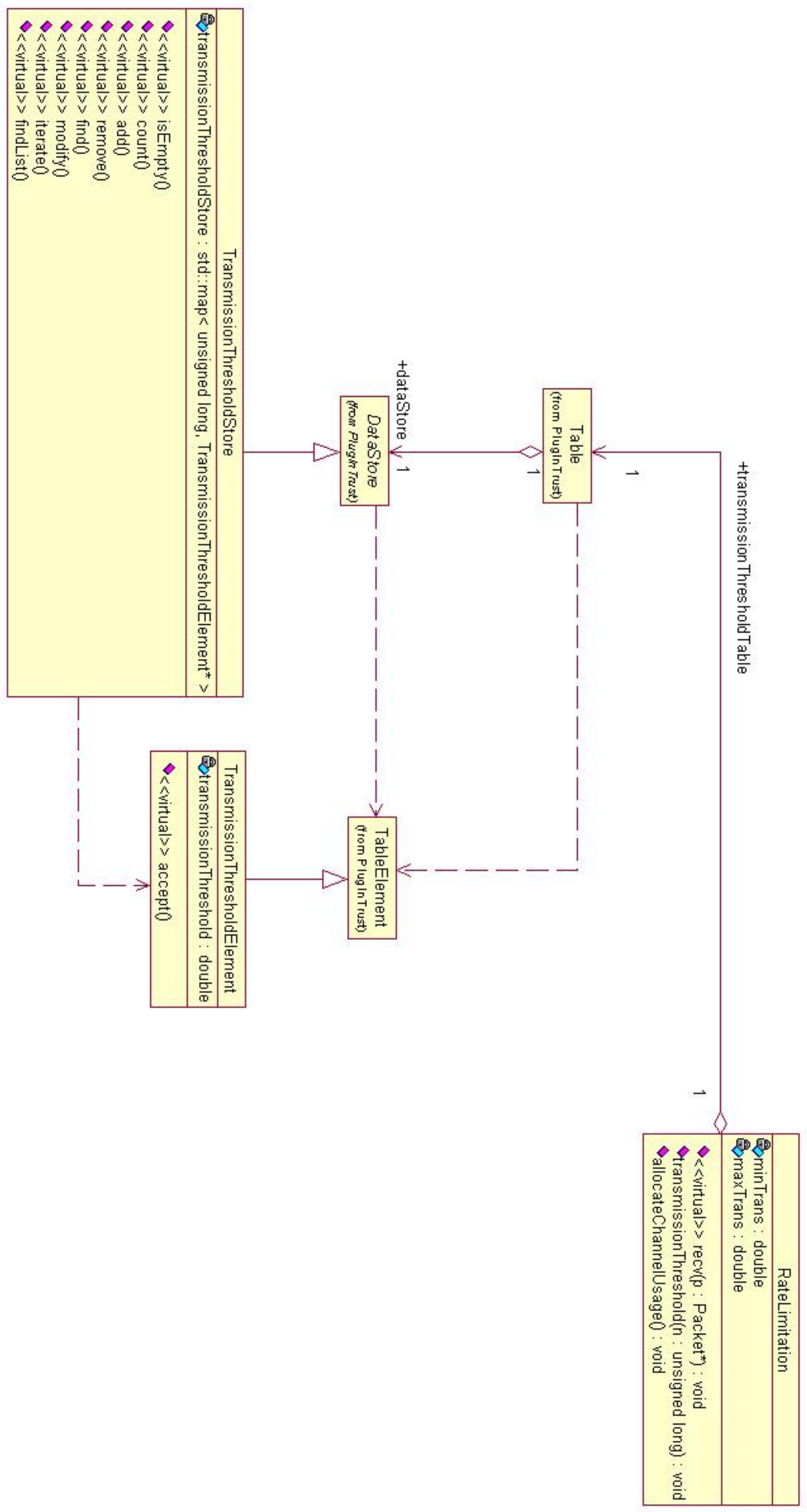


Figure A.5: Class Diagram of the Fellowship's Rate-limitation Component.

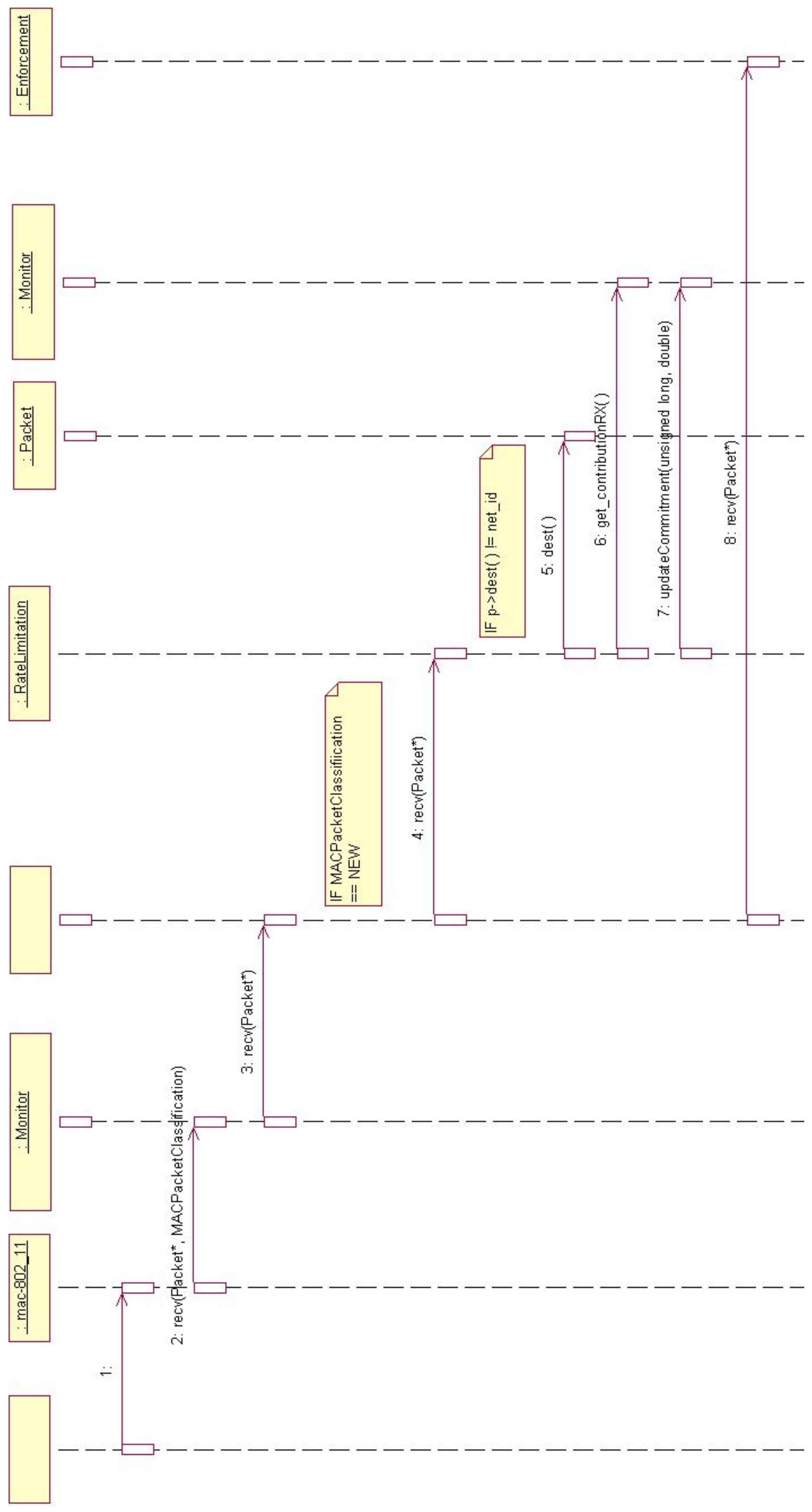


Figure A.6: Sequence Diagram for Processing a Packet by Rate-limitation.

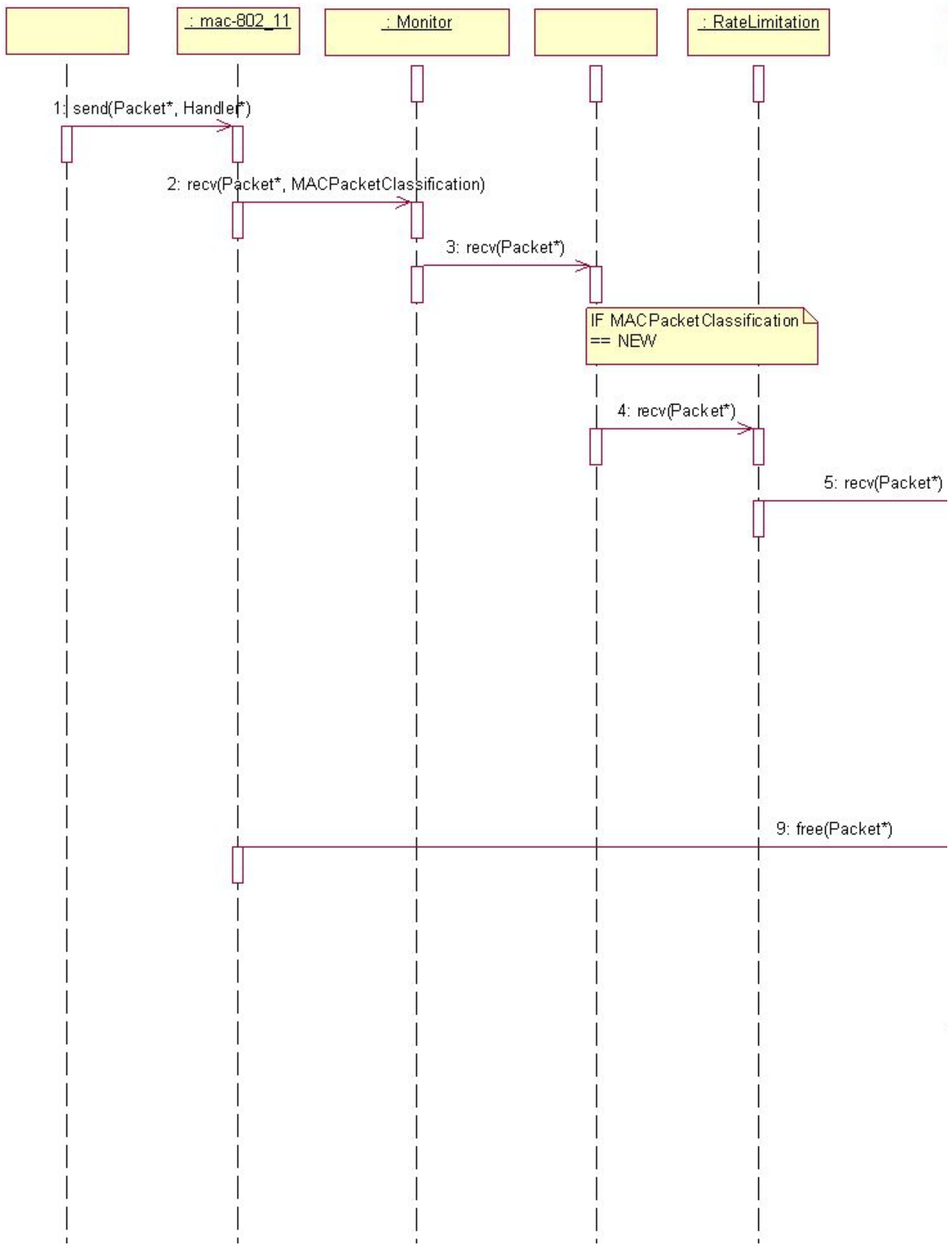


Figure A.7: Sequence Diagram for the Interactions between the Rate-limitation and Enforcement (*Left*).

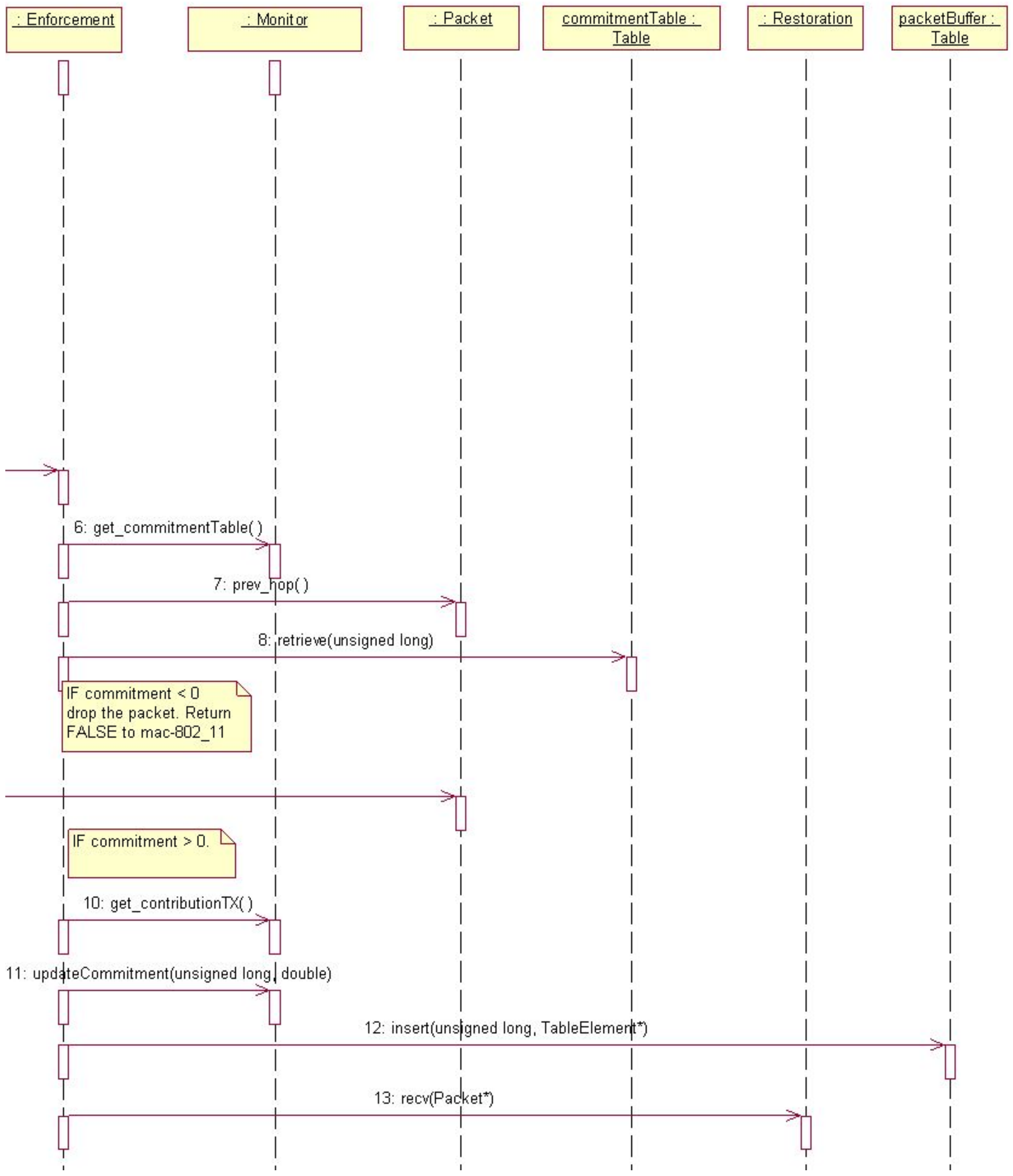

: Enforcement
: Monitor
: Packet
commitmentTable : Table
: Restoration
packetBuffer : Table
6: get_commitmentTable( )
7: prev_hop( )
8: retrieve(unsigned long)
IF commitment < 0 drop the packet. Return FALSE to mac-802_11
IF commitment > 0.
10: get_contributionTX( )
11: updateCommitment(unsigned long, double)
12: insert(unsigned long, TableElement*)
13: recv(Packet*)

Figure A.8: Sequence Diagram for the Interactions between the Rate-limitation and Enforcement (*Right*).

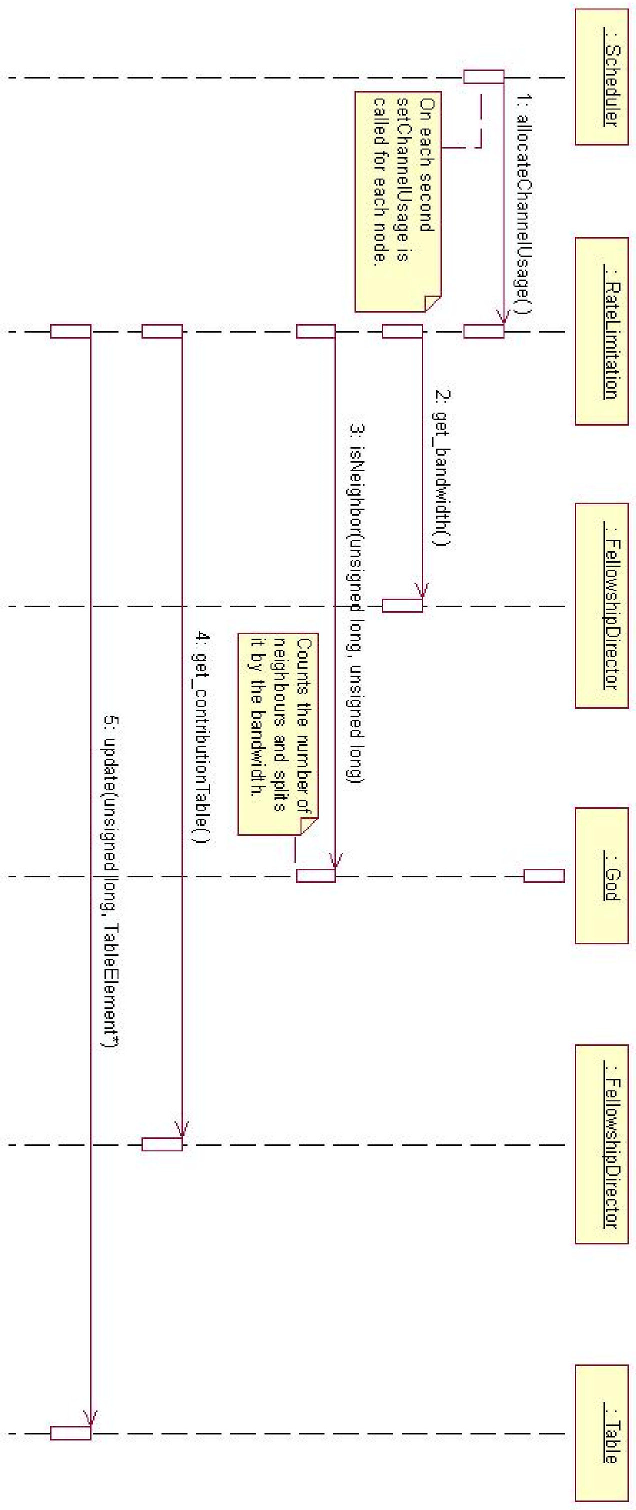

: Scheduler
: RateLimitation
: FellowshipDirector
: God
: FellowshipDirector
: Table
1: allocateChannelUsage( )
On each second setChannelUsage is called for each node.
2: get_bandwidth( )
3: isNeighbor(unsigned long, unsigned long)
Counts the number of neighbours and splits it by the bandwidth.
4: get_contributionTable( )
5: update(unsigned long, TableElement*)

Figure A.9: Sequence Diagram for the Channel Availability Computation by Rate-limitation.

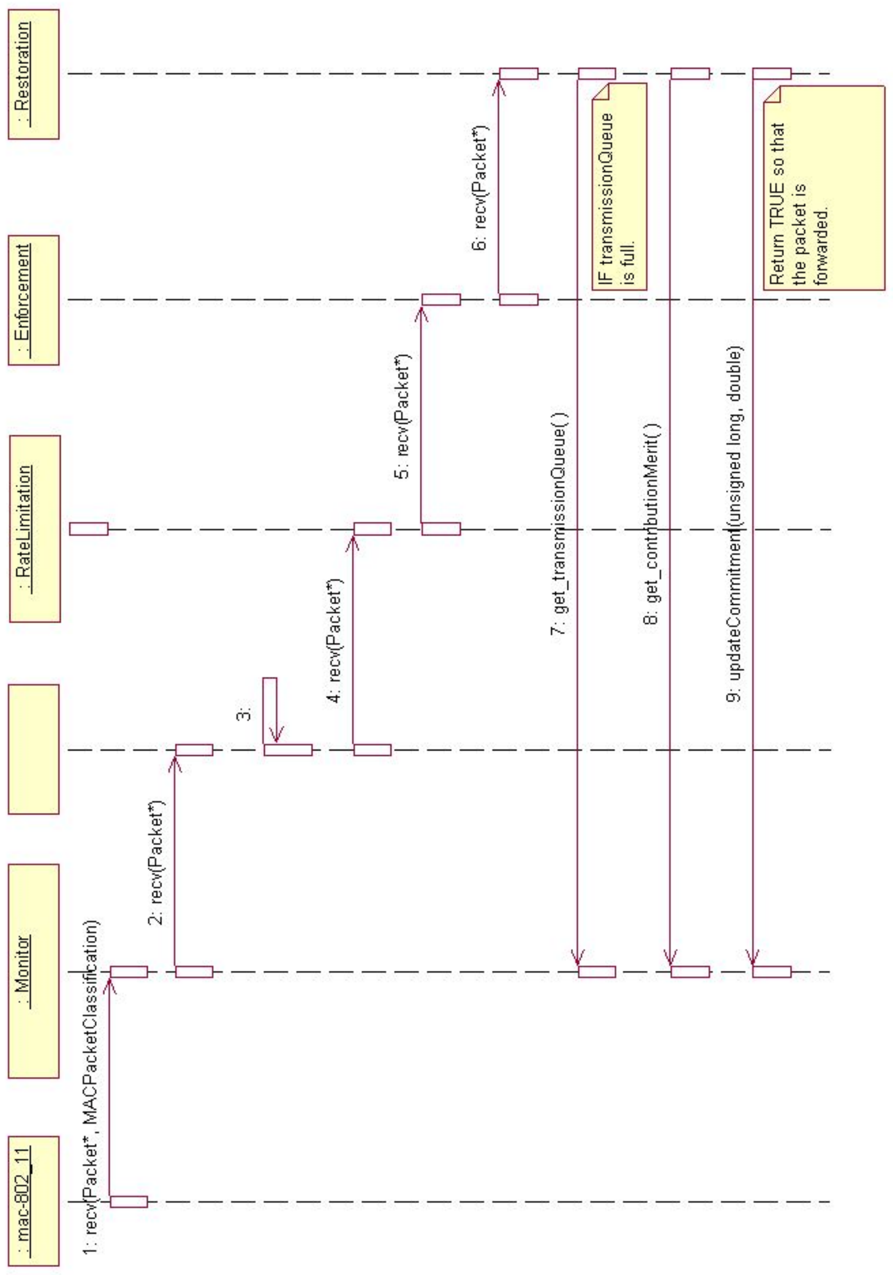

: mac-802_11
: Monitor
: RateLimitation
: Enforcement
: Restoration
1: recv(Packet*, MACPacketClassification)
2: recv(Packet*)
3:
4: recv(Packet*)
5: recv(Packet*)
6: recv(Packet*)
7: get_transmissionQueue( )
8: get_contributionMerit( )
9: updateCommitment(unsigned long, double)
IF transmissionQueue is full.
Return TRUE so that the packet is forwarded.

Figure A.10: Sequence Diagram for the Computations at the Restoration Component

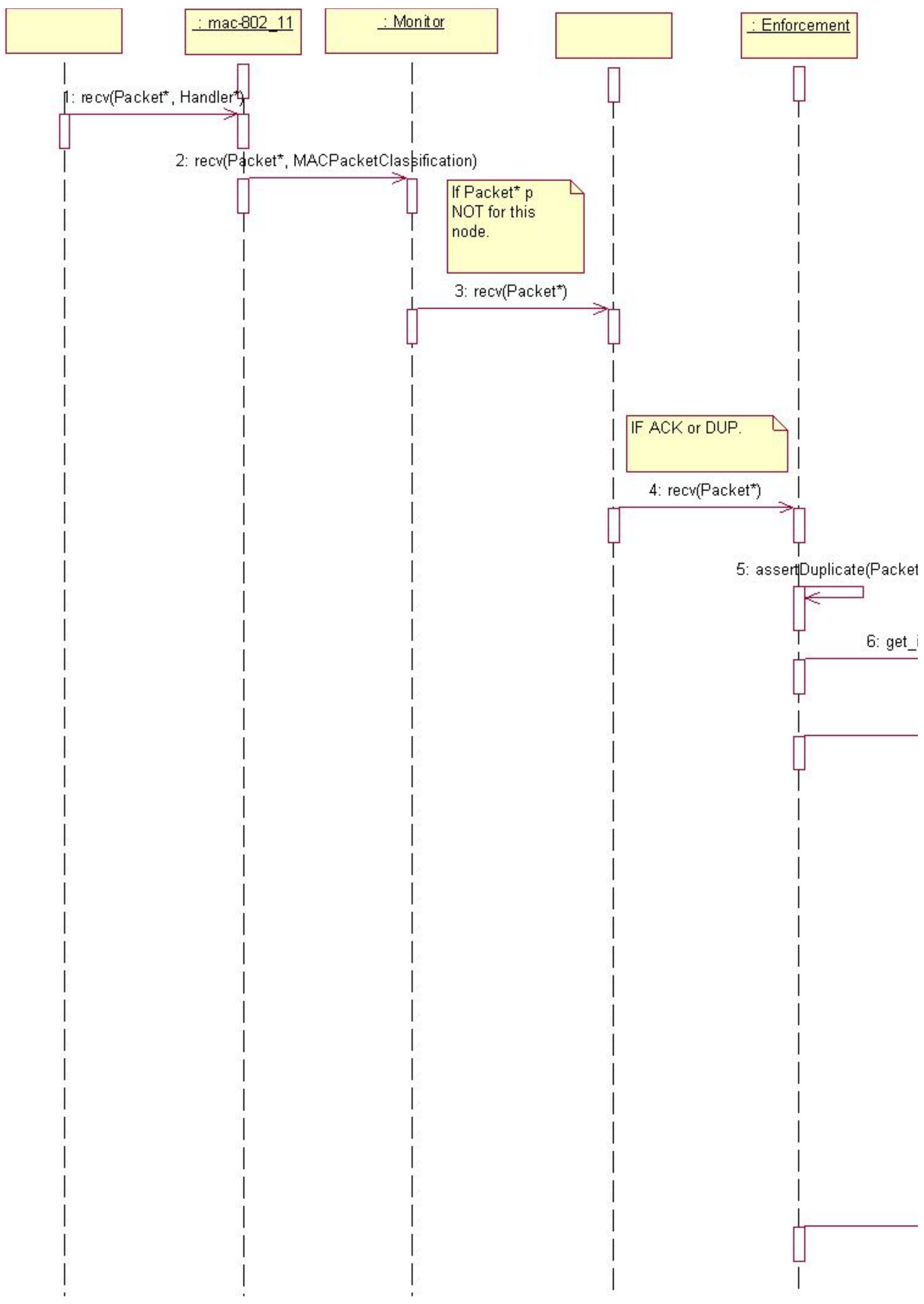

: mac-802_11
: Monitor
: Enforcement
1: recv(Packet*, Handler*)
2: recv(Packet*, MACPacketClassification)
If Packet* p NOT for this node.
3: recv(Packet*)
IF ACK or DUP.
4: recv(Packet*)
5: assertDuplicate(Packet
6: get_i

Figure A.11: Sequence Diagram for Processing the Acknowledgement and Duplicates by the Enforcement (*Left*).

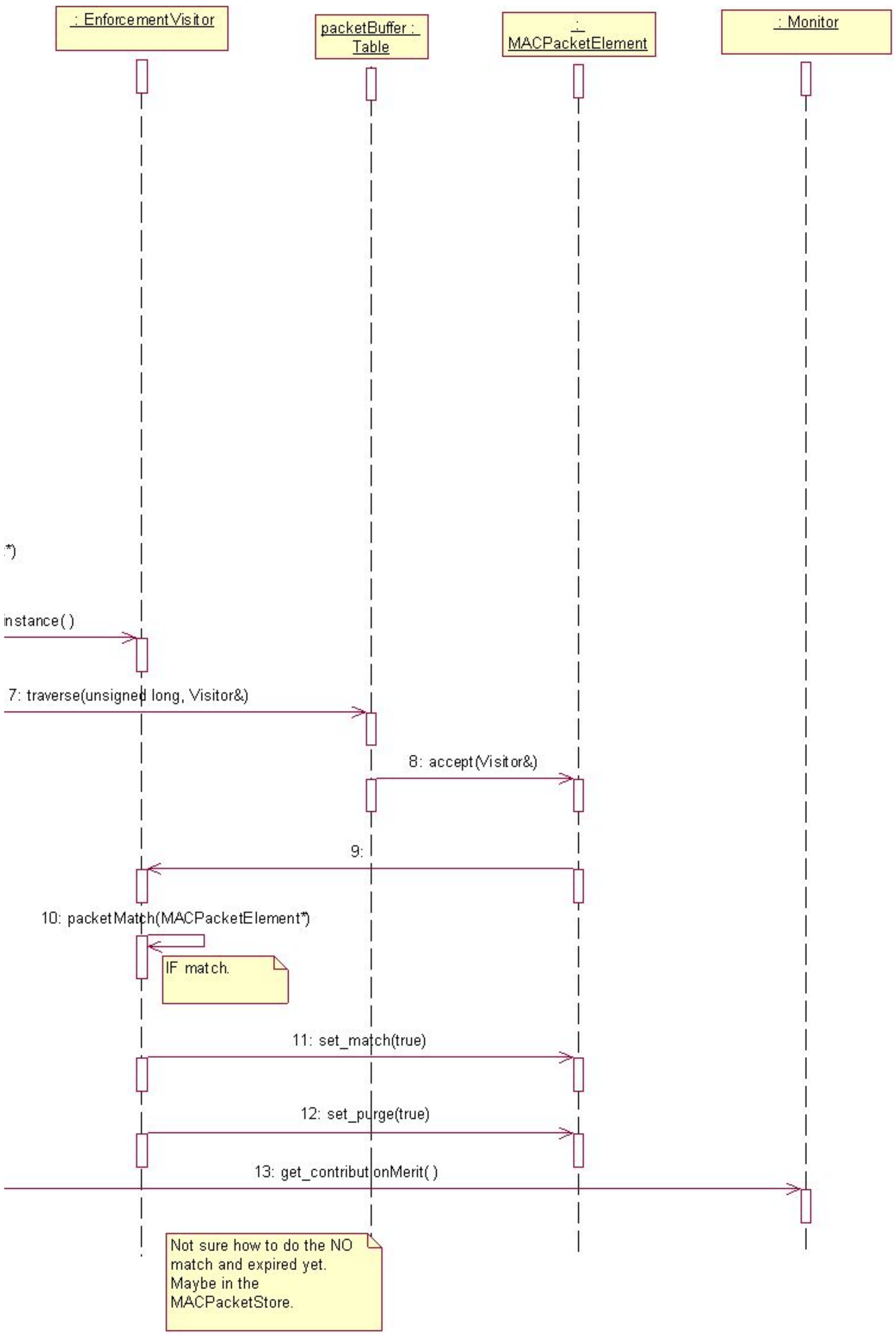

: EnforcementVisitor
packetBuffer : Table
: MACPacketElement
: Monitor
*)
nstance( )
7: traverse(unsigned long, Visitor&)
8: accept(Visitor&)
9:
10: packetMatch(MACPacketElement*)
IF match.
11: set_match(true)
12: set_purge(true)
13: get_contributionMerit( )
Not sure how to do the NO match and expired yet. Maybe in the MACPacketStore.

Figure A.12: Sequence Diagram for Processing the Acknowledgement and Duplicates by the Enforcement (*Right*).

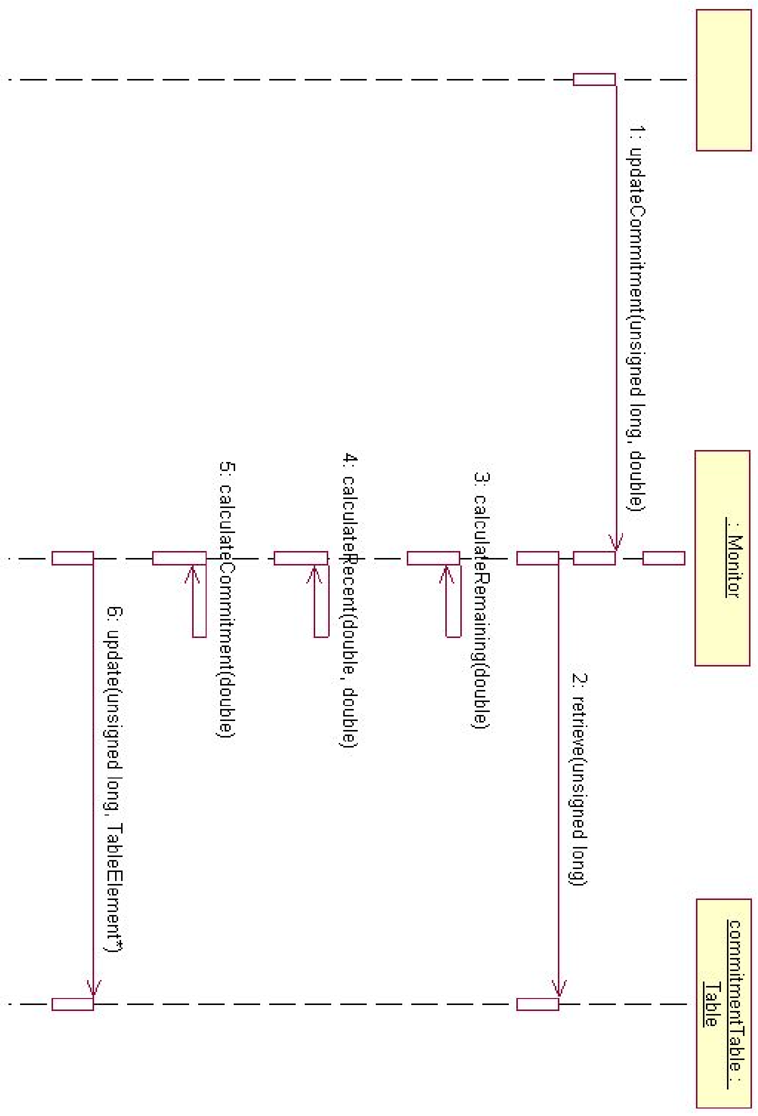

: Monitor
commitmentTable : Table
1: updateCommitment(unsigned long, double)
2: retrieve(unsigned long)
3: calculateRemaining(double)
4: calculateRecent(double, double)
5: calculateCommitment(double)
6: update(unsigned long, TableElement*)

Figure A.13: Sequence Diagram for Updating the Contributions

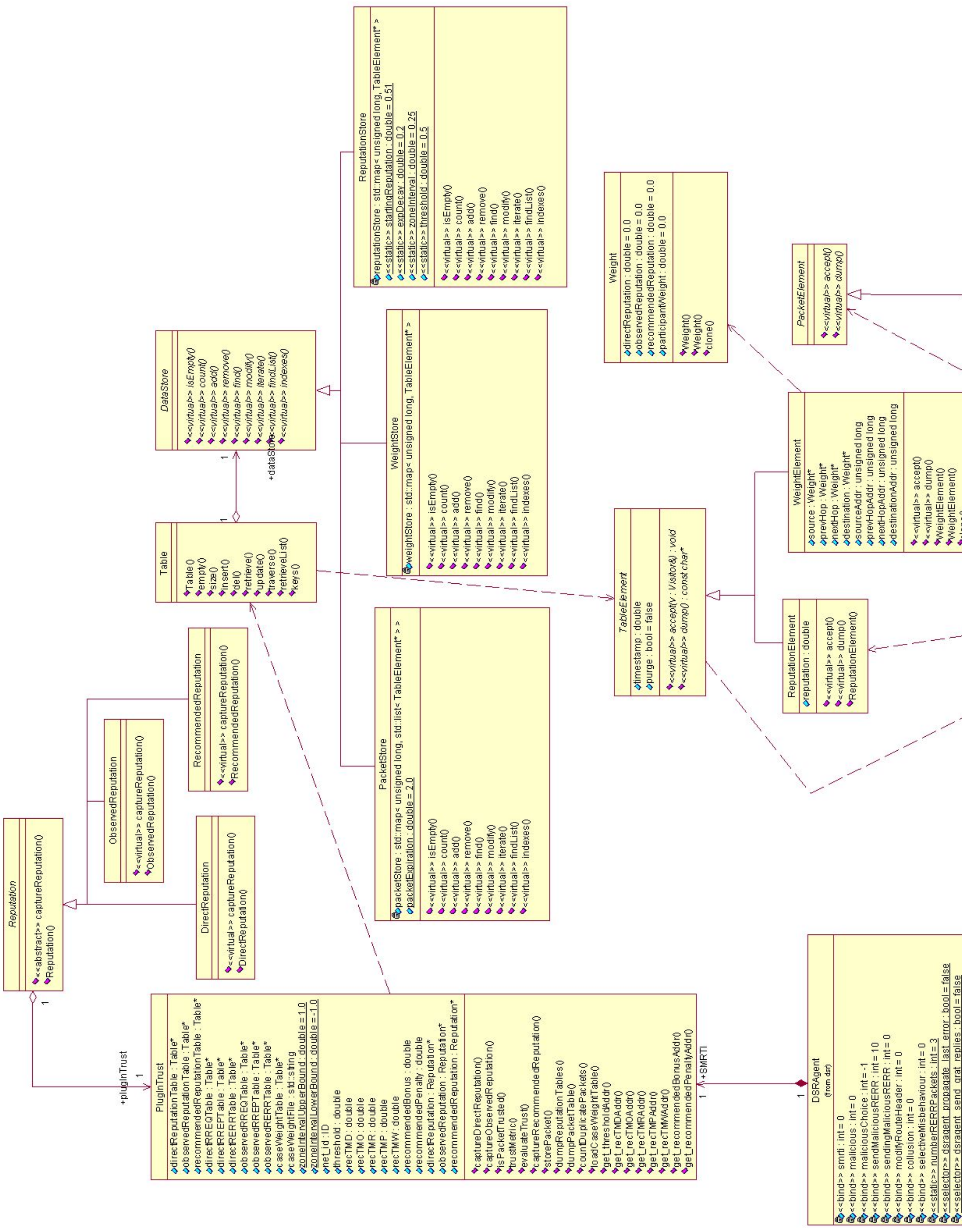

ReputationStore
reputationStore : std::map< unsigned long, TableElement* >
<<static>> startingReputation : double = 0.51
<<static>> expDecay : double = 0.2
<<static>> zoneInterval : double = 0.25
<<static>> threshold : double = 0.5
<<virtual>> isEmpty()
<<virtual>> count()
<<virtual>> add()
<<virtual>> remove()
<<virtual>> find()
<<virtual>> modify()
<<virtual>> iterate()
<<virtual>> findList()
<<virtual>> indexes()
DataStore
<<virtual>> isEmpty()
<<virtual>> count()
<<virtual>> add()
<<virtual>> remove()
<<virtual>> find()
<<virtual>> modify()
<<virtual>> iterate()
<<virtual>> findList()
<<virtual>> indexes()
WeightStore
weightStore : std::map< unsigned long, TableElement* >
Weight
directReputation : double = 0.0
observedReputation : double = 0.0
recommendedReputation : double = 0.0
participantWeight : double = 0.0
Weight()
Weight()
clone()
PacketElement
<<virtual>> accept()
<<virtual>> dump()
+dataStore
1
1
Table
Table()
empty()
size()
insert()
del()
retrieve()
update()
traverse()
retrieveList()
keys()
TableElement
timestamp : double
purge : bool = false
<<virtual>> accept(v : Visitor&) : void
<<virtual>> dump() : const char*
WeightElement
source : Weight*
prevHop : Weight*
nextHop : Weight*
destination : Weight*
sourceAddr : unsigned long
prevHopAddr : unsigned long
nextHopAddr : unsigned long
destinationAddr : unsigned long
<<virtual>> accept()
<<virtual>> dump()
WeightElement()
WeightElement()
ReputationElement
reputation : double
<<virtual>> accept()
<<virtual>> dump()
ReputationElement()
RecommendedReputation
<<virtual>> captureReputation()
RecommendedReputation()
ObservedReputation
<<virtual>> captureReputation()
ObservedReputation()
DirectReputation
<<virtual>> captureReputation()
DirectReputation()
Reputation
<<abstract>> captureReputation()
Reputation()
PacketStore
packetStore : std::map< unsigned long, std::list< TableElement* > >
packetExpiration : double = 2.0
1
+pluginTrust
1
PlugInTrust
directReputationTable : Table*
observedReputationTable : Table*
recommendedReputationTable : Table*
directRREQTable : Table*
directRREPTable : Table*
directRERRTable : Table*
observedRREQTable : Table*
observedRREPTable : Table*
observedRERRTable : Table*
caseWeightTable : Table*
caseWeightFile : std::string
zoneIntervalUpperBound : double = 1.0
zoneIntervalLowerBound : double = -1.0
net_id : ID
threshold : double
recTMD : double
recTMO : double
recTMR : double
recTMP : double
recTMW : double
recommendedBonus : double
recommendedPenalty : double
directReputation : Reputation*
observedReputation : Reputation*
recommendedReputation : Reputation*
captureDirectReputation()
captureObservedReputation()
isPacketTrusted()
trustMetric()
evaluateTrust()
captureRecommendedReputation()
storePacket()
dumpReputationTables()
dumpPacketTable()
countDuplicatePackets()
loadCaseWeightTable()
get_thresholdAddr()
get_recTMDAddr()
get_recTMOAddr()
get_recTMRAddr()
get_recTMPAddr()
get_recTMWAddr()
get_recommendedBonusAddr()
get_recommendedPenaltyAddr()
1
+SMRTI
1
DSRAgent
(from dsr)
<<bind>> smrti : int = 0
<<bind>> malicious : int = 0
<<bind>> maliciousChoice : int = -1
<<bind>> sendMaliciousRERR : int = 10
<<bind>> sendingMaliciousRERR : int = 0
<<bind>> modifyRouteHeader : int = 0
<<bind>> collusion : int = 0
<<bind>> selectiveMisbehaviour : int = 0
<<static>> numberRERRPackets : int = 3
<<selector>> dsragent_propagate_last_error : bool = false
<<selector>> dsragent_send_grat_replies : bool = false

FIGURE A.14: Class Diagram of the SMRTI Model (*Top*).

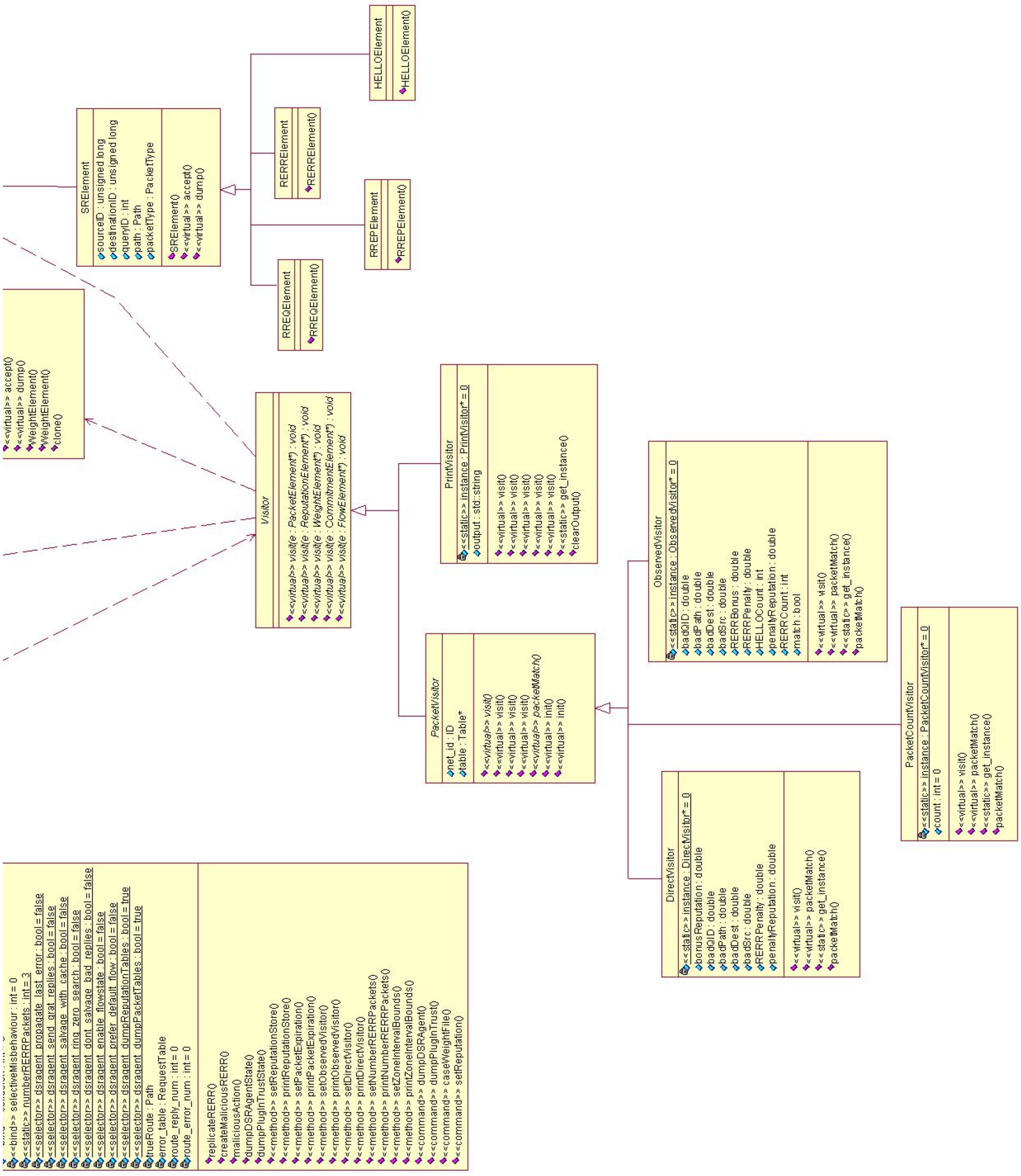

SRElement
sourceID : unsigned long
destinationID : unsigned long
queryID : int
path : Path
packetType : PacketType
SRElement()
<<virtual>> accept()
<<virtual>> dump()
HELLOElement
HELLOElement()
RERRElement
RERRElement()
RREPElement
RREPElement()
RREQElement
RREQElement()
<<virtual>> accept()
<<virtual>> dump()
WeightElement()
WeightElement()
clone()
Visitor
<<virtual>> visit(e : PacketElement*) : void
<<virtual>> visit(e : ReputationElement*) : void
<<virtual>> visit(e : WeightElement*) : void
<<virtual>> visit(e : CommitmentElement*) : void
<<virtual>> visit(e : FlowElement*) : void
PrintVisitor
<<static>> instance : PrintVisitor* = 0
output : std::string
<<virtual>> visit()
<<virtual>> visit()
<<virtual>> visit()
<<virtual>> visit()
<<virtual>> visit()
<<static>> get_instance()
clearOutput()
PacketVisitor
net_id : ID
table : Table*
<<virtual>> visit()
<<virtual>> visit()
<<virtual>> visit()
<<virtual>> visit()
<<virtual>> packetMatch()
<<virtual>> init()
<<virtual>> init()
ObservedVisitor
<<static>> instance : ObservedVisitor* = 0
badQID : double
badPath : double
badDest : double
badSrc : double
RERRBonus : double
RERRPenalty : double
HELLOCount : int
penaltyReputation : double
RERRCount : int
match : bool
<<virtual>> visit()
<<virtual>> packetMatch()
<<static>> get_instance()
packetMatch()
PacketCountVisitor
<<static>> instance : PacketCountVisitor* = 0
count : int = 0
<<virtual>> visit()
<<virtual>> packetMatch()
<<static>> get_instance()
packetMatch()
DirectVisitor
<<static>> instance : DirectVisitor* = 0
bonusReputation : double
badQID : double
badPath : double
badDest : double
badSrc : double
RERRPenalty : double
penaltyReputation : double
<<virtual>> visit()
<<virtual>> packetMatch()
<<static>> get_instance()
packetMatch()
<<bind>> selectiveMisbehaviour : int = 0
<<static>> numberRERRPackets : int = 3
<<selector>> dsragent_propagate_last_error : bool = false
<<selector>> dsragent_send_grat_replies : bool = false
<<selector>> dsragent_salvage_with_cache : bool = false
<<selector>> dsragent_ring_zero_search : bool = false
<<selector>> dsragent_dont_salvage_bad_replies : bool = false
<<selector>> dsragent_enable_flowstate : bool = false
<<selector>> dsragent_prefer_default_flow : bool = false
<<selector>> dsragent_dumpReputationTables : bool = true
<<selector>> dsragent_dumpPacketTables : bool = true
trueRoute : Path
error_table : RequestTable
route_reply_num : int = 0
route_error_num : int = 0
replicateRERR()
createMaliciousRERR()
maliciousAction()
dumpDSRAgentState()
dumpPlugInTrustState()
<<method>> setReputationStore()
<<method>> printReputationStore()
<<method>> setPacketExpiration()
<<method>> printPacketExpiration()
<<method>> setObservedVisitor()
<<method>> printObservedVisitor()
<<method>> setDirectVisitor()
<<method>> printDirectVisitor()
<<method>> setNumberRERRPackets()
<<method>> printNumberRERRPackets()
<<method>> setZoneIntervalBounds()
<<method>> printZoneIntervalBounds()
<<command>> dumpDSRAgent()
<<command>> dumpPlugInTrust()
<<command>> caseWeightFile()
<<command>> setReputation()

Figure A.15: Class Diagram of the SMRTI Model (*Bottom*).

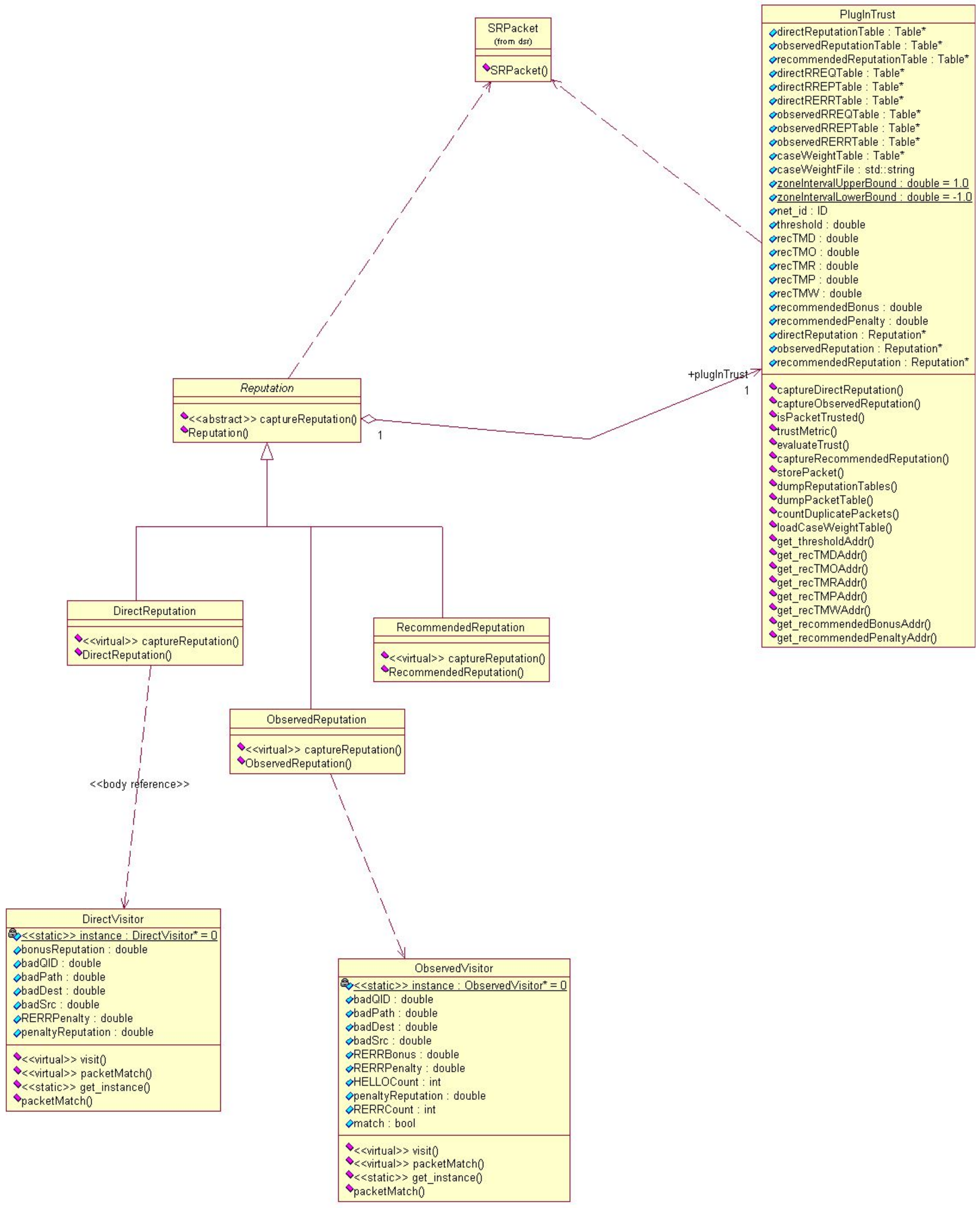


Figure A.16: Class Diagram of the Trust and Reputation Structures of the SMRTI.

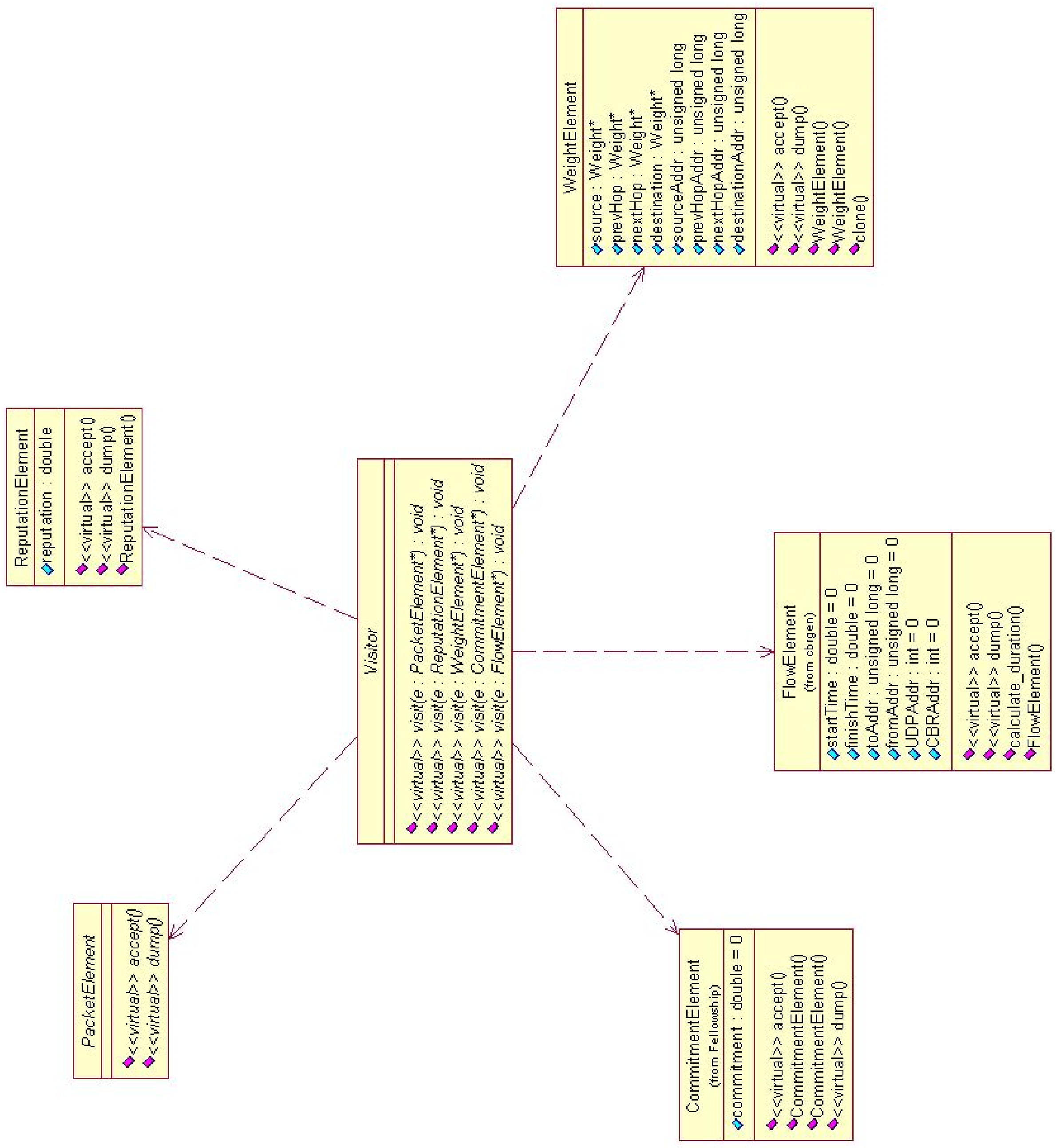
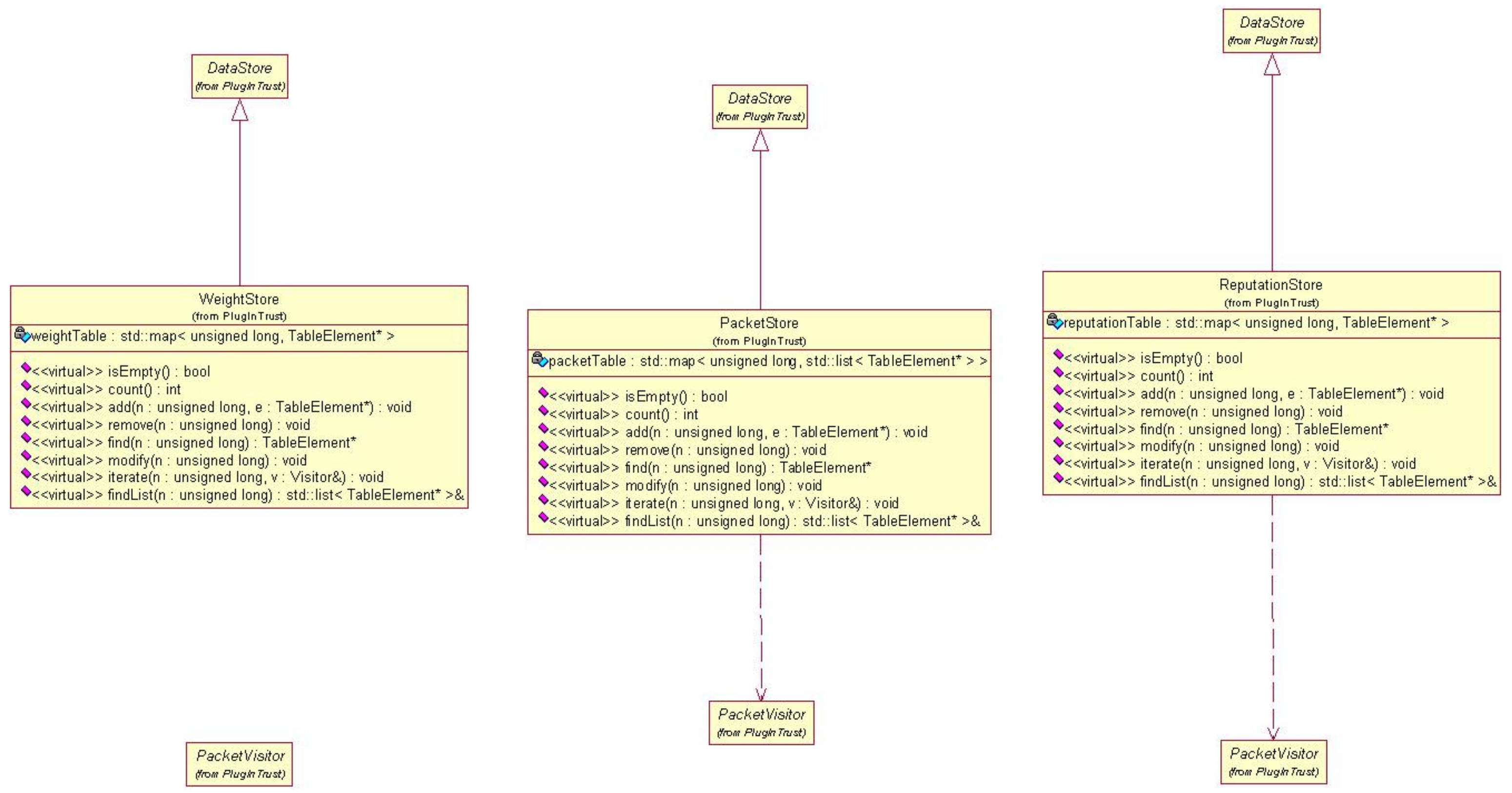


Figure A.17: Class Diagram for the Data Interfaces.


DataStore
(from PlugInTrust)
WeightStore
(from PlugInTrust)
weightTable : std::map< unsigned long, TableElement* >
<<virtual>> isEmpty() : bool
<<virtual>> count() : int
<<virtual>> add(n : unsigned long, e : TableElement*) : void
<<virtual>> remove(n : unsigned long) : void
<<virtual>> find(n : unsigned long) : TableElement*
<<virtual>> modify(n : unsigned long) : void
<<virtual>> iterate(n : unsigned long, v : Visitor&) : void
<<virtual>> findList(n : unsigned long) : std::list< TableElement* >&
PacketVisitor
(from PlugInTrust)
DataStore
(from PlugInTrust)
PacketStore
(from PlugInTrust)
packetTable : std::map< unsigned long, std::list< TableElement* > >
<<virtual>> isEmpty() : bool
<<virtual>> count() : int
<<virtual>> add(n : unsigned long, e : TableElement*) : void
<<virtual>> remove(n : unsigned long) : void
<<virtual>> find(n : unsigned long) : TableElement*
<<virtual>> modify(n : unsigned long) : void
<<virtual>> iterate(n : unsigned long, v : Visitor&) : void
<<virtual>> findList(n : unsigned long) : std::list< TableElement* >&
PacketVisitor
(from PlugInTrust)
DataStore
(from PlugInTrust)
ReputationStore
(from PlugInTrust)
reputationTable : std::map< unsigned long, TableElement* >
<<virtual>> isEmpty() : bool
<<virtual>> count() : int
<<virtual>> add(n : unsigned long, e : TableElement*) : void
<<virtual>> remove(n : unsigned long) : void
<<virtual>> find(n : unsigned long) : TableElement*
<<virtual>> modify(n : unsigned long) : void
<<virtual>> iterate(n : unsigned long, v : Visitor&) : void
<<virtual>> findList(n : unsigned long) : std::list< TableElement* >&
PacketVisitor
(from PlugInTrust)

(a) Weightstore (b) Packetstore (c) Reputationstore

Figure A.18: Class Diagram for the Data Structures.

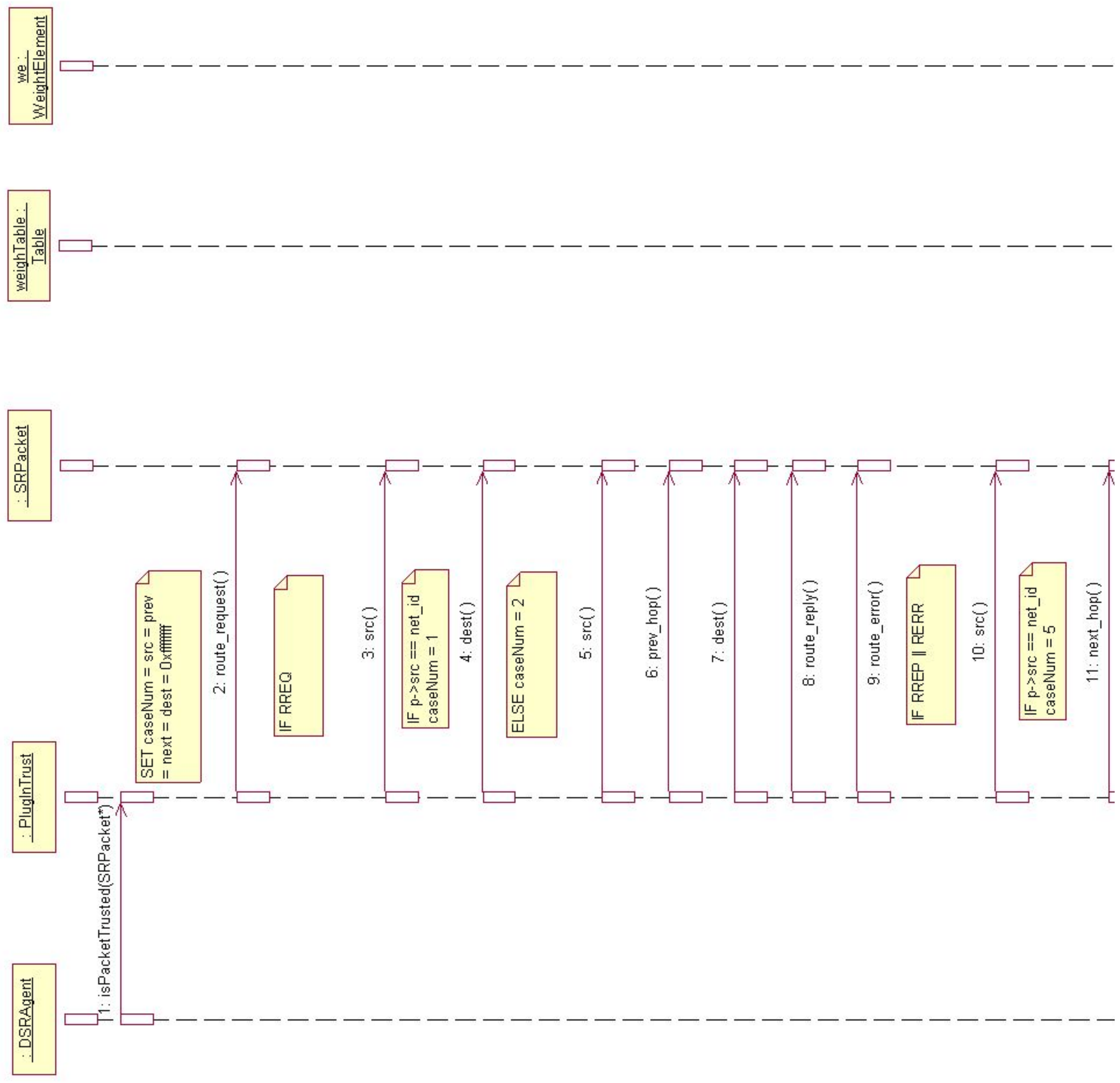


Figure A.19: Sequence Diagram for Determining the Trust of a Context by the SMRTI (*Top*).

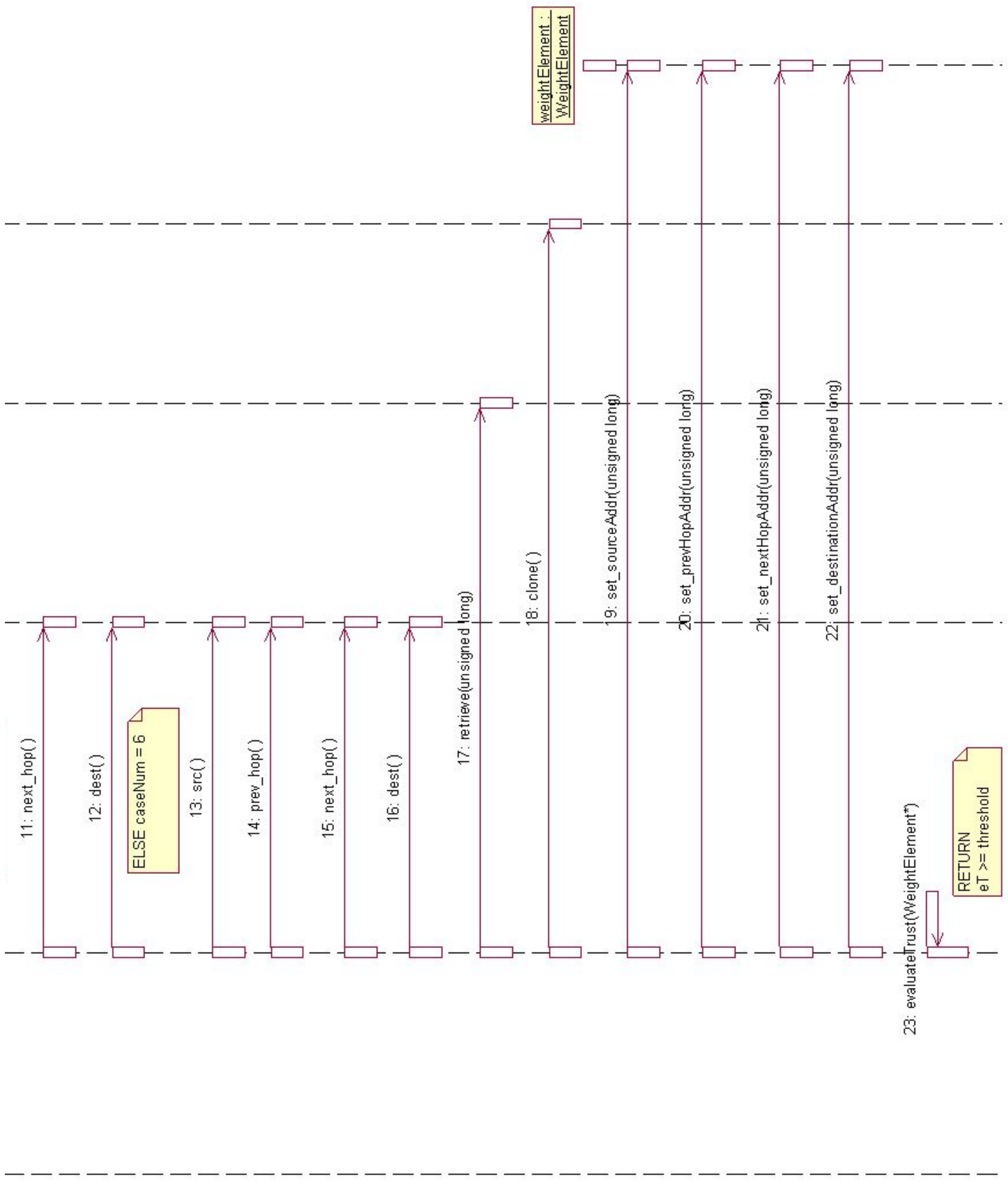

weightElement :
WeightElement
11: next_hop( )
12: dest( )
ELSE caseNum = 6
13: src( )
14: prev_hop( )
15: next_hop( )
16: dest( )
17: retrieve(unsigned long)
18: clone( )
19: set_sourceAddr(unsigned long)
20: set_prevHopAddr(unsigned long)
21: set_nextHopAddr(unsigned long)
22: set_destinationAddr(unsigned long)
23: evaluateTrust(WeightElement*)
RETURN
eT >= threshold

Figure A.20: Sequence Diagram for Determining the Trust of a Context by the SMRTI (*Bottom*).

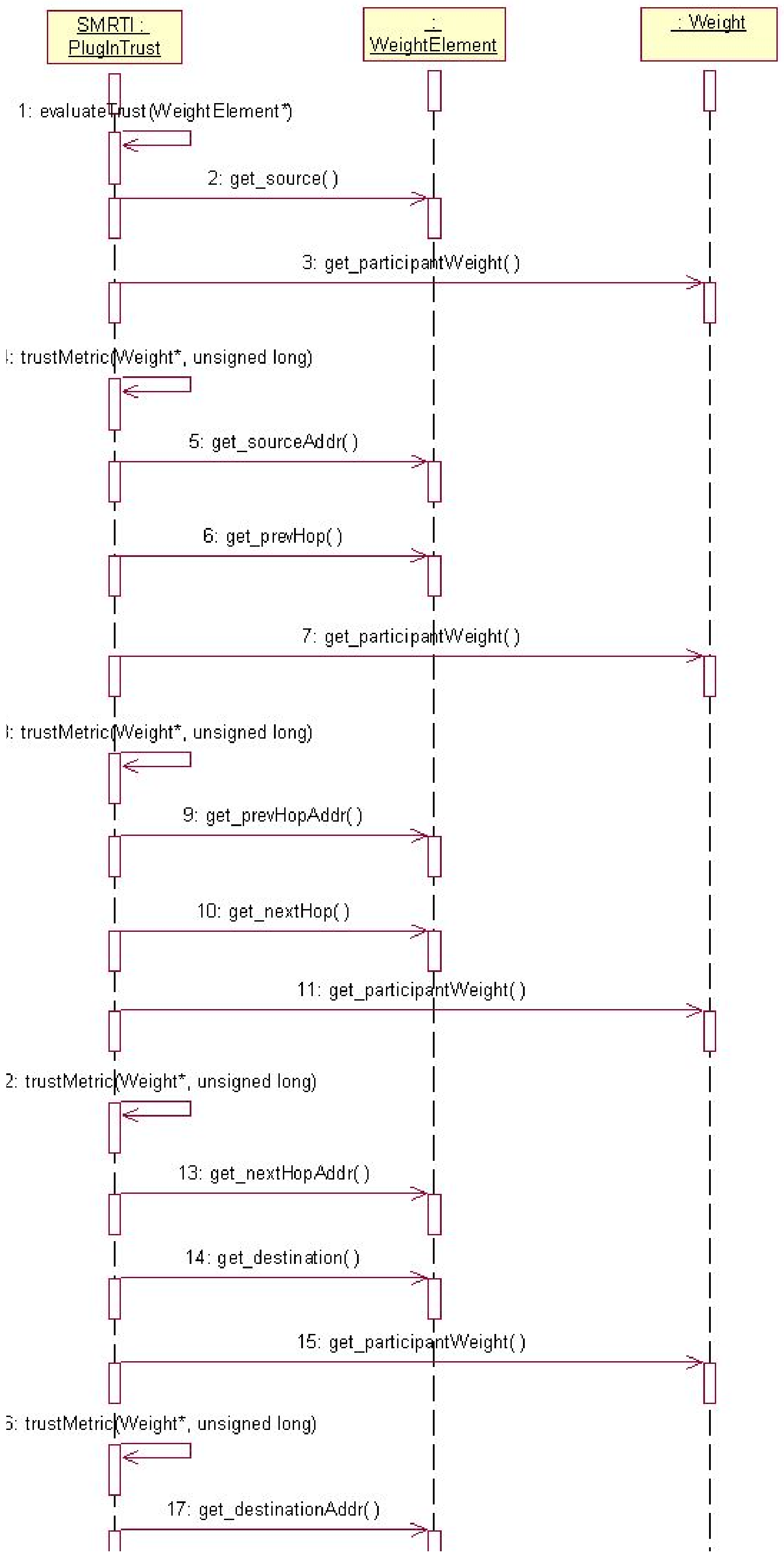

SMRTI : PlugInTrust
: WeightElement
: Weight
1: evaluateTrust(WeightElement*)
2: get_source( )
3: get_participantWeight( )
4: trustMetric(Weight*, unsigned long)
5: get_sourceAddr( )
6: get_prevHop( )
7: get_participantWeight( )
8: trustMetric(Weight*, unsigned long)
9: get_prevHopAddr( )
10: get_nextHop( )
11: get_participantWeight( )
12: trustMetric(Weight*, unsigned long)
13: get_nextHopAddr( )
14: get_destination( )
15: get_participantWeight( )
16: trustMetric(Weight*, unsigned long)
17: get_destinationAddr( )

Figure A.21: Sequence Diagram for the Trust Evaluation of a Packet by the SMRTI.

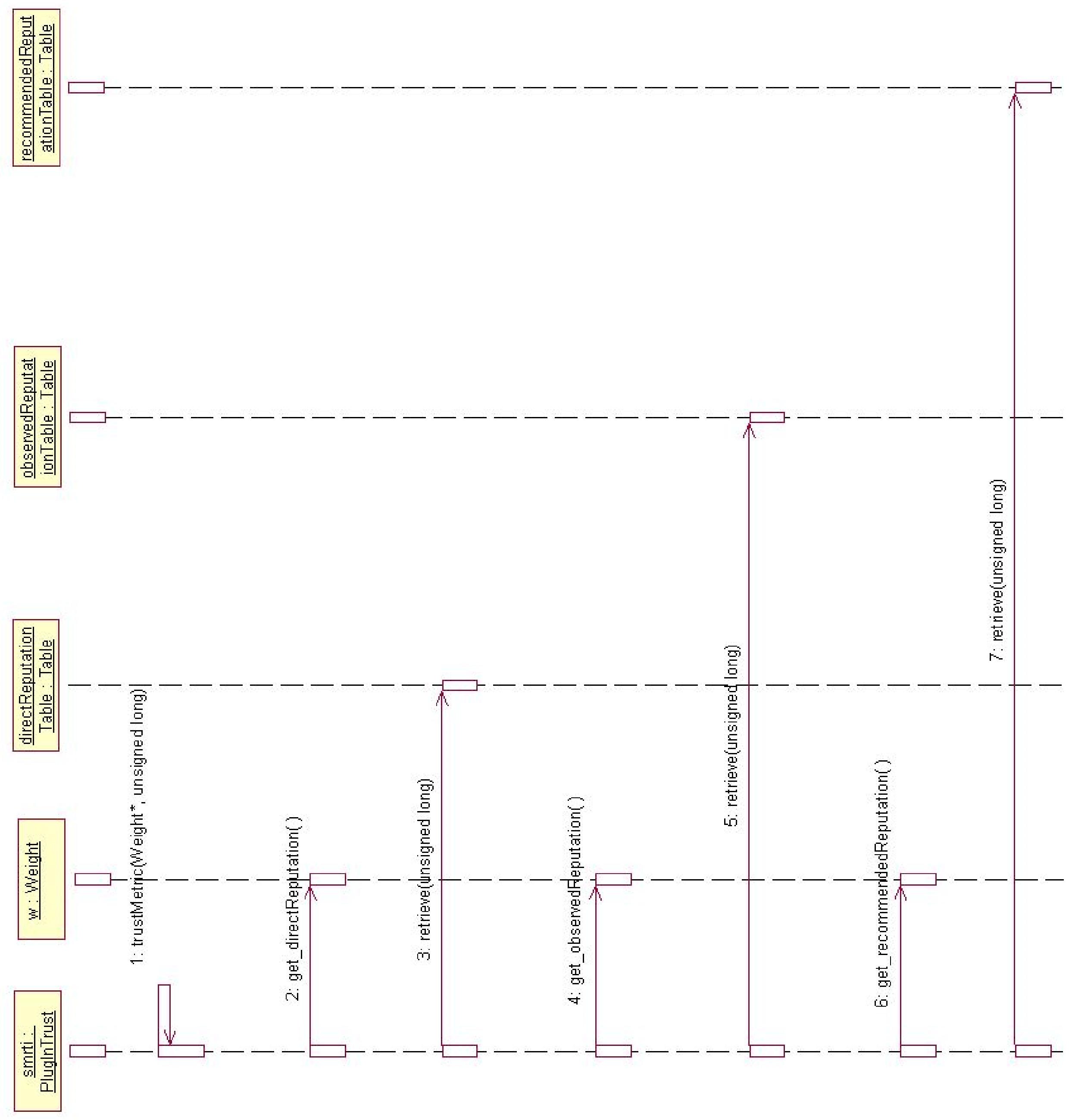

smrti : PlugInTrust
w : Weight
directReputation Table : Table
observedReputat ionTable : Table
recommendedReput ationTable : Table
1: trustMetric(Weight*, unsigned long)
2: get_directReputation( )
3: retrieve(unsigned long)
4: get_observedReputation( )
5: retrieve(unsigned long)
6: get_recommendedReputation( )
7: retrieve(unsigned long)

Figure A.22: Sequence Diagram for the Trust Evaluation of a Node by the SMRTI.

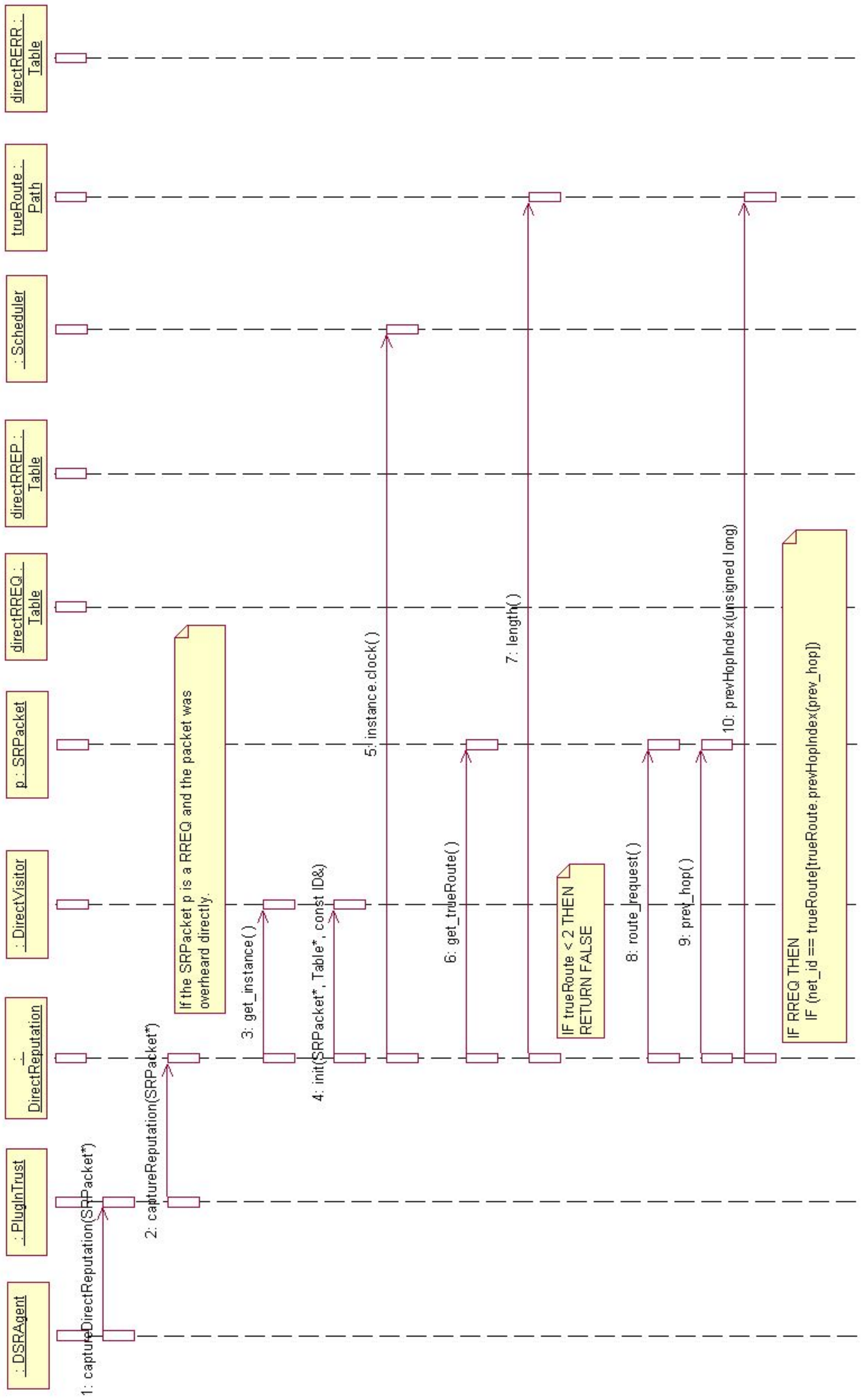


Figure A.23: Sequence Diagram for Capturing the Direct Reputation of a Next-hop (*Top*).

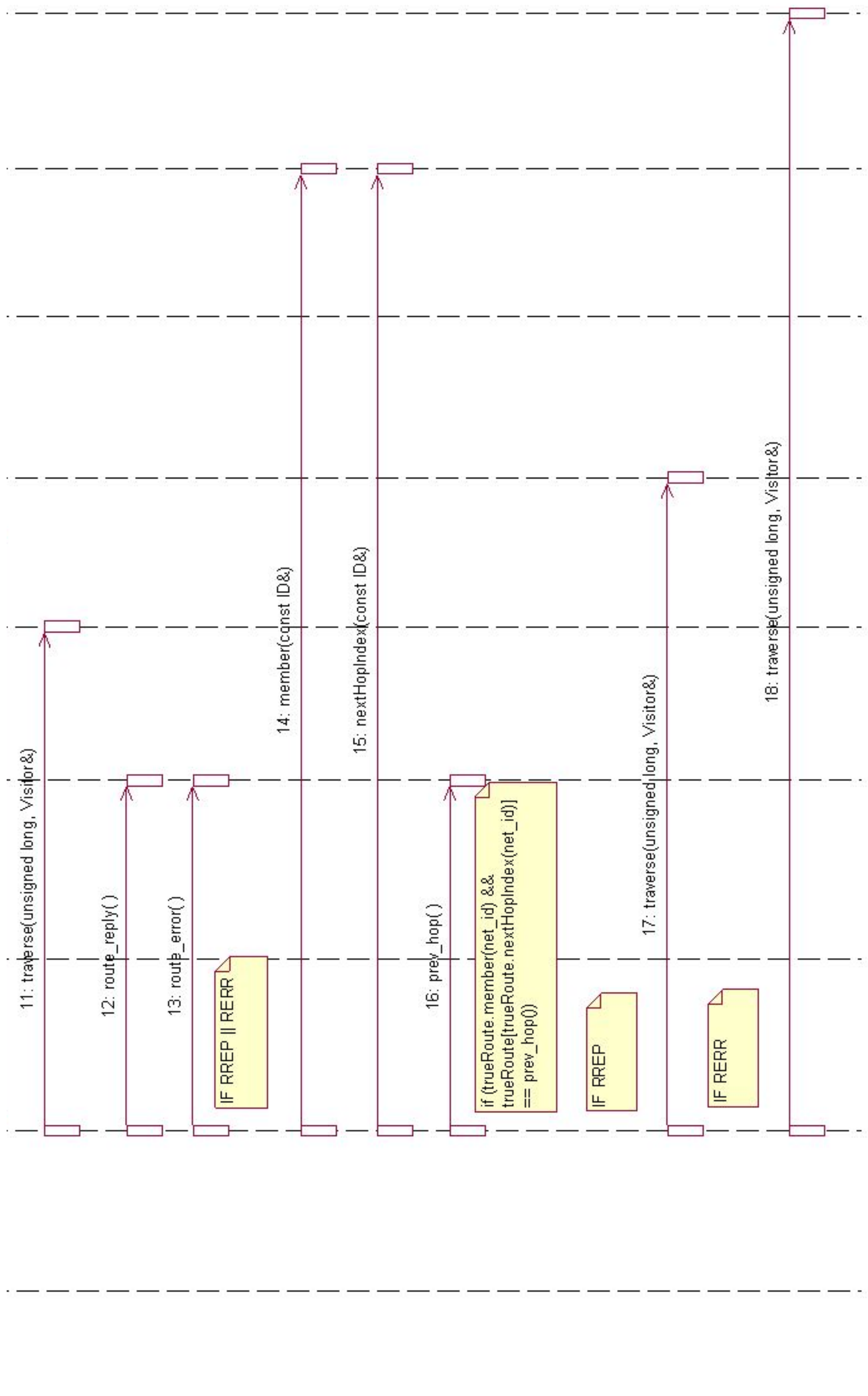

11: traverse(unsigned long, Visitor&)
12: route_reply( )
13: route_error( )
IF RREP || RERR
14: member(const ID&)
15: nextHopIndex(const ID&)
16: prev_hop( )
if (trueRoute.member(net_id) &&
trueRoute[trueRoute.nextHopIndex(net_id)]
== prev_hop())
IF RREP
17: traverse(unsigned long, Visitor&)
IF RERR
18: traverse(unsigned long, Visitor&)

Figure A.24: Sequence Diagram for Capturing the Direct Reputation of a Next-hop (*Bottom*).

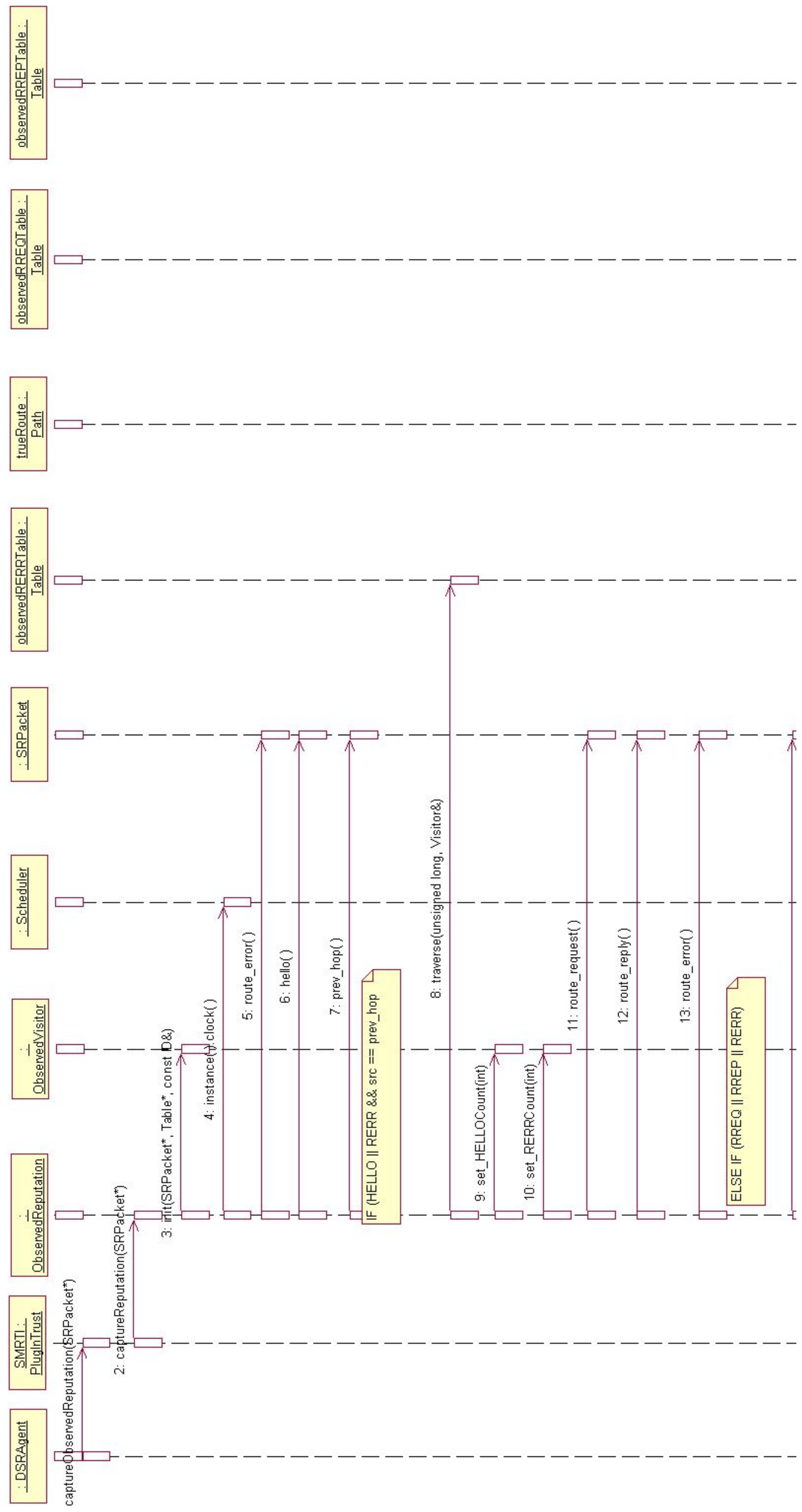

: DSRAgent
SMRTL : PlugInTrust
: ObservedReputation
: ObservedVisitor
: Scheduler
: SRPacket
observedRERRTable : Table
trueRoute : Path
observedRREQTable : Table
observedRREPTable : Table
captureObservedReputation(SRPacket*)
2: captureReputation(SRPacket*)
3: init(SRPacket*, Table*, const ID&)
4: instance( ).clock( )
5: route_error( )
6: hello( )
7: prev_hop( )
IF (HELLO || RERR && src == prev_hop
8: traverse(unsigned long, Visitor&)
9: set_HELLOCount(int)
10: set_RERRCount(int)
11: route_request( )
12: route_reply( )
13: route_error( )
ELSE IF (RREQ || RREP || RERR)

FIGURE A.25: Sequence Diagram for Capturing the Observed Reputation of a Next-hop (*Top*).

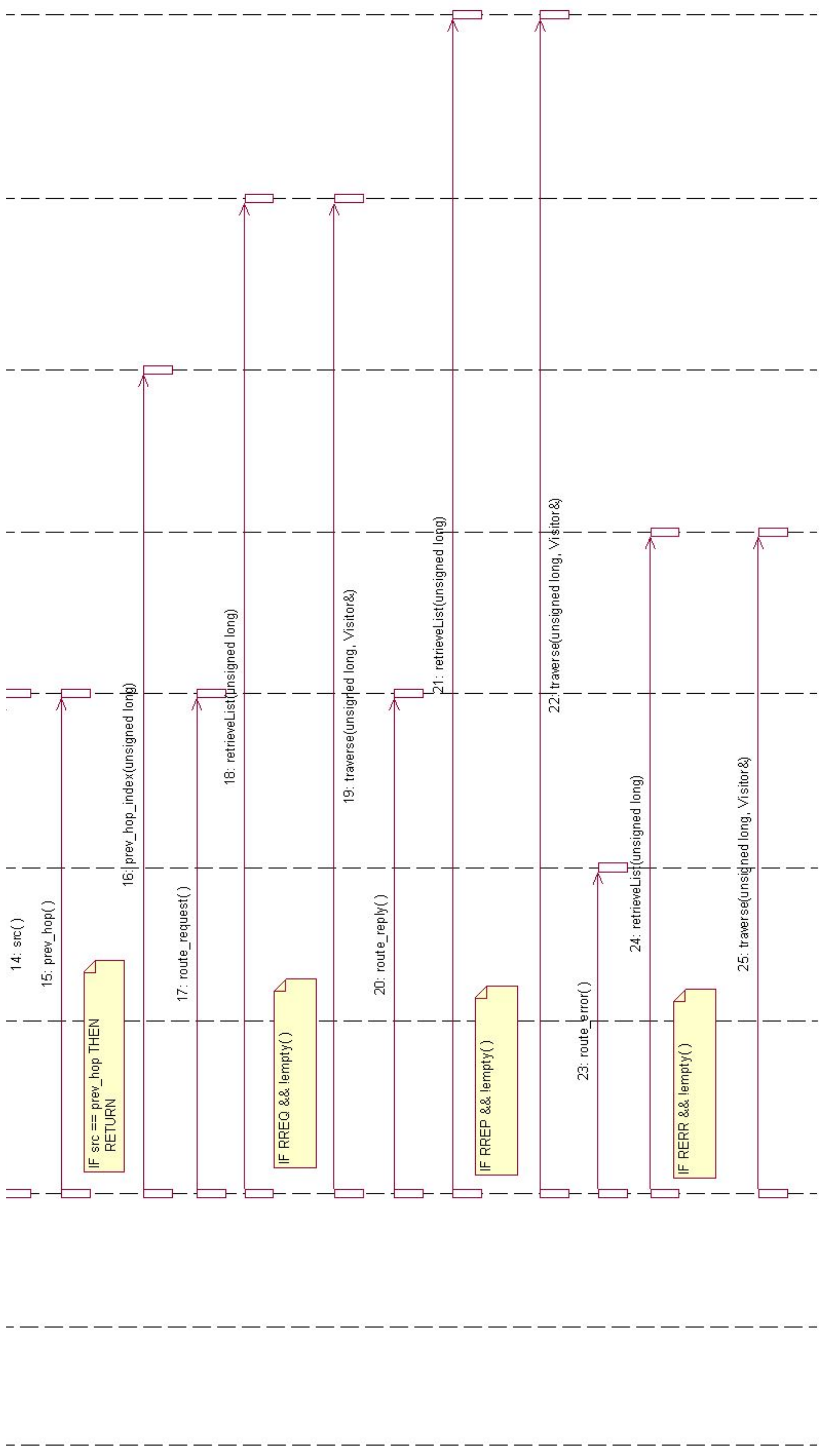

14: src( )
15: prev_hop( )
IF src == prev_hop THEN RETURN
16: prev_hop_index(unsigned long)
17: route_request( )
18: retrieveList(unsigned long)
IF RREQ && !empty( )
19: traverse(unsigned long, Visitor&)
20: route_reply( )
21: retrieveList(unsigned long)
IF RREP && !empty( )
22: traverse(unsigned long, Visitor&)
23: route_error( )
24: retrieveList(unsigned long)
IF RERR && !empty( )
25: traverse(unsigned long, Visitor&)

Figure A.26: Sequence Diagram for Capturing the Observed Reputation of a Next-hop (*Bottom*).

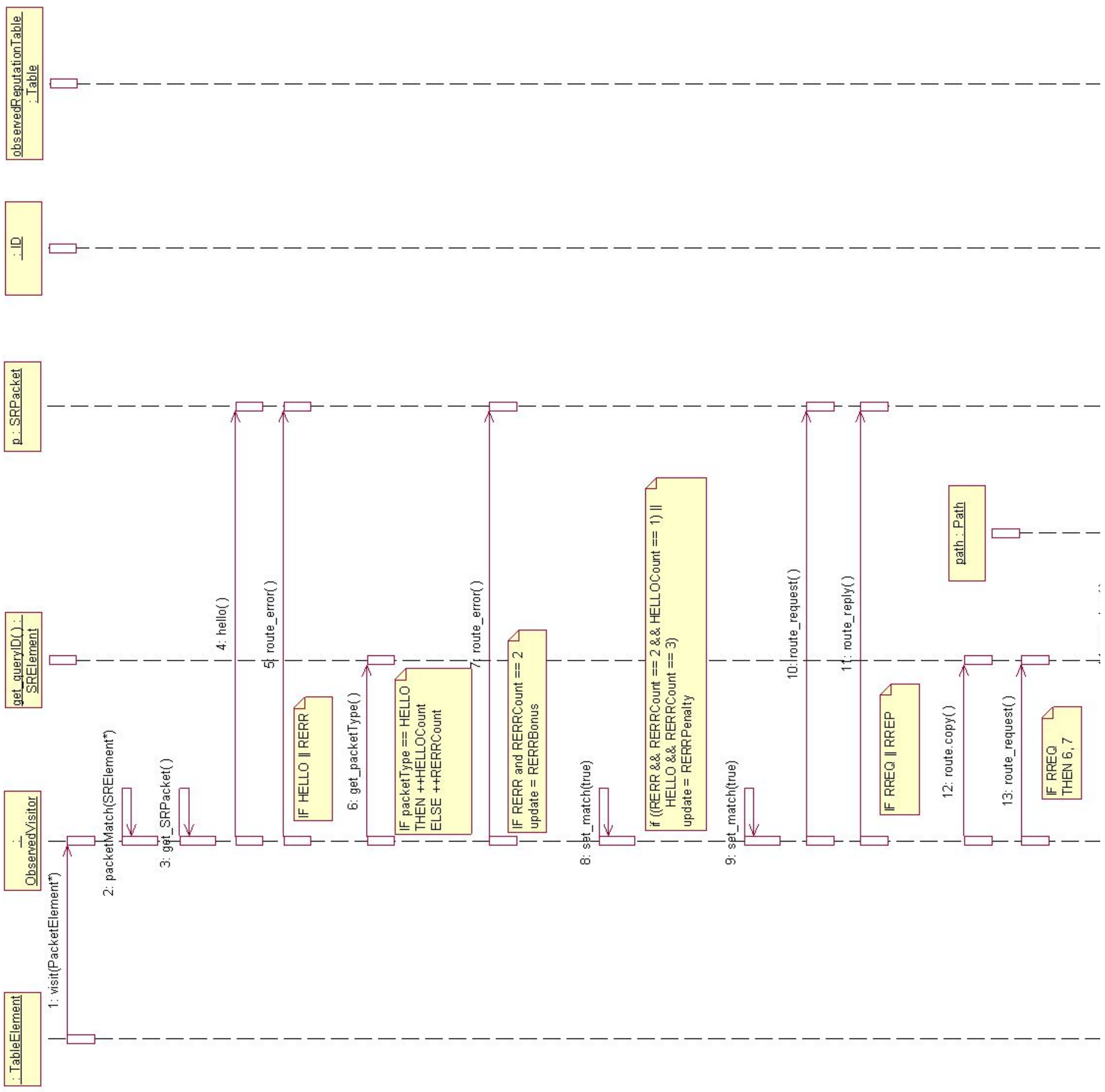


Figure A.27: Sequence Diagram for Traversing the Observed Reputation's Packet Buffer (*Top*).

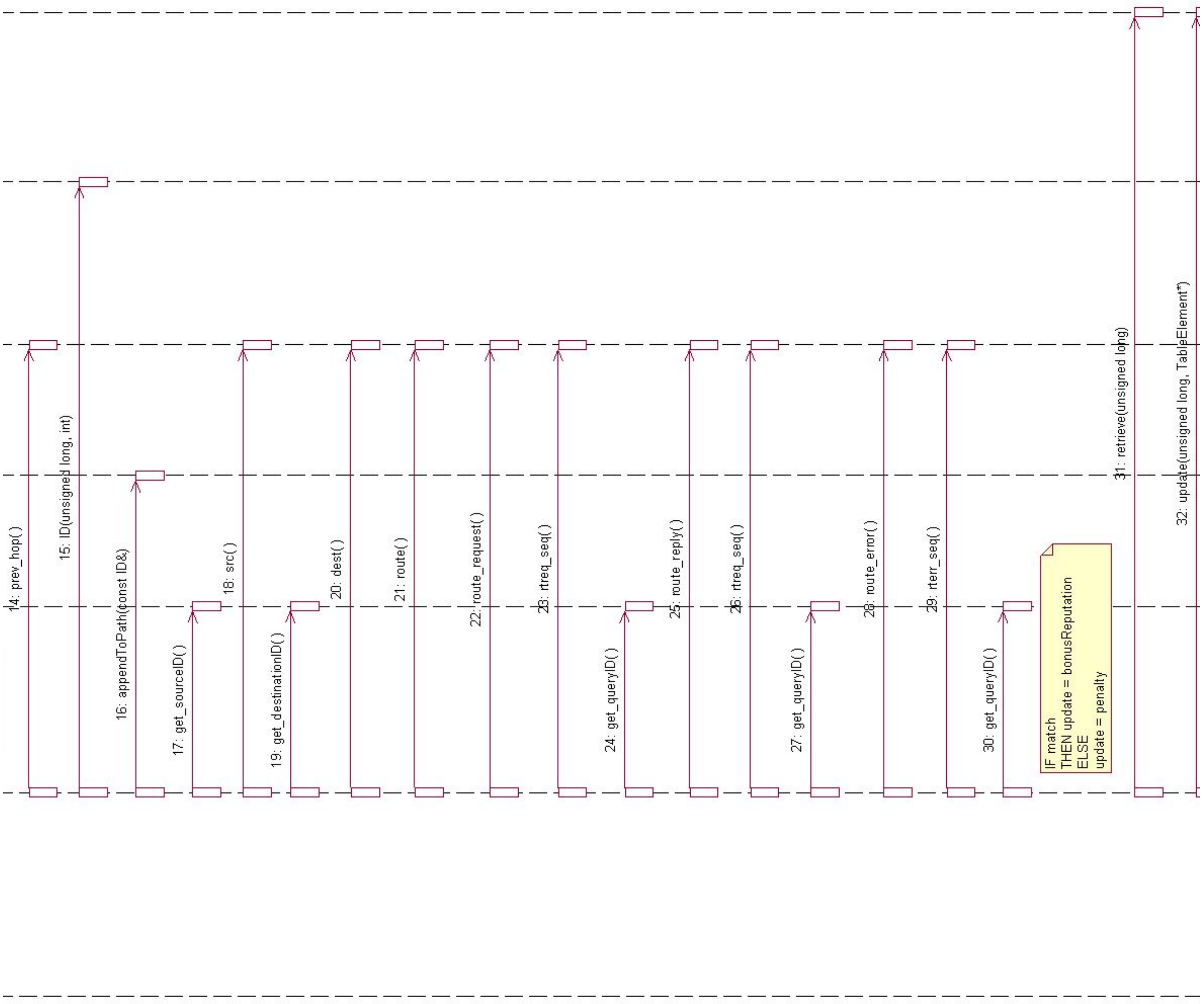


Figure A.28: Sequence Diagram for Traversing the Observed Reputation's Packet Buffer (*Bottom*).

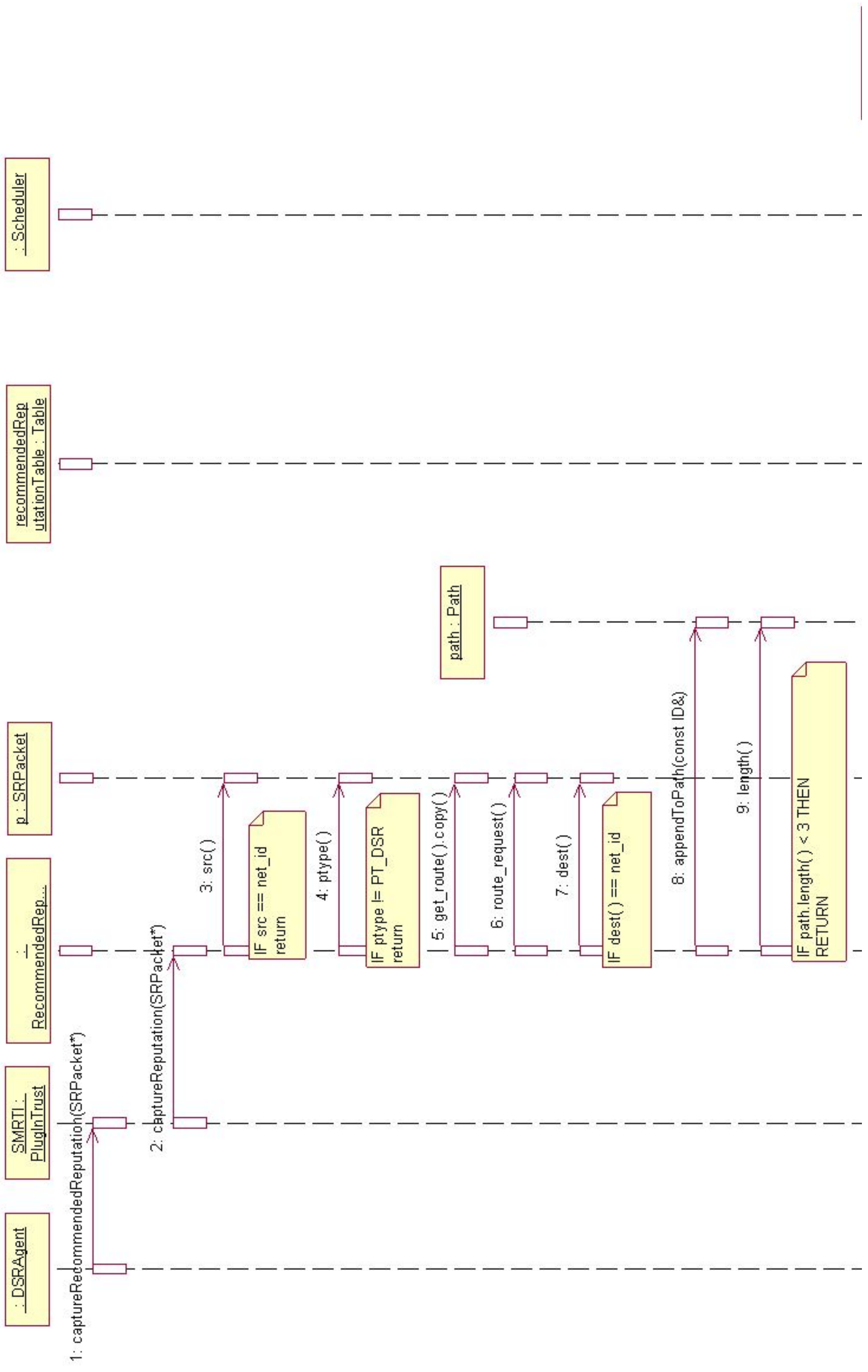


Figure A.29: Sequence Diagram for Capturing the Recommended Reputation (*Top*).

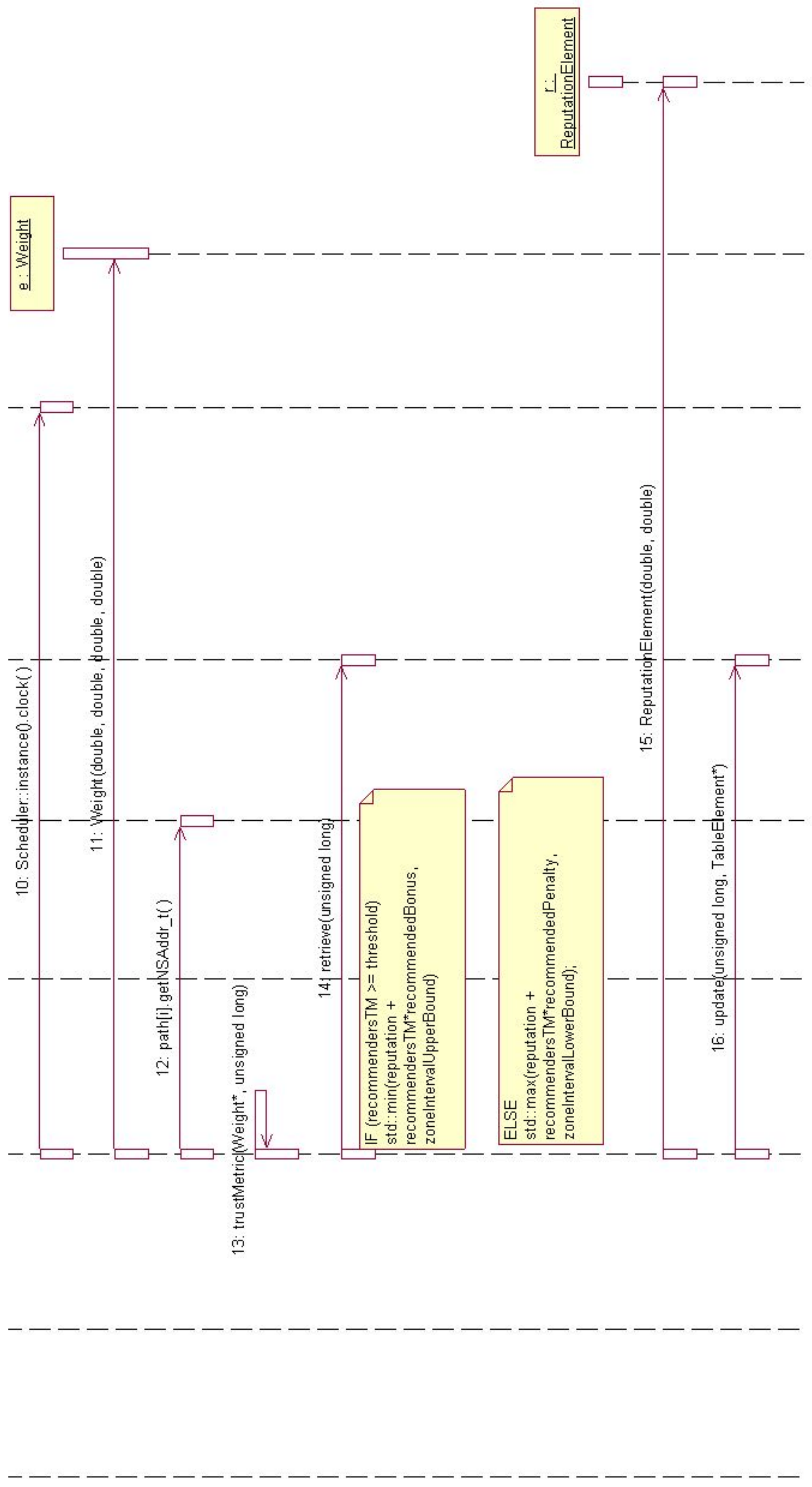

e : Weight
r : ReputationElement
10: Scheduler::instance().clock( )
11: Weight(double, double, double, double)
12: path[i].getNSAddr_t( )
13: trustMetric(Weight*, unsigned long)
14: retrieve(unsigned long)
IF (recommendersTM >= threshold)
std::min(reputation +
recommendersTM*recommendedBonus,
zoneIntervalUpperBound)
ELSE
std::max(reputation +
recommendersTM*recommendedPenalty,
zoneIntervalLowerBound);
15: ReputationElement(double, double)
16: update(unsigned long, TableElement*)

Figure A.30: Sequence Diagram for Capturing the Recommended Reputation (*Bottom*).

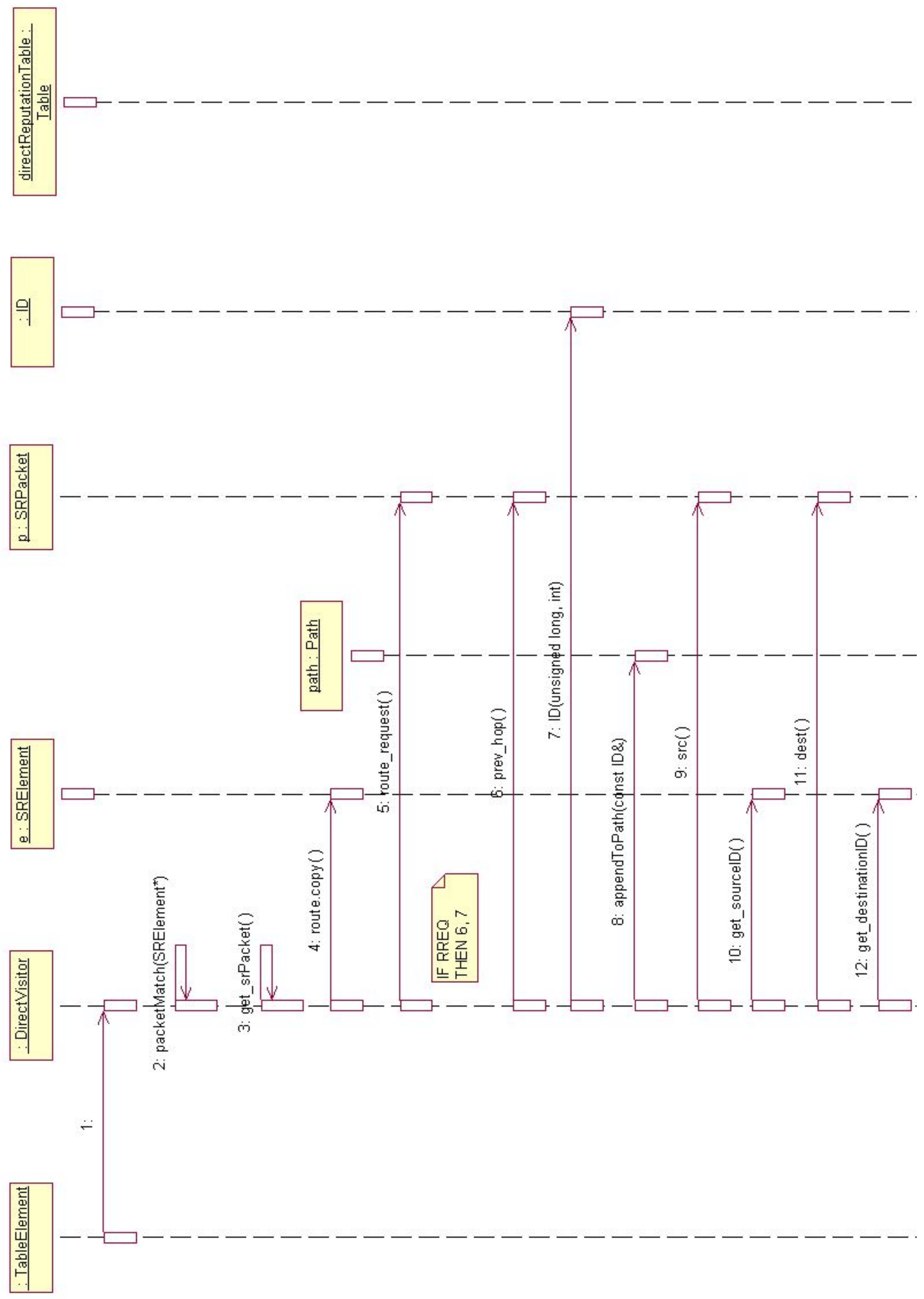


Figure A.31: Sequence Diagram for Matching a Duplicate with a Buffered Packet (*Top*).

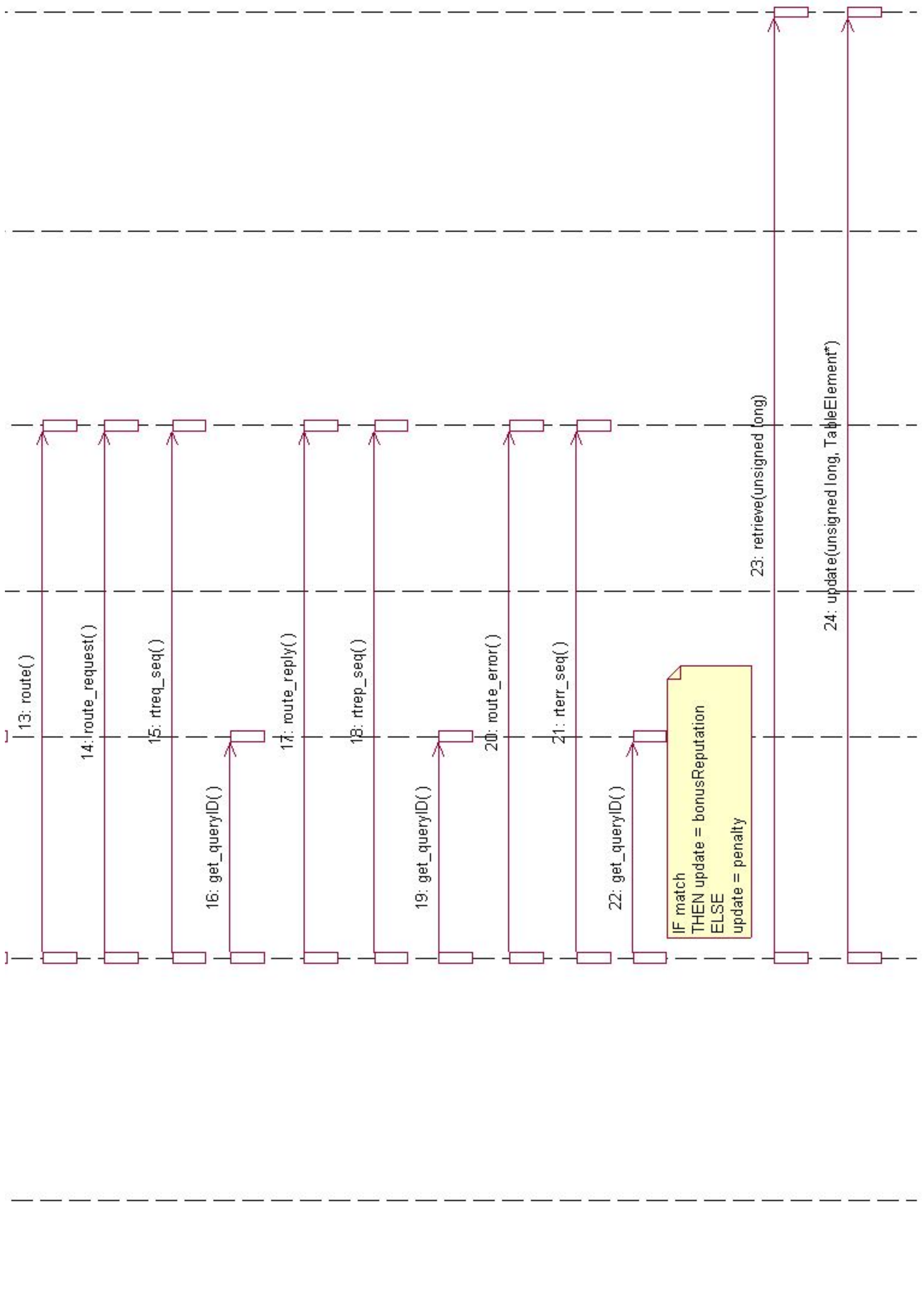


Figure A.32: Sequence Diagram for Matching a Duplicate with a Buffered Packet (*Bottom*).

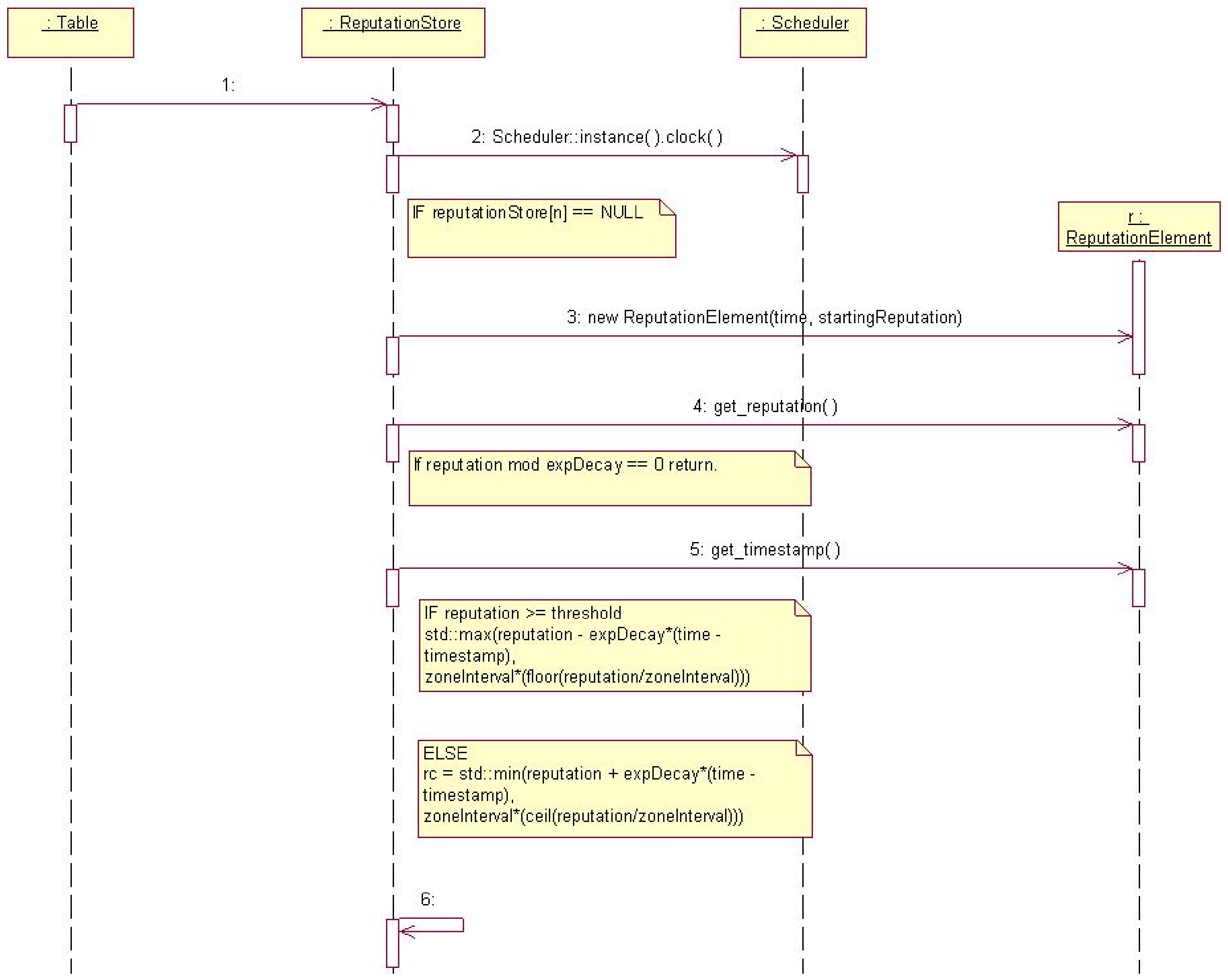

: Table
: ReputationStore
: Scheduler
r : ReputationElement
1:
2: Scheduler::instance( ).clock( )
IF reputationStore[n] == NULL
3: new ReputationElement(time, startingReputation)
4: get_reputation( )
If reputation mod expDecay == 0 return.
5: get_timestamp( )
IF reputation >= threshold
std::max(reputation - expDecay*(time -
timestamp),
zoneInterval*(floor(reputation/zoneInterval)))
ELSE
rc = std::min(reputation + expDecay*(time -
timestamp),
zoneInterval*(ceil(reputation/zoneInterval)))
6:

Figure A.33: Sequence Diagram for locating the Reputation of a Mobile Node.

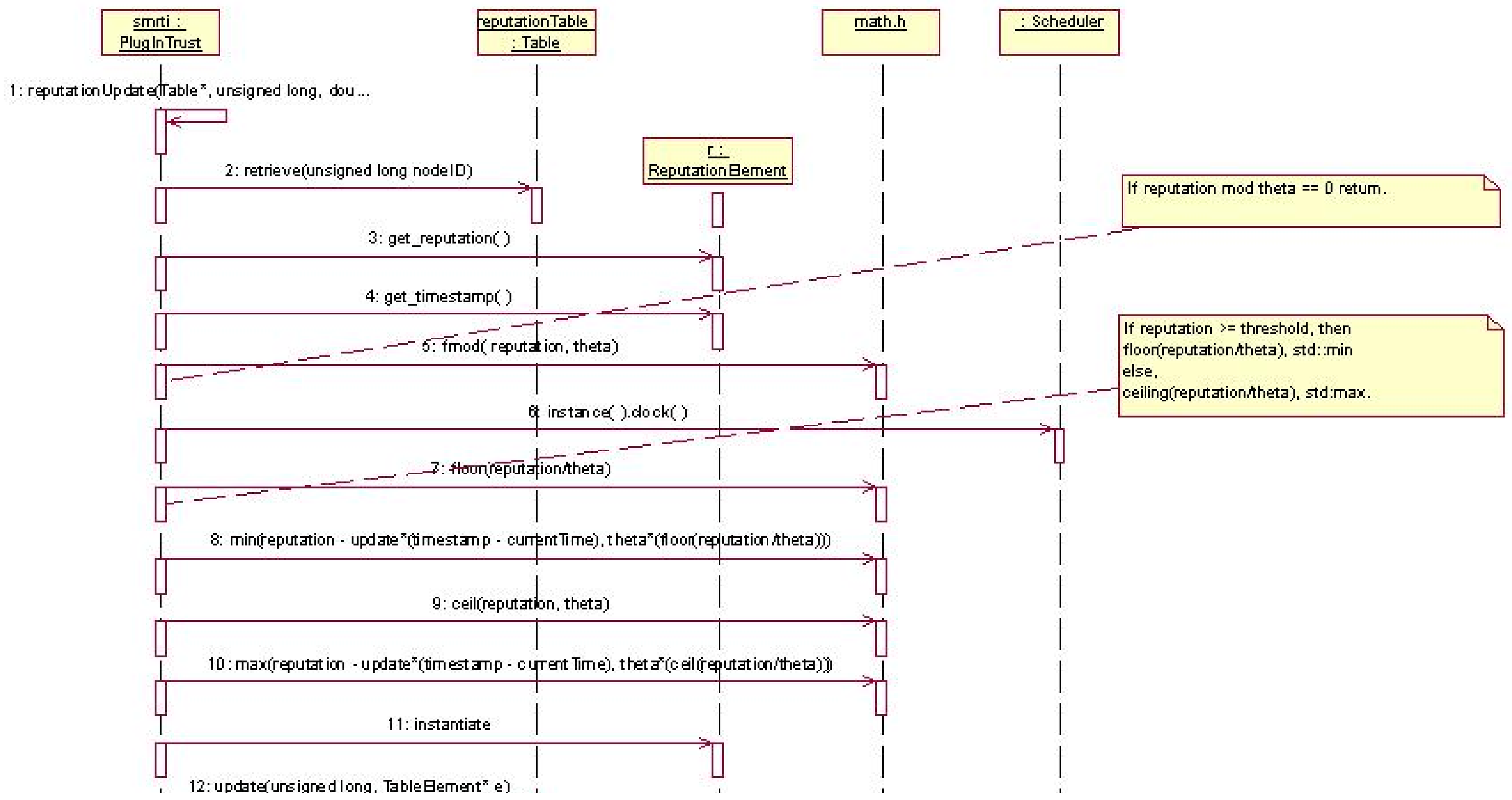

smrti : PlugInTrust
reputationTable : Table
r : ReputationElement
math.h
: Scheduler
1: reputationUpdate(Table*, unsigned long, dou...
2: retrieve(unsigned long nodeID)
3: get_reputation( )
4: get_timestamp( )
5: fmod(reputation, theta)
6: instance( ).clock( )
7: floor(reputation/theta)
8: min(reputation - update*(timestamp - currentTime), theta*(floor(reputation/theta)))
9: ceil(reputation, theta)
10: max(reputation - update*(timestamp - currentTime), theta*(ceil(reputation/theta)))
11: instantiate
12: update(unsigned long, TableElement* e)
If reputation mod theta == 0 return.
If reputation >= threshold, then floor(reputation/theta), std::min else, ceiling(reputation/theta), std:max.

Figure A.34: Sequence Diagram Changing the Reputation in the Absence of Evidence.

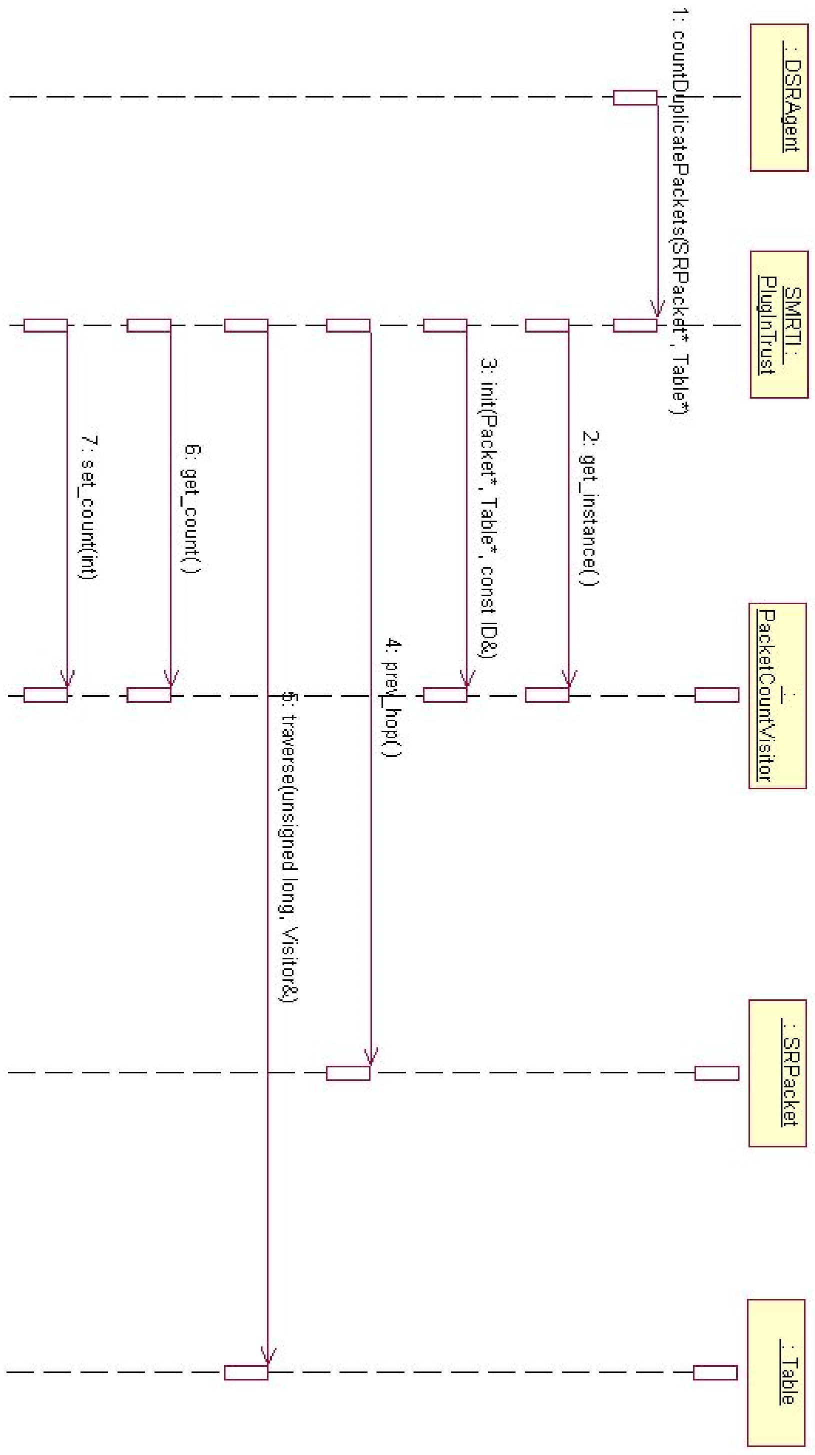


Figure A.35: Sequence Diagram for Counting Duplicate Packets in a Packet Buffer.

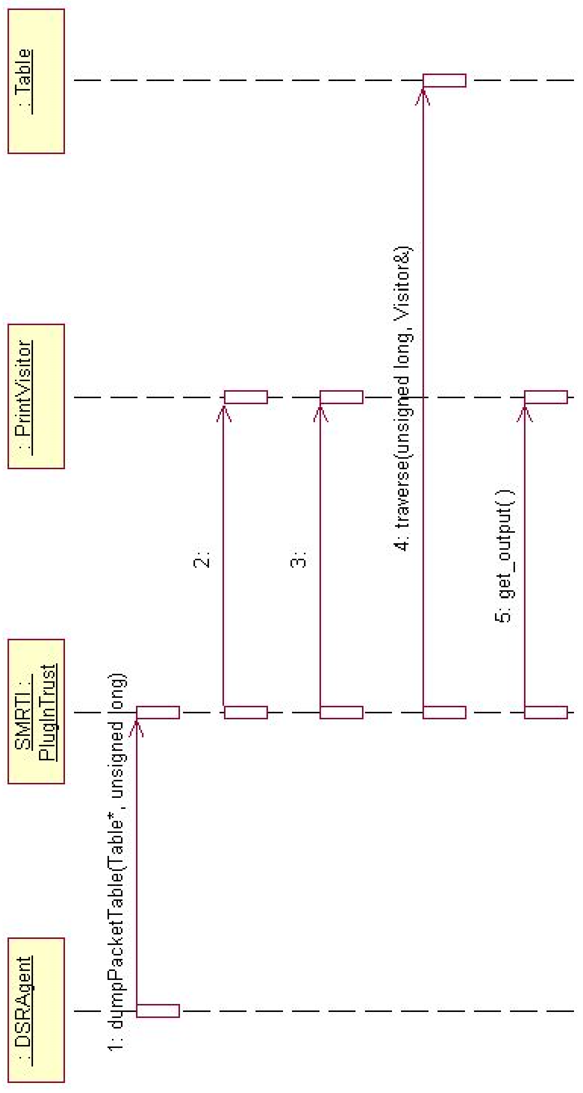

: DSRAgent
SMRTI : PlugInTrust
: PrintVisitor
: Table
1: dumpPacketTable(Table*, unsigned long)
2:
3:
4: traverse(unsigned long, Visitor&)
5: get_output( )

Figure A.36: Sequence Diagram for Dumping the Contents of a Packet Table.

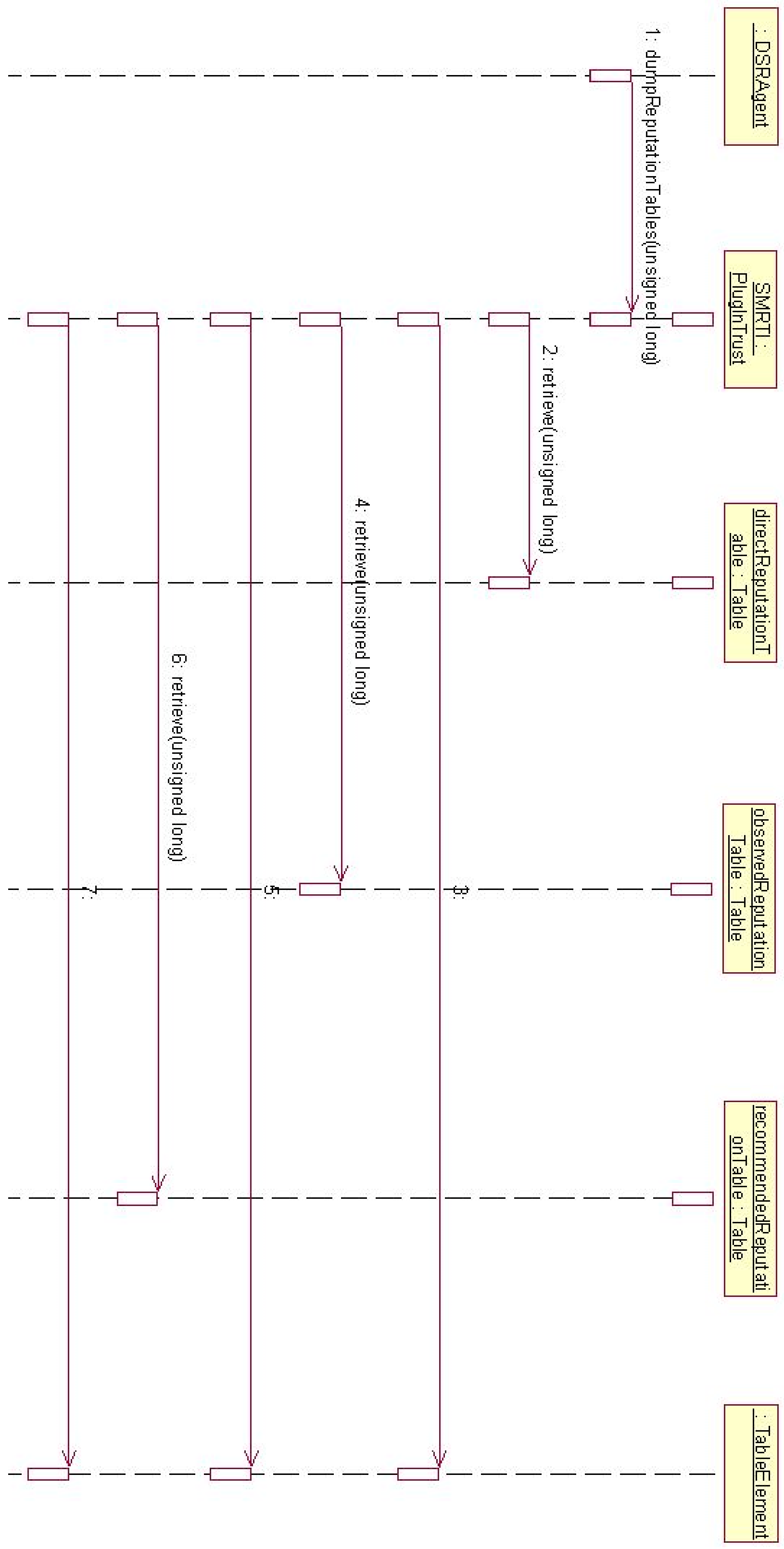

: DSRAgent
SMRTI : PluginTrust
directReputationTable : Table
observedReputationTable : Table
recommendedReputationTable : Table
: TableElement
1: dumpReputationTables(unsigned long)
2: retrieve(unsigned long)
3:
4: retrieve(unsigned long)
5:
6: retrieve(unsigned long)
7:

Figure A.37: Sequence Diagram for Dumping the Contents of a Reputation Table.

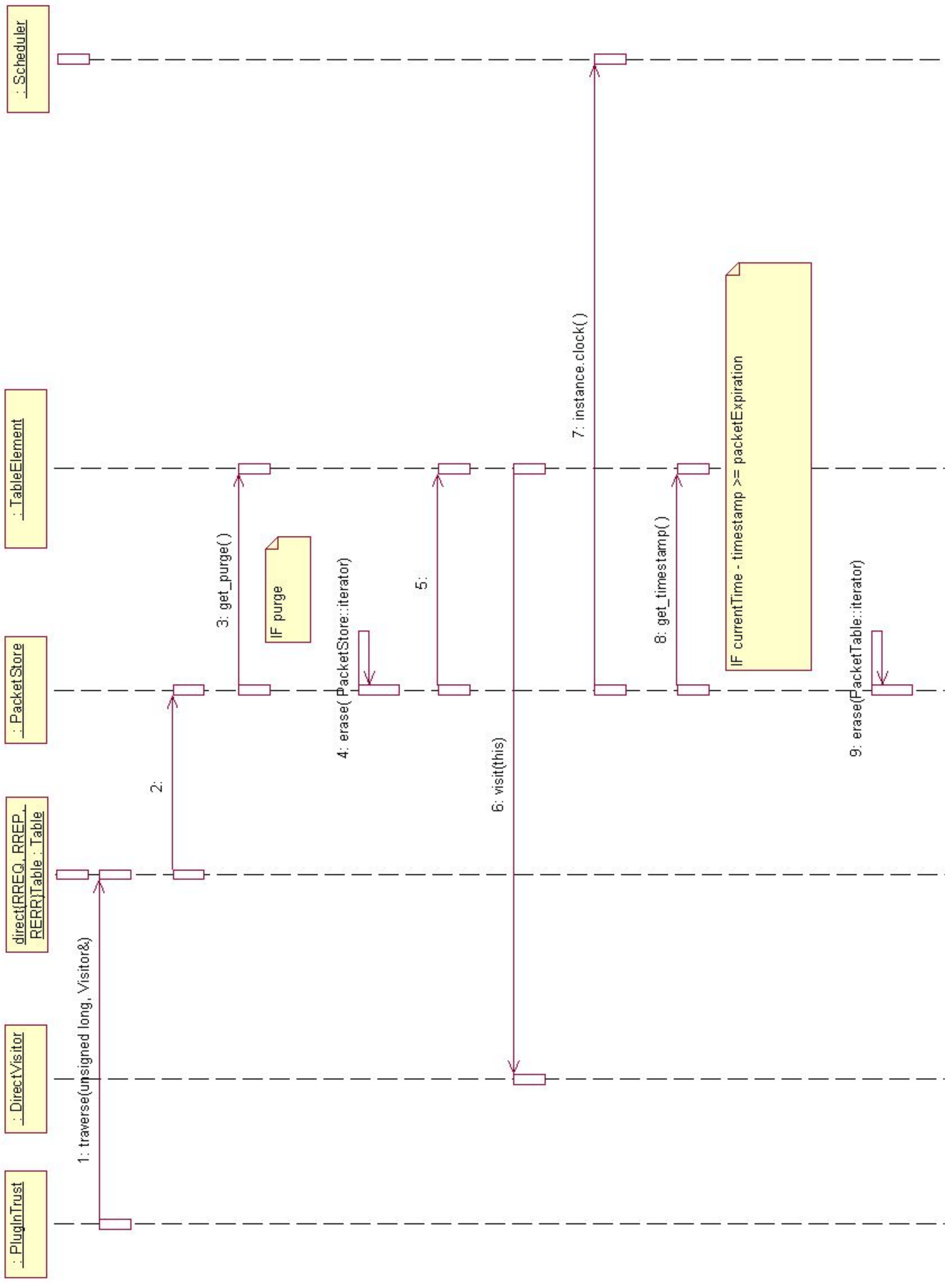

: PlugInTrust
: DirectVisitor
direct(RREQ, RREP, RERR)Table : Table
: PacketStore
: TableElement
: Scheduler
1: traverse(unsigned long, Visitor&)
2:
3: get_purge( )
IF purge
4: erase(PacketStore::iterator)
5:
6: visit(this)
7: instance.clock( )
8: get_timestamp( )
IF currentTime - timestamp >= packetExpiration
9: erase(PacketTable::iterator)

Figure A.38: Sequence Diagram for Purging an Expired Packet from a Buffer.

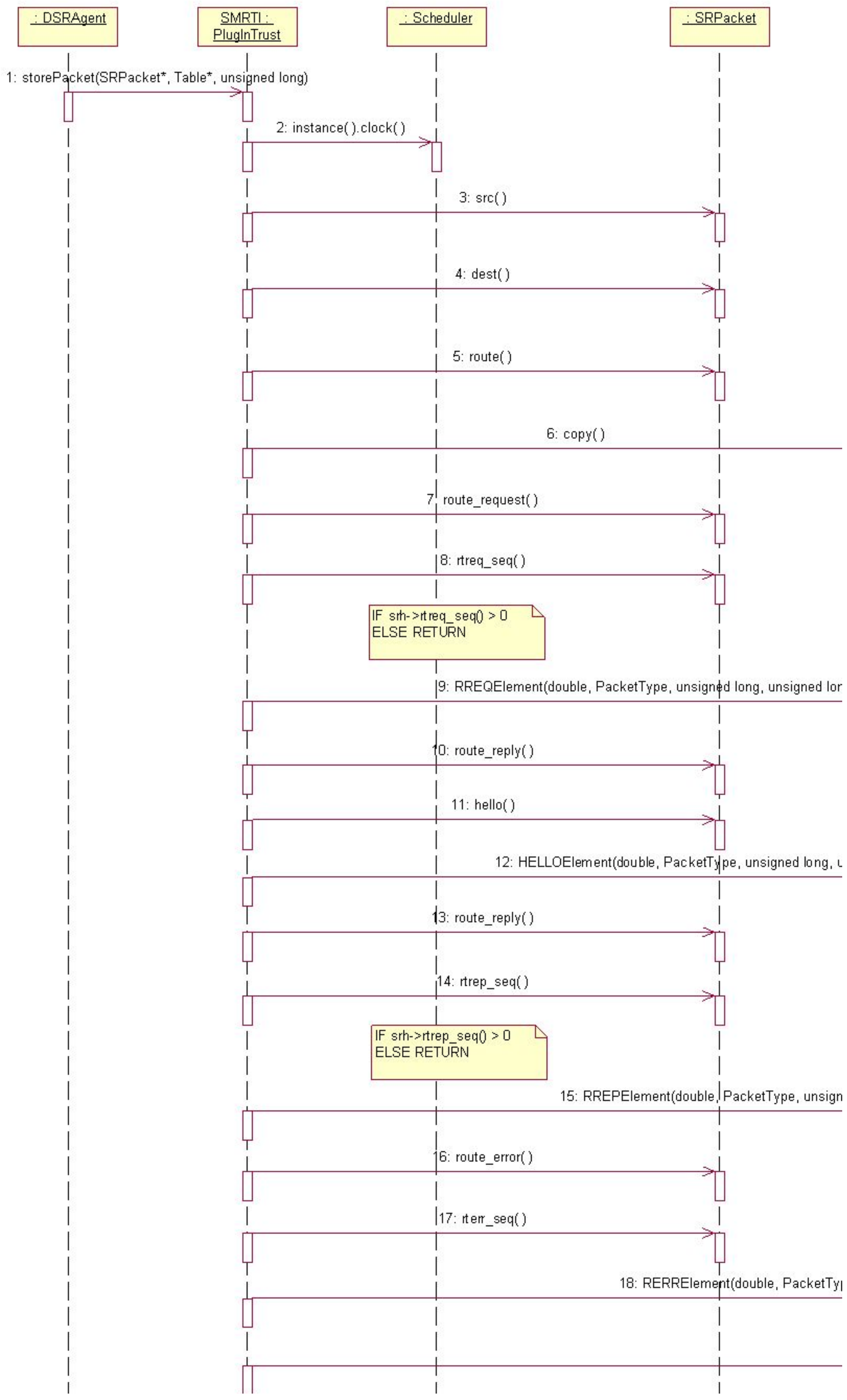
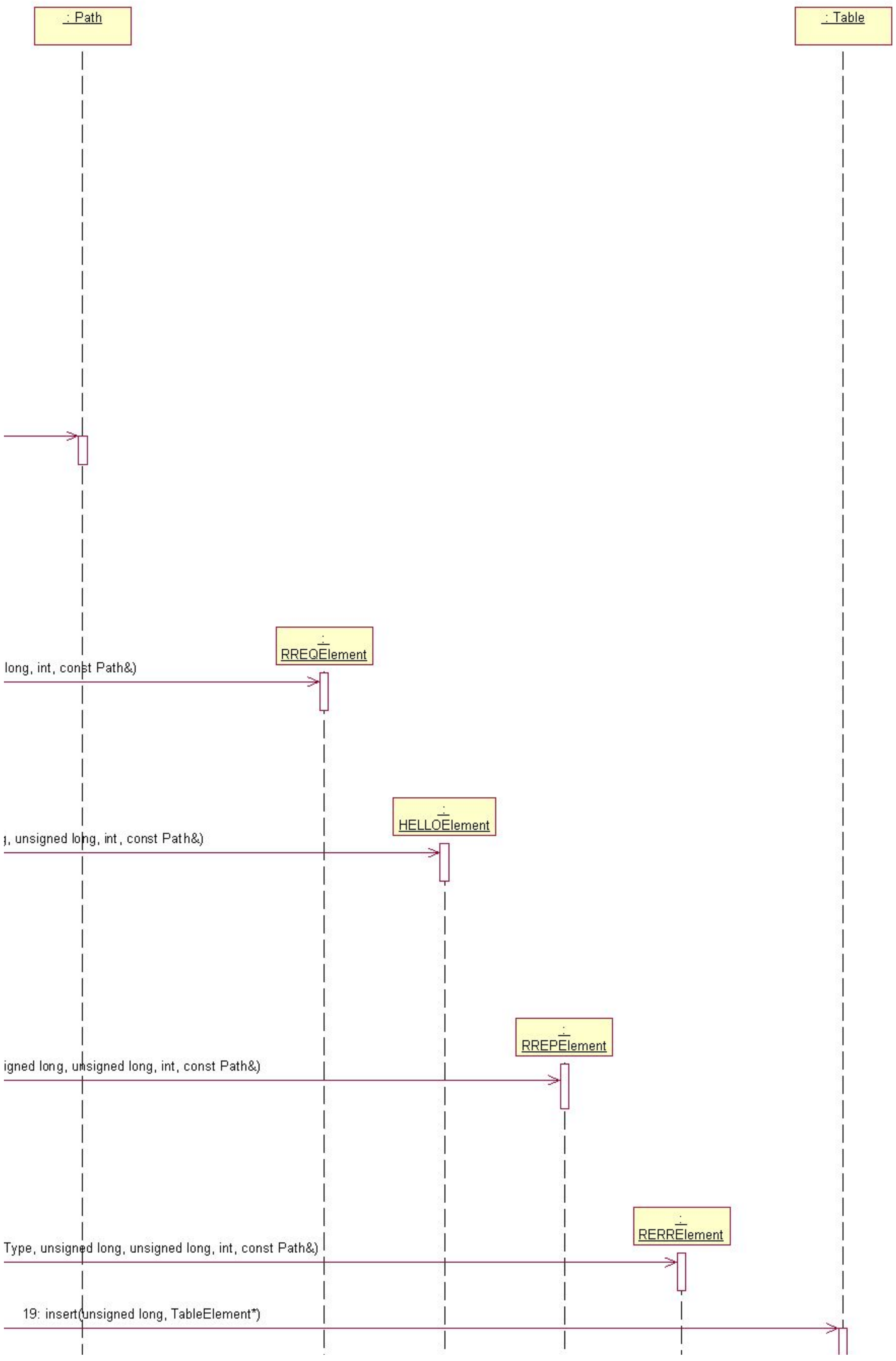


Figure A.39: Sequence Diagram for Storing a Packet in a Buffer (*Left*).



: Path
: Table
: RREQElement
long, int, const Path&)
: HELLOElement
, unsigned long, int, const Path&)
: RREPElement
igned long, unsigned long, int, const Path&)
: RERRElement
Type, unsigned long, unsigned long, int, const Path&)
19: insert(unsigned long, TableElement*)

Figure A.40: Sequence Diagram for Storing a Packet in a Buffer (*Bottom*).

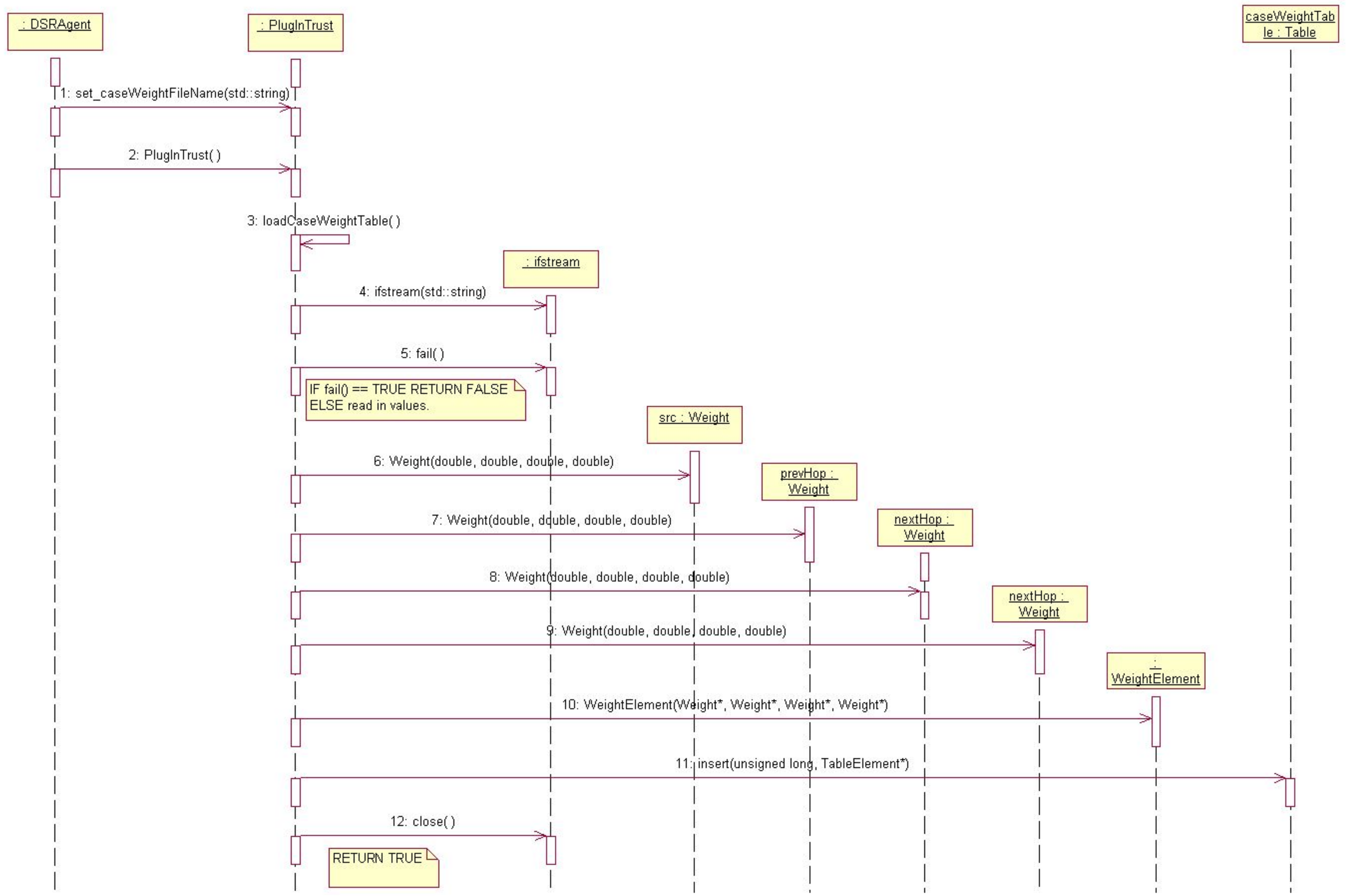

: DSRAgent
: PlugInTrust
caseWeightTable : Table
1: set_caseWeightFileName(std::string)
2: PlugInTrust( )
3: loadCaseWeightTable( )
: ifstream
4: ifstream(std::string)
5: fail( )
IF fail() == TRUE RETURN FALSE
ELSE read in values.
src : Weight
6: Weight(double, double, double, double)
prevHop : Weight
7: Weight(double, double, double, double)
nextHop : Weight
8: Weight(double, double, double, double)
nextHop : Weight
9: Weight(double, double, double, double)
: WeightElement
10: WeightElement(Weight*, Weight*, Weight*, Weight*)
11: insert(unsigned long, TableElement*)
12: close( )
RETURN TRUE

Figure A.41: Sequence Diagram for Assigning Weights to Various Participants during a Trust Evaluation.

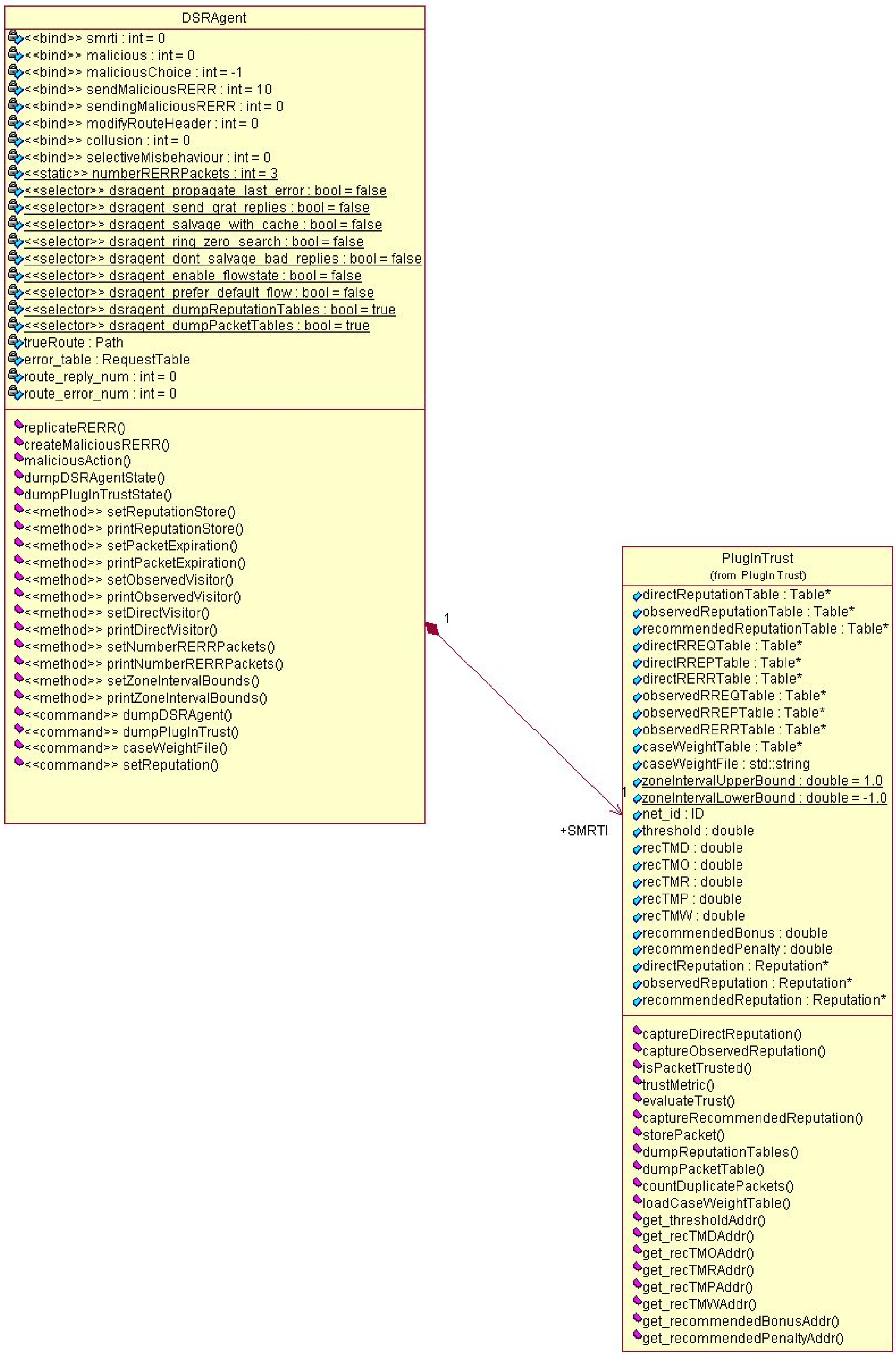

DSRAgent
<<bind>> smrti : int = 0
<<bind>> malicious : int = 0
<<bind>> maliciousChoice : int = -1
<<bind>> sendMaliciousRERR : int = 10
<<bind>> sendingMaliciousRERR : int = 0
<<bind>> modifyRouteHeader : int = 0
<<bind>> collusion : int = 0
<<bind>> selectiveMisbehaviour : int = 0
<<static>> numberRERRPackets : int = 3
<<selector>> dsragent_propagate_last_error : bool = false
<<selector>> dsragent_send_grat_replies : bool = false
<<selector>> dsragent_salvage_with_cache : bool = false
<<selector>> dsragent_ring_zero_search : bool = false
<<selector>> dsragent_dont_salvage_bad_replies : bool = false
<<selector>> dsragent_enable_flowstate : bool = false
<<selector>> dsragent_prefer_default_flow : bool = false
<<selector>> dsragent_dumpReputationTables : bool = true
<<selector>> dsragent_dumpPacketTables : bool = true
trueRoute : Path
error_table : RequestTable
route_reply_num : int = 0
route_error_num : int = 0
replicateRERR()
createMaliciousRERR()
maliciousAction()
dumpDSRAgentState()
dumpPlugInTrustState()
<<method>> setReputationStore()
<<method>> printReputationStore()
<<method>> setPacketExpiration()
<<method>> printPacketExpiration()
<<method>> setObservedVisitor()
<<method>> printObservedVisitor()
<<method>> setDirectVisitor()
<<method>> printDirectVisitor()
<<method>> setNumberRERRPackets()
<<method>> printNumberRERRPackets()
<<method>> setZoneIntervalBounds()
<<method>> printZoneIntervalBounds()
<<command>> dumpDSRAgent()
<<command>> dumpPlugInTrust()
<<command>> caseWeightFile()
<<command>> setReputation()
1
1
+SMRTI
PlugInTrust
(from PlugIn Trust)
directReputationTable : Table*
observedReputationTable : Table*
recommendedReputationTable : Table*
directRREQTable : Table*
directRREPTable : Table*
directRERRTable : Table*
observedRREQTable : Table*
observedRREPTable : Table*
observedRERRTable : Table*
caseWeightTable : Table*
caseWeightFile : std::string
zoneIntervalUpperBound : double = 1.0
zoneIntervalLowerBound : double = -1.0
net_id : ID
threshold : double
recTMD : double
recTMO : double
recTMR : double
recTMP : double
recTMW : double
recommendedBonus : double
recommendedPenalty : double
directReputation : Reputation*
observedReputation : Reputation*
recommendedReputation : Reputation*
captureDirectReputation()
captureObservedReputation()
isPacketTrusted()
trustMetric()
evaluateTrust()
captureRecommendedReputation()
storePacket()
dumpReputationTables()
dumpPacketTable()
countDuplicatePackets()
loadCaseWeightTable()
get_thresholdAddr()
get_recTMDAddr()
get_recTMOAddr()
get_recTMRAddr()
get_recTMPAddr()
get_recTMWAddr()
get_recommendedBonusAddr()
get_recommendedPenaltyAddr()

Figure A.42: Class Diagram showing Association between the DSR and SMRTI.

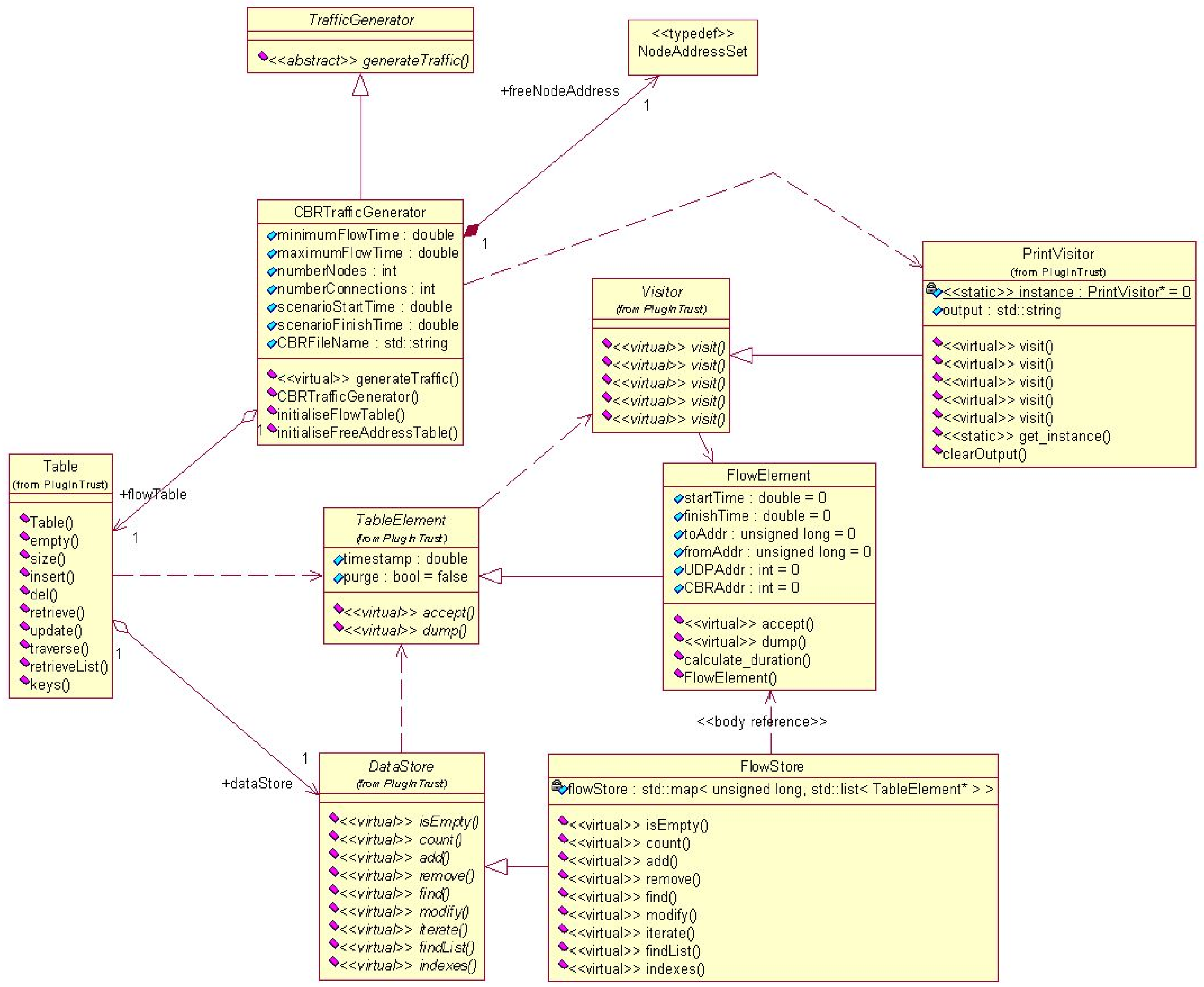


Figure A.43: Class Diagram showing the Design, Associations, Aggregations, and Relationships of CBR Generator.

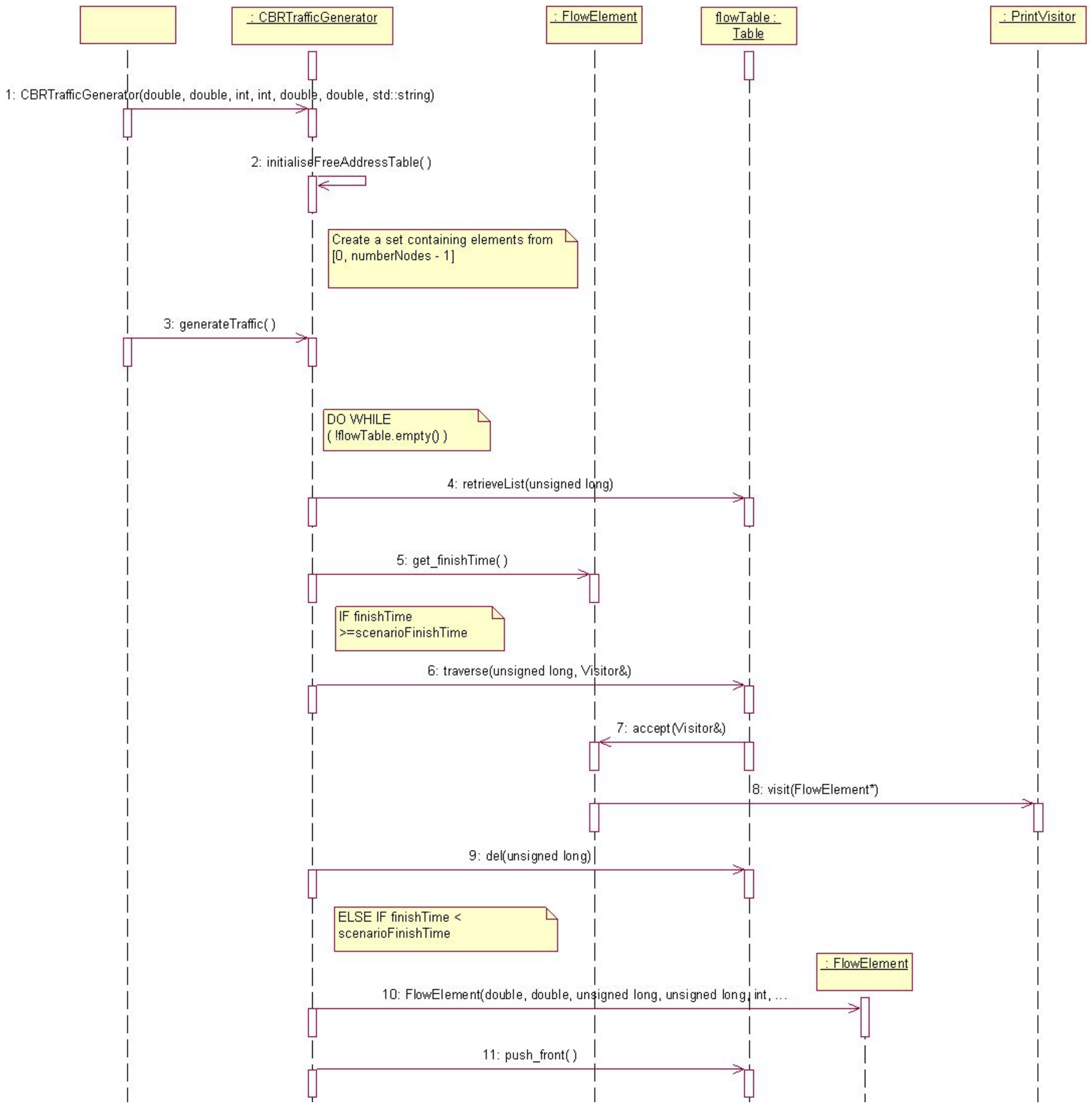

: CBRTrafficGenerator
: FlowElement
flowTable : Table
: PrintVisitor
1: CBRTrafficGenerator(double, double, int, int, double, double, std::string)
2: initialiseFreeAddressTable( )
Create a set containing elements from [0, numberNodes - 1]
3: generateTraffic( )
DO WHILE (!flowTable.empty())
4: retrieveList(unsigned long)
5: get_finishTime( )
IF finishTime >=scenarioFinishTime
6: traverse(unsigned long, Visitor&)
7: accept(Visitor&)
8: visit(FlowElement*)
9: del(unsigned long)
ELSE IF finishTime < scenarioFinishTime
: FlowElement
10: FlowElement(double, double, unsigned long, unsigned long, int, ...
11: push_front( )

Figure A.44: Sequence Diagram for the CBR Traffic Generation.

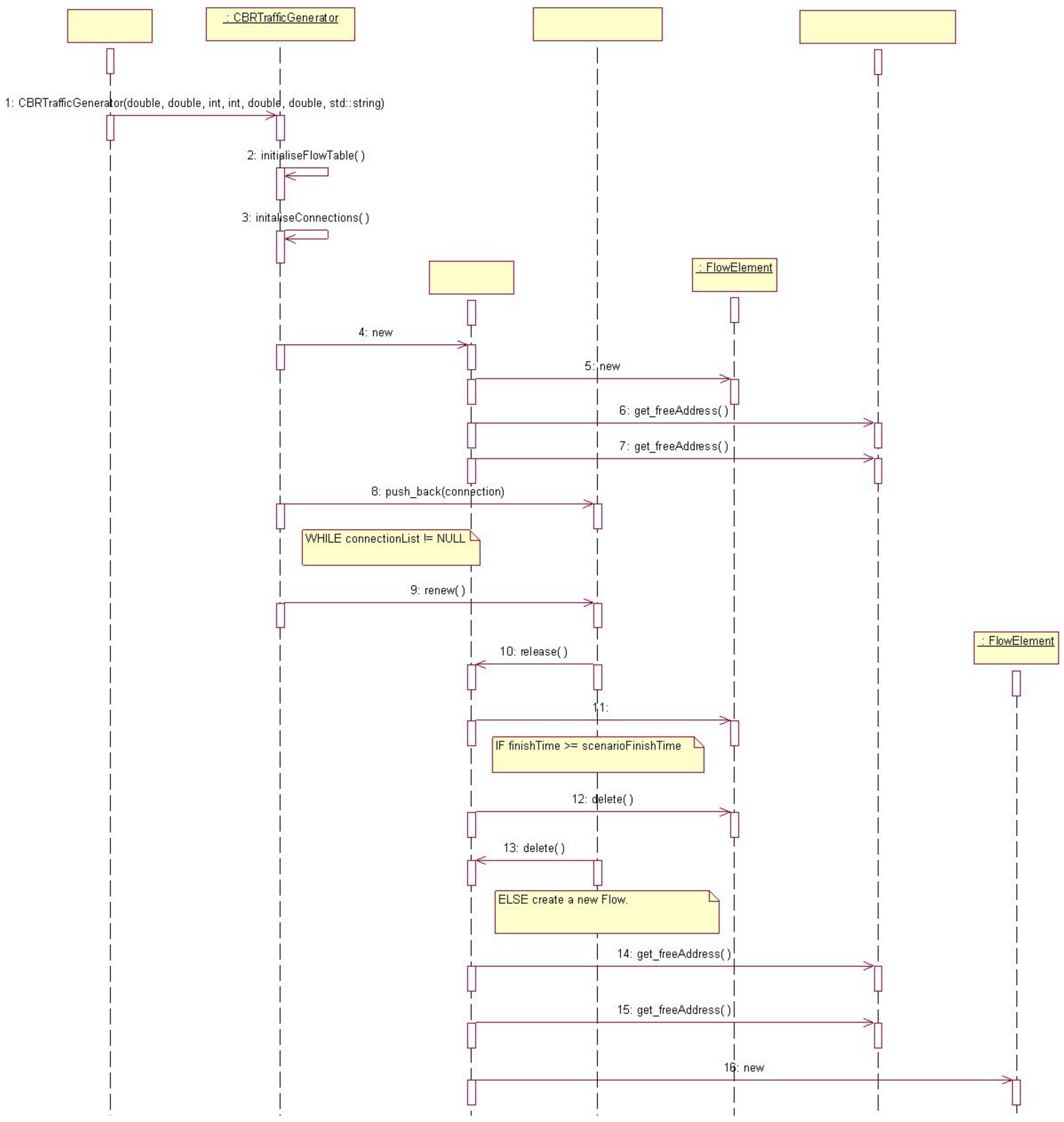


Figure A.45: Sequence Diagram for the Inner Working of CBR Traffic Generator.

B

Code Snippets

Given that Appendix A has already presented the UML design diagrams that details the data fields and methods of the classes, we present the snippets of code that provides an insight into the operation of the fellowship and SMRTI models. In the following Section B.1, we present the code snippets that are unique to the fellowship model. Section B.2 provides the code snippets that are unique to the SMRTI model and also common to both the models.

B.1 Fellowship Implementation

In the following, we list few classes with snippet of codes that play a significant role in the implementation of the fellowship model.

- **Mac-80-11.cc (Listing B.1)** presents the headers included for the successful operation of the fellowship model. This class also lists the methods that are called from the OTcl level. These methods serve as a platform for initialising or varying the simulation parameters to providing the details of the fellowship's operation for the purpose of debugging. Finally, it reveals the flow of controls of when the fellowship is activated from the perspective of a mobile node.

- **Wireless-phy.cc (Listing B.2)** is used to measure the time taken to transmit and receive a packet so that the energy contributed for transmitting a packet on behalf of the previous-hop or receiving a packet from the previous-hop can be measured respectively. As noted in Chapter 3, contributing the battery energy based on the packet-length enables a mobile node to defend against intelligent selfish nodes.

- **Monitor.cc (Listing B.3)** is the main class for the fellowship model from where the controls are initiated for various scenarios. Furthermore, the contribution for other nodes depending on their benign or misbehaviours are captured and computed at this class and used as a reference for making future decisions.

- **RateLimitation.cc (Listing B.4)** is invoked whenever a packet is received from the previous-hop. This class transfers the control to the enforcement component to determine the credibility of the previous-hop. It updates the contribution for the previous-hop based on the behaviour exhibited by the previous-hop.

- **Enforcement.cc (Listing B.5)** is called by the rate-limitation to determine whether or not a packet can be accepted for forwarding. In addition, it also keeps track of the next-hop's forwarding behaviour.

- **Restoration.cc (Listing B.6)** completes the sequence of actions initiated from the rate-limitation. Given the transmission channel is free from contention, it transmits the packet on the behalf of the previous-hop. Alternatively, it buffers the packet for a future transmission at the time of contention or adds the contribution for the previous-hop if the transmission buffer experiences congestion.

- **EnforcementVisitor.cc (Listing B.7)** takes a packet and visits the packet buffer to discover a matching combination.
- **CommitmentElement.cc (Listing B.8)** is used by a mobile node to hold the total count of packets that was sent to a neighbour and the total count of packets that has been forwarded by the neighbour. These values are used to determine whether or not the neighbour can be trusted for forwarding packets.
- **CommitmentStore.cc (Listing B.9)** is used to store data structures of the type *CommitmentElement*.
- **MacPacketStore.cc (Listing B.10)** is used to store data structures of the type *MACPacketElement*. It contains methods that iterate the table to discover packets for which a duplicate has been received or not and also the packets that have expired. It then computes the parameters to determine the contribution for the next-hop.

Listing B.1: Mac-802-11.cc

```
// Injection at the MAC-802.11
#include "mac/MACPacketStore.h"
#include "mac/CommitmentStore.h"
#include "dsr/hdr_sr.h"
#include "mac/MACPacketElement.h"
#include "dsr/dsragent.h"
#include "mac/EnforcementVisitor.h"

// For the purpose of debugging
void Mac802_11::dumpMac802_11State() {
        Tcl& tcl = Tcl::instance();
        tcl.evalf("puts \"\tusingFellowship = %d\"", usingFellowship);
        tcl.evalf("puts \"\tmaliciousPacketDrop = %d\"", maliciousPacketDrop);
}

// For the purpose of debugging
void Mac802_11::dumpFellowshipState() {
        Tcl& tcl = Tcl::instance();
        tcl.evalf("puts \"\tnetif_ address = %d\"", fellowship->get_netif());
        tcl.evalf("puts \"\tll_ address = %d\"", fellowship->get_ll());
        tcl.evalf("puts \"\tcommitmentBonus = %f\"",
                              fellowship->get_commitmentBonus());
        tcl.evalf("puts \"\tcommitmentPenalty = %f\"",
                              fellowship->get_commitmentPenalty());
        tcl.evalf("puts \"\tpacketDeliveryThreshold = %f\"",
                              fellowship->get_packetDeliveryThreshold());
        tcl.evalf("puts \"\tcheckFlooding = %f\"", fellowship->get_checkFlooding());
}

// Activating fellowship as the next sequence after MAC-80.11
void Mac802_11::configureFellowship() {
        fellowship->set_netif(netif_);  // for WirelessPhy
        LL* ll = (LL*)((Connector*)uptarget_)->get_target();
        fellowship->set_ll(ll); // for LL
        unsigned long net_id = Mac::addr();
        fellowship->set_net_id(net_id);
        installTap(fellowship); // for Promiscous tap.
        DSRAgent* dsragent = (DSRAgent*)((Connector*)ll->uptarget())->get_target();
}

// Registering the above actions at the command interface
int Mac802_11::command(int argc, const char*const* argv) {
        if(argc == 2) {
                .
                .
                .
                if (strcmp(argv[1], "dumpFellowship") == 0) {
                        printf("\tMac802_11::dumpFellowshipState\n");
                        dumpFellowshipState();
                        return TCL_OK;
                }
                if (strcmp(argv[1], "configureFellowship") == 0) {
                        printf("\tMac802_11::configureFellowship\n");
                        configureFellowship();
                        return TCL_OK;
                }
        }
        .
        .
        .
}

// TCL Hooks for the simulator
static class Mac802_11Class : public TclClass {
public:
        Mac802_11Class() : TclClass("Mac/802_11") {}
        TclObject* create(int, const char*const*) {
                return (new Mac802_11());
        }
        // To vary static parameters
        virtual void bind();
        virtual int method(int argc, const char*const* argv);
} class_mac802_11;

// Hooks for accessing from the simulation scripts
void Mac802_11Class::bind() {
        TclClass::bind();
        add_method("setMACPacketStore");
        add_method("printMACPacketStore");
        add_method("setCommitmentStore");
        add_method("printCommitmentStore");
        add_method("setRate");
        add_method("printRate");
}

// Methods for the above bounds
int Mac802_11Class::method(int ac, const char*const* av) {
    Tcl& tcl = Tcl::instance();
    int argc = ac - 2;
    const char*const* argv = av + 2;
        if (argc == 2) {
                if (strcmp(argv[1], "printMACPacketStore") == 0) {
                        tcl.evalf("puts \"\tMACPacketStore::packetExpiration = %f\"",
                                        MACPacketStore::packetExpiration);
                        return (TCL_OK);
                }
                if (strcmp(argv[1], "printCommitmentStore") == 0) {
                        tcl.evalf("puts \"\tCommitmentStore::initialCommitment =
                                        %f\"", CommitmentStore::initialCommitment);
                        return (TCL_OK);
                }
                if (strcmp(argv[1], "printRate") == 0) {
                        tcl.evalf("puts \"\tMonitor::Rate = %d\"", Monitor::Rate);
                        return (TCL_OK);
                }
        } else if (argc == 3) {
                if (strcmp(argv[1], "setMACPacketStore") == 0) {
                        MACPacketStore::packetExpiration = atof(argv[2]);
                        return (TCL_OK);
                } else if (strcmp(argv[1], "setCommitmentStore") == 0) {
                        CommitmentStore::initialCommitment = atof(argv[2]);
                        return (TCL_OK);
                } else if (strcmp(argv[1], "setRate") == 0) {
                        Monitor::Rate = atoi(argv[2]);
                        return (TCL_OK);
                }
        } return TclClass::method(ac, av);
}

// Initialisations for fellowship while MAC-802.11 is initialised
Mac802_11::Mac802_11() {
        .
        .
        .
        fellowship = new Monitor(netif_, ll_);
        bind("usingFellowship", &usingFellowship);
        bind("commitmentBonus", fellowship->get_commitmentBonusAddr());
        bind("commitmentPenalty", fellowship->get_commitmentPenaltyAddr());
        bind("packetDeliveryThreshold",
                 fellowship->get_packetDeliveryThresholdAddr());
        bind("checkFlooding", fellowship->get_checkFloodingAddr());
        bind("floodingPenalty", fellowship->get_floodingPenaltyAddr());
        bind("maliciousPacketDrop", &maliciousPacketDrop);
}

/**************************************************
 Sequence initiated after the packet been processed
 by the MAC-802.11 layer. Core functionality of
 fellowship is called from here.
 **************************************************/
void Mac802_11::recv(Packet *p, Handler *h)
{
        .
        .
        .
        // Handle outgoing packets.
        if(hdr->direction() == hdr_cmn::DOWN) {
                EnforcementVisitor* v = EnforcementVisitor::get_instance();
                int bufferIndex = 0;
                fellowship->get_packetBuffer()->traverse(bufferIndex, *v);
                // By default all MAC layer packets should be forwarded.
                bool forward = true;
                // The MAC identifier of this node.
                int net_id = Mac::addr();
                hdr_cmn* cmnh = hdr_cmn::access(p);
                unsigned long next_hop = cmnh->next_hop();
```

```
hdr_mac* mach = hdr_mac::access(p);
mach->next_hop() = next_hop;
unsigned long prev_hop = mach->prev_hop();
hdr_sr* srh = hdr_sr::access(p);
unsigned long src = srh->true_addrs()[0].addr;
if(srh->route_reply() && usingFellowship && src != net_id) {
forward = fellowship->checkCommitment(p, 0.0, prev_hop);
}
else if (cmnh->ptype() == PT_CBR && usingFellowship) {
        /*************************************************
         A node originating a packet is not required to
         check whether it should forward a DATA packet on
         its own behalf. Next, get the src of the packet.
         *************************************************/
        if (src == net_id) {
                forward =
                        fellowship->checkCommitment(p, 0.0, next_hop);
                if (!forward) {
                        double time = Scheduler::instance().clock();
                        ID from_id = ID(net_id, ::IP);
                        ID to_id = ID(next_hop, ::IP);
                        LL* ll = (LL*)((Connector*)uptarget_)->
                                get_target();
                        DSRAgent* dsragent = (DSRAgent*)
                                ((Connector*)ll->uptarget())->
                                        get_target();
                        dsragent->get_malicious_route_cache()->
                                noticeDeadLink(from_id, to_id, time);
                        dsragent->get_route_cache()->
                                noticeDeadLink(from_id, to_id, time);
                } else {
                        // Store the MACPacketElement.
                        double time = Scheduler::instance().clock();
                        int uid = cmnh->uid();
                        MACPacketElement* m = new MACPacketElement
                                (time, uid, next_hop);

                        /********************************
                         Store the uid in the packetBuffer
                         indexed by the next hop's id.
                         ********************************/
                        fellowship->get_packetBuffer()->insert(0, m);
                        fellowship->updateCommitment
                                (next_hop, 0.0, SENT);
                }
```

```
                    } else {
                            /************************************************
                             The RateLimitation module injection point. Checks
                             if there is enough commitment to forward the
                             outgoing packet on behalf of the requesting node.
                             ************************************************/
                            forward = fellowship->recv(p);
                    }
            }

            // Malicious packet drop code.
            if((maliciousPacketDrop && cmnh->ptype() == PT_CBR) ||
            (!forward && !fellowship->get_sendRERR())) {
                    Packet::free(p);
                    p = 0;
                    h->handle((Event*) 0);
            } else {
                    /************************************
                     The packet is being sent out. So this
                     node becomes the previous-hop now.
                     ************************************/
                    mach->prev_hop() = net_id;
                    send(p, h );
            } return;
    }
    .
    .
    .
}
```

Listing B.2: Wireless-phy.cc

```
double WirelessPhy::get_TxEnergy(double txtime) {
        double time = Scheduler::instance().clock();
        double start_time = MAX(channel_idle_time_, time);
        double end_time = MAX(channel_idle_time_, time + txtime);
        double actual_txtime = end_time - start_time;
        double TxEnergy = em()->calculate_TxEnergy(actual_txtime, Pt_consume_);
        return TxEnergy;
}

double WirelessPhy::get_RxEnergy(double rcvtime) {
        double time = Scheduler::instance().clock();
        double start_time = MAX(channel_idle_time_, time);
        double end_time = MAX(channel_idle_time_, time + rcvtime);
        double actual_rcvtime = end_time - start_time;
        double RxEnergy = em()->calculate_RxEnergy(actual_rcvtime, Pr_consume_);
        return RxEnergy;
}
```

Listing B.3: Monitor.cc

```
bool Monitor::recv (Packet* p) {
        double currentTime = Scheduler::instance().clock();
        EnforcementVisitor* v = EnforcementVisitor::get_instance();
        int bufferIndex = 0;
        packetBuffer->traverse(bufferIndex, *v);
        bool forward = false;
        forward = rateLimitation->recv(p);
        return forward;
}

bool Monitor::createFellowship () {
        rateLimitation = new RateLimitation(this);
        enforcement = new Enforcement(this);
        restoration = new Restoration(this);
        bool createFellowshipOK = false;
        if (rateLimitation != 0 && enforcement != 0 && restoration != 0) {
                createFellowshipOK = true;
        } return createFellowshipOK;
}

double Monitor::calculateRemaining (double commitment) {
        double energy = netif->node()->energy_model()->energy();
        double remaining = energy * commitment;
        return remaining;
}

double Monitor::calculateRecent (double remaining, double contribution) {
        double recent = remaining + contribution;
        return recent;
}

double Monitor::calculateCommitment (double recent) {
        double energy = netif->node()->energy_model()->energy();
        if (fabs(energy) < EPS) {
                energy = EPS;
        }
        double commitment = recent/energy;
        return commitment;
}
```

```
double Monitor::updateCommitment (unsigned long nodeID, double contribution,
                                PacketsDelivered packetsDelivered) {
        CommitmentElement* e = (CommitmentElement*)commitmentTable->retrieve(nodeID);
        double percentageEnergy = e->get_percentageEnergy();
        double remaining = calculateRemaining(percentageEnergy);
        double recent = calculateRecent(remaining, contribution);
        double commitment = calculateCommitment(recent);
        double time = Scheduler::instance().clock();
        int packetsSent = e->get_packetsSent();
        int packetsForwarded = e->get_packetsForwarded();
        if (packetsDelivered == SENT) {
                ++packetsSent;
        } else if (packetsDelivered == FORWARDED) {
                ++packetsForwarded;
        }
        std::map< unsigned long, RateOfFlow* > oldRateOfFlow = e->get_rateOfFlow();
        commitmentTable->update(nodeID, new
                CommitmentElement(time, commitment, packetsSent, packetsForwarded));
        CommitmentElement* newE =
                (CommitmentElement*)commitmentTable->retrieve(nodeID);
        newE->set_rateOfFlow(oldRateOfFlow);
        return commitment;
}

bool Monitor::checkCommitment (Packet* p, double contribution, unsigned long nodeID) {
        bool enoughCommitment = true;
        CommitmentElement* c = (CommitmentElement*)commitmentTable->retrieve(nodeID);
        double commitment = c->get_percentageEnergy();
        double remaining = calculateRemaining(commitment);
        // Contribution is a -ve quantity
        if ((remaining + contribution) < 0.0) {
                /*********************************************
                 Do not transmit. Return false to indicate not
                 to forward the Packet p to the monitor.
                 *********************************************/
                enoughCommitment = false;
        } else {
                updateCommitment(nodeID, contribution, NOTHING);
        } return enoughCommitment;
}

bool Monitor::is_ifqFull (Packet* p) {
        DSRAgent* dsrAgent = (DSRAgent*)((Connector*)ll->uptarget())->get_target();
        CMUPriQueue* ifq = dsrAgent->get_ifq();
        bool is_ifqFull = ifq->prq_isfull(p);
        return is_ifqFull;
}

double Monitor::calculate_packetDeliveryRatio (unsigned long nodeID) {
        CommitmentElement* e = (CommitmentElement*)commitmentTable->retrieve(nodeID);
        double packetsSent = e->get_packetsSent();
        double packetsForwarded = e->get_packetsForwarded();
        double packetDeliveryRatio = 0.0;
        if (packetsSent > 0) {
                packetDeliveryRatio = packetsForwarded/packetsSent;
        } else if (packetsSent == 0) {
                packetDeliveryRatio = 1.0;
        } return packetDeliveryRatio;
}

void Monitor::tap (const Packet* p) {
        /****************************************************
         The enforcement module's injection point. The Packet p
         is being received. So check if this node's fellowship
         module has a copy of the received Packet p in its
         packet buffer. If it does then increase the commitment
         of the previous-hop by contributionTx.
         ****************************************************/
        struct hdr_cmn* cmnh = hdr_cmn::access(p);
        // Listen to only cbr packets at this point.
        if (cmnh->ptype() == PT_CBR) {
                enforcement->captureCommitment(p);
        }
}
```

Listing B.4: RateLimitation.cc

```
bool RateLimitation::recv (Packet* p) {
        /********************************************************
         Check if the maliciousFlooding is turned on at Mac-80.11.
         If its turned OFF then skip the following line and move
         on to the Enforcement for the evaluation purpose.
         ********************************************************/
        bool forward = true;
        Monitor* fellowshipDirector = get_director();
        Enforcement* enforcement = fellowshipDirector->get_enforcement();
        bool checkFlooding = fellowshipDirector->get_checkFlooding();
        if (checkFlooding) {
                hdr_mac* mach = hdr_mac::access(p);
                unsigned long prev_hop = mach->prev_hop();
                CommitmentElement* e = (CommitmentElement*)(fellowshipDirector->
                    get_commitmentTable()->retrieve(prev_hop));
            hdr_sr* srh = hdr_sr::access(p);
            unsigned long src = srh->true_addrs()[0].addr;
            e->checkFlow(src);
            double intervalStartTime = e->get_intervalStartTime(src);
            int receivedIntervalCount = e->get_receivedIntervalCount(src);
            int rate = Monitor::Rate;
            receivedIntervalCount += 1;
            e->set_receivedIntervalCount(src, receivedIntervalCount);
            double time = Scheduler::instance().clock();
            if (receivedIntervalCount == 1) {
                    e->set_intervalStartTime(src, time);
            }
            if (receivedIntervalCount >= rate) {
                    double timeElapsed = time - intervalStartTime;
                    if (timeElapsed < 1.0) {
                            if(receivedIntervalCount > rate) {
                                    e->set_exceededRate(src, true);
                                    e->set_receivedIntervalCount(src, 0);
                            }
                    } else {
                            e->set_exceededRate(src, false);
                    }
            }
            bool exceededRate = e->get_exceededRate(src);
            if (exceededRate) {
                    hdr_cmn* cmnh = hdr_cmn::access(p);
                    Monitor::TxTime = cmnh->txtime();
                    double contributionTx = ((WirelessPhy*)fellowshipDirector->
                            get_netif())->get_TxEnergy(Monitor::TxTime);
                    double floodingPenalty =
                            fellowshipDirector->get_floodingPenalty();
                    double contribution =
                            -1.0*contributionTx*(floodingPenalty + 1.0);
                    fellowshipDirector->updateCommitment
                            (prev_hop, contribution, NOTHING);
            }
    }
    forward = enforcement->recv(p);
    return forward;
}
```

Listing B.5: Enforcement.cc

```
bool Enforcement::recv (Packet* p) {
    Monitor* fellowshipDirector = get_director();
```

```
        Restoration* restoration = fellowshipDirector->get_restoration();
        bool forward = false;
        bool enoughCommitmentPrevHop = true;
        bool checkFlooding = fellowshipDirector->get_checkFlooding();
        if (checkFlooding) {
                hdr_mac* mach = hdr_mac::access(p);
                unsigned long prev_hop = mach->prev_hop();
                enoughCommitmentPrevHop =
                        fellowshipDirector->checkCommitment(p, 0.0, prev_hop);
        }
        /*********************************************************
         If 'MaliciousFlooding' is turned OFF at MAC-802.11 then we
         can skip the following line which decreases the commitment
         for the previous-hop. This is because only packet dropping
         is concerned.
         *********************************************************/
        if (enoughCommitmentPrevHop) {
                /*********************************************************
                 Pass the packet 'p' to the restoration module to see if
                 the packet can be forwarded for the requesting node.
                 *********************************************************/
                forward = restoration->recv(p);
        } return forward;
}

void Enforcement::captureCommitment (const Packet* p) {
        PacketCountVisitor* v = PacketCountVisitor::get_instance();
        Monitor* fellowshipDirector = get_director();
        Table* packetBuffer = fellowshipDirector->get_packetBuffer();
        v->set_packet(const_cast<Packet*>(p));
        packetBuffer->traverse(0, *v);
        /*************************************************************
         A match means that the packet has been forwarded on the behalf
         of this node by the previous-hop. Therefore, update the energy
         and forward count.
         *************************************************************/
        if (v->get_count()) {
                hdr_mac* mach = hdr_mac::access(p);
                unsigned long prev_hop = mach->prev_hop();
                double commitmentBonus = fellowshipDirector->get_commitmentBonus();
                hdr_cmn* cmnh = hdr_cmn::access(p);
                Monitor::TxTime = cmnh->txtime();
                double contributionTx = ((WirelessPhy*)fellowshipDirector->
                        get_netif())->get_TxEnergy(Monitor::TxTime);
                double contribution = contributionTx*(commitmentBonus + 1.0);
                double commitment = fellowshipDirector->
                        updateCommitment(prev_hop, contribution, FORWARDED);
                v->set_count(0);
        }
}
```

Listing B.6: Restoration.cc

```
bool Restoration::recv (Packet* p) {
        /*****************************************************
         If the Packet p gets here it should be forwarded unless
         the network interface queue ifq is full.
         *****************************************************/
        bool forward = true;
        hdr_mac* mach = hdr_mac::access(p);
        unsigned long next_hop = mach->next_hop();
        hdr_sr* srh = hdr_sr::access(p);
        int addr_len = srh->num_addrs();
        unsigned long destination = srh->addrs()[addr_len - 1].addr;
        hdr_cmn* cmnh = hdr_cmn::access(p);
        Monitor* fellowshipDirector = get_director();
        bool is_ifqFull = fellowshipDirector->is_ifqFull(p);
        bool enoughCommitmentNextHop =
                fellowshipDirector->checkCommitment(p, 0.0, next_hop);
        if (is_ifqFull || !enoughCommitmentNextHop) {
                if (!enoughCommitmentNextHop) {
                        unsigned long src = srh->addrs()[0].addr;
                        unsigned long net_id = fellowshipDirector->get_net_id();
                        if(src != net_id) {
                                fellowshipDirector->set_sendRERR(true);
                        }
                }
                forward = false;
        } else if (next_hop != destination) {
                /*****************************************************
                 There is no point in adding the next_hop if it is the
                 destination. The Packet p is being sent out. So take
                 a copy of the unique identifier uid to compare with
                 the packets received in the future.
                 *****************************************************/
                Table* packetBuffer = fellowshipDirector->get_packetBuffer();
                double time = Scheduler::instance().clock();
                int uid = cmnh->uid();
                MACPacketElement* m = new MACPacketElement(time, uid, next_hop);
                /*****************************************************
                 Store the uid in the packetBuffer indexed by the next
```

```
                  hop's id.
                  ****************************************************/
                 packetBuffer->insert(0, m);
                 fellowshipDirector->updateCommitment(next_hop, 0.0, SENT);
        } else if (next_hop == destination) {
                 double commitmentBonus = fellowshipDirector->get_commitmentBonus();
                 Monitor::TxTime = cmnh->txtime();
                 double contributionTx = ((WirelessPhy*)fellowshipDirector->
                         get_netif())->get_TxEnergy(Monitor::TxTime);
                 double contribution = contributionTx*(commitmentBonus + 1.0);
                 fellowshipDirector->updateCommitment(next_hop, contribution, NOTHING);
        } return forward;
}
```

Listing B.7: EnforcementVisitor.cc

```
void EnforcementVisitor::packetMatch (PacketElement* e) {
    MACPacketElement* s = 0;
    if ((s = dynamic_cast<MACPacketElement*>(e))) {
        packetMatch(s);
    }
}

EnforcementVisitor* EnforcementVisitor::get_instance () {
    if (instance == 0) {
            instance = new EnforcementVisitor;
    } return instance;
}
```

Listing B.8: CommitmentElement.cc

```
void CommitmentElement::checkFlow(unsigned long key) {
        if (r_Flows[key] == 0) {
                r_Flows[key] = new RateOfFlow(0, false, 0.0);
        }
}

int CommitmentElement::get_receivedIntervalCount (unsigned long key) {
        int receivedIntervalCount = 0;
        r_Itr = r_Flows.find(key);
        if (r_Itr != r_Flows.end()) {
                receivedIntervalCount =
                        ((*r_Itr).second)->get_receivedIntervalCount();
        } return receivedIntervalCount;
}
```

```
bool CommitmentElement::get_exceededRate (unsigned long key) {
        bool exceededRate = false;
        r_Itr = r_Flows.find(key);
        if (r_Itr != r_Flows.end()) {
                exceededRate = ((*r_Itr).second)->get_exceededRate();
        } return exceededRate;
}

double CommitmentElement::get_intervalStartTime (unsigned long key) {
        double intervalStartTime = 0.0;
        r_Itr = r_Flows.find(key);
        if (r_Itr != r_Flows.end()) {
                intervalStartTime = ((*r_Itr).second)->get_intervalStartTime();
        } return intervalStartTime;
}

void CommitmentElement::set_receivedIntervalCount (unsigned long key, int value) {
        r_Itr = r_Flows.find(key);
        if (r_Itr != r_Flows.end()) {
                ((*r_Itr).second)->set_receivedIntervalCount(value);
        }
}

void CommitmentElement::set_exceededRate (unsigned long key, bool value) {
        r_Itr = r_Flows.find(key);
        if (r_Itr != r_Flows.end()) {
                ((*r_Itr).second)->set_exceededRate(value);
        }
}

void CommitmentElement::set_intervalStartTime (unsigned long key, double value) {
        r_Itr = r_Flows.find(key);
        if (r_Itr != r_Flows.end()) {
                ((*r_Itr).second)->set_intervalStartTime(value);
        }
}
```

Listing B.9: CommitmentStore.cc

```
void CommitmentStore::add (unsigned long n, TableElement* e) {
        commitmentStore.insert(
                std::map<unsigned long,TableElement*>::value_type(n, e));
}

TableElement* CommitmentStore::find (unsigned long n) {
        double time = Scheduler::instance().clock();
        if (commitmentStore[n] == 0) {
                int packetsSent = 0;
                int packetsForwarded = 0;
                commitmentStore[n] = new CommitmentElement(time,CommitmentStore::
                        initialCommitment, packetsSent, packetsForwarded);
        }
        return commitmentStore[n];
}

void CommitmentStore::modify (unsigned long n, TableElement* e) {
        delete commitmentStore[n];
        commitmentStore[n] = e;
}

void CommitmentStore::iterate (unsigned long n, Visitor& v) {
        for(std::map<unsigned long,TableElement*>::iterator commitment =
        commitmentStore.begin(); commitment != commitmentStore.end();
        ++commitment) {
                (*commitment).second->accept(v);
        }
}
```

Listing B.10: MacPacketStore.cc

```
void MACPacketStore::iterate (unsigned long n, Visitor& v) {
        double currentTime = Scheduler::instance().clock();
        std::list<TableElement*>::iterator macPacket = macPacketStore[n].begin();
        while (macPacket != macPacketStore[n].end()) {
                if ((*macPacket)->get_purge()) {
                        macPacket = macPacketStore[n].erase(macPacket);
                        continue;
                }
                (*macPacket)->accept(v);
                double timestamp = (*macPacket)->get_timestamp();
                double timeElapsed = currentTime - timestamp;
                if (timeElapsed >= MACPacketStore::packetExpiration) {
                        /*****************************************************
                         Penalise the dest of frame for not forwarding a packet
                         on the behalf of this node within the MAC packet
                         expiration time.
                         *****************************************************/
                        MACPacketElement* macPacketElement =
                                (MACPacketElement*)(*macPacket);
                        unsigned long id = macPacketElement->get_id();
                        double packetDeliveryRatio =
                                fellowshipDirector->calculate_packetDeliveryRatio(id);
                                double packetDeliveryThreshold =
                                        fellowshipDirector->get_packetDeliveryThreshold();
                                if (packetDeliveryRatio < packetDeliveryThreshold) {
                                        double commitmentPenalty =
                                                fellowshipDirector->get_commitmentPenalty();
                                        double contributionTx = ((WirelessPhy*)
                                                fellowshipDirector->get_netif())->
                                                        get_TxEnergy(Monitor::TxTime);
                                        double contribution =
                                                -1.0*contributionTx*(commitmentPenalty + 1.0);
                                        fellowshipDirector->updateCommitment
                                                (id, contribution, NOTHING);
                                }
                                macPacket = macPacketStore[n].erase(macPacket);
                        } else {
                                ++macPacket;
                        }
                }
}
```

B.2 SMRTI Implementation

In the following, we list the methods that play a significant role in the implementation of the SMRTI model and also a substratum to both the fellowship and SMRTI models.

- **DsrAgent.cc (Listing B.11)** contains the injection points for the SMRTI, the parameters and commands that need to be initialised and invoked from the OTcl level respectively. It also host the code for malicious nodes that can be activated from the OTcl level. In the case of the SMRTI, it is invoked whenever packets are captured promiscuously. Finally, it contains the control mechanism to print the operations of the SMRTI for the purpose of debugging.

- **Path.cc (Listing B.12)** is overloaded to provide the additional operations required for the SMRTI model.

- **PlugInTrust.cc (Listing B.13)** is the core implementation of the SMRTI model from which controls are dispatched for capturing direct, observed and recommended reputations. Furthermore, it contains the modules for determining the trust of a context depending on the embedded policies.

- **DirectReputation.cc (Listing B.14)** facilitates a mobile node to determine whether or not the next-hop neighbour has tampered the packet while forwarding on its behalf.
- **DirectVisitor.cc (Listing B.15)** compares a promiscuously captured packet with the packets stored in the packet-buffer related to direct reputation. On successfully finding a match, it updates the reputation of the next-hop neighbour depending on the exhibited behaviour.
- **ObservedReputation.cc (Listing B.16)** facilitates a mobile node to determine whether or not a neighbour has tampered the packet while forwarding on the behalf of its common neighbour.
- **ObservedVisitor.cc (Listing B.17)** compares a captured packet with the packets stored in the packet-buffer related to observed reputation. On successfully finding a match, it updates the reputation of the common neighbour depending on the exhibited behaviour.
- **RecommendedReputation.cc (Listing B.18)** presents the approach adopted for deriving recommendations from the route contained in a data packet.
- **ReputationStore.cc (Listing B.19)** holds the reputation table and contains modules for searching the reputation of node and updating a node's reputation based on the duration for which there has been no evidence.
- **ReputationElement.cc (Listing B.20)** holds the reputation value of a node and also a module for printing the reputation value of the node for the purpose of debugging.
- **PacketStore.cc (Listing B.21)** maintains the packet-buffer by purging the expired packets.

- **SRElement.cc (Listing B.22)** encompasses packet types related to the DSR protocol, *i.e.* RREQ, RREP, RERR, etc and contains a module for printing their contents for the purpose of debugging.
- **WeightStore.cc (Listing B.23)** translates the context-oriented policies into weights such that those weights are used during the trust evaluation process.
- **PacketCountVisitor.cc (Listing B.24)** facilitates a mobile node to count the packets that have the same identifier in its packet buffer.
- **PrintVisitor.cc (Listing B.25)** enables a mobile node to print the contents of a packet while visiting the packet-buffer.

Listing B.11: DsrAgent.cc

```
// Injections for the operation of SMRTI
#include "DirectVisitor.h"
#include "ObservedVisitor.h"
#include "PacketStore.h"
#include "ReputationStore.h"
#include "ReputationElement.h"
#include "Table.h"

/************************* selectors *************************/
static const bool dsragent_enable_flowstate = false;
static const bool dsragent_prefer_default_flow = false;
// Only respond to the first route request received from a host?
bool dsragent_reply_only_to_first_rtreq = false;
// Send gratuitous replies to effect route shortening?
bool dsragent_send_grat_replies = false;
/*****************************************************
 Consult cache for a route if a xmitfailure is received
 and salvage the packet using the route if possible.
 *****************************************************/
bool dsragent_salvage_with_cache = false;
/*******************************************************
 Send a non-propagating route request as the first action
 in each route discovery action?
 *******************************************************/
bool dsragent_ring_zero_search = false;
// Give errors we forward to our cache?
bool dsragent_snoop_forwarded_errors = true;
```

```
// Should we snoop on any source routes we see?
bool dsragent_snoop_source_routes = true;
/*********************************************************
 Take the data from the last route error msg sent to us and
 propagate it around on the next propagating route request?
 This is aka grat route error propagation.
 *********************************************************/
bool dsragent_propagate_last_error = false;
// Should we listen to a promiscuous tap?
bool dsragent_use_tap = true;
// Consult the route cache before propagating rt req's?
bool dsragent_reply_from_cache_on_propagating = true;
/****************************************************************
 If we have an xmit failure on a packet, and the packet contains a
 route reply, should we scan the reply to see if contains the dead
 link? If it does, we won't salvage the packet unless there's
 something aside from a reply in it (in which case we salvage, but
 cut out the rt reply).
 ****************************************************************/
bool dsragent_dont_salvage_bad_replies = false;
// Do we need to have bidirectional source routes?
bool dsragent_require_bi_routes = true;

#if 0
/***************************************************************
 If we have a cached route to reply to route_request with, should
 we hold off and not send it for a while?
 ***************************************************************/
bool lsnode_holdoff_rt_reply = true;
/**************************************************************
 Require to hear a route requestor use a route before we withold
 our route, or is hearing another (better) route reply enough?
 **************************************************************/
bool lsnode_require_use = true;
#endif

// Prints out what is in forwarded packets
static const int verbose = 0;
// Prints out what is received by the source node
static const int verbose_srr = 0;
// Clean up the trace files
static const int verbose_ssalv = 0;
/************************************************************
 Dump the direct and observed reputation table contents to the
 trace file. This is used as a selector when running tests to
 confirm that the PlugInTrust module is behaving as expected.
 ************************************************************/
bool dsragent_dumpReputationTables = false;
/************************************************************
 Dump the contents of the direct and observed packetTables to
 the trace file.
 ***********************************************************/
bool dsragent_dumpPacketTables = false;

// Print simulation parameters that are activated from OTcl
void DSRAgent::dumpDSRAgentState() {
        Tcl& tcl = Tcl::instance();
        tcl.evalf("puts \"\taddr = %d\"", net_id.getNSAddr_t());
        tcl.evalf("puts \"\tsmrti = %d\"", smrti);
        tcl.evalf("puts \"\tmalicious = %d\"", malicious);
        tcl.evalf("puts \"\tmaliciousChoice = %d\"", maliciousChoice);
        tcl.evalf("puts \"\tsendMaliciousRERR = %d\"", sendMaliciousRERR);
        tcl.evalf("puts \"\tsendingMaliciousRERR = %d\"", sendingMaliciousRERR);
        tcl.evalf("puts \"\tmodifyRouteHeader = %d\"", modifyRouteHeader);
        tcl.evalf("puts \"\tcollusion = %d\"", collusion);
        tcl.evalf("puts \"\tselectiveMisbehaviour = %d\"", selectiveMisbehaviour);
        tcl.evalf("puts \"\tnumberRERRPackets = %d\"", numberRERRPackets);
}

// Print SMRTI parameters that are activated from OTcl
void DSRAgent::dumpPlugInTrustState() {
        Tcl& tcl = Tcl::instance();
        tcl.evalf("puts \"\tcaseWeightFile = %s\"",
                SMRTI.get_caseWeightFile().c_str());
        tcl.evalf("puts \"\tzoneIntervalUpperBound = %f\"",
                PlugInTrust::zoneIntervalUpperBound);
        tcl.evalf("puts \"\tzoneIntervalLowerBound = %f\"",
                PlugInTrust::zoneIntervalLowerBound);
        tcl.evalf("puts \"\tnode addr = %d\"", net_id.getNSAddr_t());
        tcl.evalf("puts \"\tthreshold = %f\"", SMRTI.get_threshold());
        tcl.evalf("puts \"\trecTMD = %f\"", SMRTI.get_recTMD());
        tcl.evalf("puts \"\trecTMO = %f\"", SMRTI.get_recTMO());
        tcl.evalf("puts \"\trecTMR = %f\"", SMRTI.get_recTMR());
        tcl.evalf("puts \"\trecTMP = %f\"", SMRTI.get_recTMP());
        tcl.evalf("puts \"\trecTMW = %f\"", SMRTI.get_recTMW());
        tcl.evalf("puts \"\trecommendedBonus = %f\"", SMRTI.get_recommendedBonus());
        tcl.evalf("puts \"\trecommendedPenalty = %f\"",
                SMRTI.get_recommendedPenalty());
}
```

```
/*****************************************************************
 Create a RERR packet from any kind of packet. This is replicating
 a lot of code but until a better solution is found it has to stay.
 *****************************************************************/
void DSRAgent::createMaliciousRERR(SRPacket& p) {
        hdr_sr* srh = hdr_sr::access(p.pkt);
        hdr_ip* iph =  hdr_ip::access(p.pkt);
        hdr_cmn* cmh = hdr_cmn::access(p.pkt);
        /*****************************************************************
         sendOutPacketWithRoute has incremented the srh->cur_addr() in order to
         forward to the next hop so this has to be undone.
         *****************************************************************/
        --srh->cur_addr();
        /*****************************************************************
         Craft the malicious RERR packet here from the CBR packet. This says that
         the next link from this node is broken and sends back a RERR message.
         *****************************************************************/
        link_down* deadlink = &(srh->down_links()[srh->num_route_errors()]);
        deadlink->addr_type = srh->addrs()[srh->cur_addr()].addr_type;
        deadlink->from_addr = srh->addrs()[srh->cur_addr()].addr;
        deadlink->to_addr = srh->addrs()[srh->cur_addr()+1].addr;
        deadlink->tell_addr = srh->addrs()[0].addr;
        srh->num_route_errors() += 1;
        if (verbose) {
                trace("Sdebug %.5f _%s_ sending into dead-link (nest %d) tell
                %d  %d -> %d", Scheduler::instance().clock(), net_id.dump(),
                srh->num_route_errors(), deadlink->tell_addr, deadlink->from_addr,
                deadlink->to_addr);
        }
        // This is a RERR.
        srh->route_error() = 1;
        srh->route_reply() = 0;
        srh->route_request() = 0;
        srh->flow_header() = 0;
        srh->flow_timeout() = 0;
        // Set up the IP header.
        iph->daddr() = Address::instance().create_ipaddr(deadlink->tell_addr,RT_PORT);
        iph->dport() = RT_PORT;
        iph->saddr() = Address::instance().create_ipaddr(net_id.addr,RT_PORT);
        iph->sport() = RT_PORT;
        iph->ttl() = 255;
        // Set up common header. sendOutPacketWithRoute will craft the rest.
        cmh->ptype() = PT_DSR;
        cmh->size() = IP_HDR_LEN;
        cmh->num_forwards() = 0;
        cmh->uid() = uidcnt_++;
        // Construct the RERR SRPacket.
        p = SRPacket(p.pkt, srh);
        p.route.setLength(p.route.index()+1);
        p.route.reverseInPlace();
        p.trueRoute.setLength(p.trueRoute.index()+1);
        p.trueRoute.reverseInPlace();
        ID tell_id(srh->addrs()[0].addr,
                (ID_Type) srh->addrs()[srh->cur_addr()].addr_type);
        p.dest = tell_id;
        p.src = net_id;
        /*********************************************************************
         The need to uniquely identify RERR is required to match a HELLO packet
         which responds with the same RERR id.
         *********************************************************************/
        srh->rterr_seq() = route_error_num++;
        /*********************************************************************
         Filling in the rest of the details for the RERR packet which would be
         missed since sendOutPacketWithRoute is no longer called when creating
         the RERR packet.
         *********************************************************************/
        p.route.resetIterator();
        p.trueRoute.resetIterator();
        p.route.fillSR(srh);
        p.trueRoute.fillTrueSR(srh);
        cmh->direction() = hdr_cmn::DOWN;
        assert(p.src != net_id || !srh->flow_header());
        cmh->size() += srh->size();
        cmh->next_hop() = srh->get_next_addr();
        cmh->addr_type() = srh->get_next_type();
        srh->cur_addr() = srh->cur_addr() + 1;
        cmh->prev_hop() = net_id.getNSAddr_t();
}

void DSRAgent::maliciousAction(SRPacket& p) {
        hdr_sr* srh = hdr_sr::access(p.pkt);
        hdr_ip* iph =  hdr_ip::access(p.pkt);
        hdr_cmn* cmh = hdr_cmn::access(p.pkt);
        // Initialise random number generator.
        srand(time(0));
        /*************************************************************************
                                          malicious
                                          |
                          _________________________________
                         |                                 |
  sendingMaliciousRERR                   modifyRouteHeader
  $ This node sends 1 RERR               $ This node can create modifications
  for 10 CBR packets.                            (0) add nodes to route
                                                 (1) delete nodes from route
                                                 (2) increment RREQ seq number
  ****************************************************************************/
 if (cmh->ptype() == PT_CBR && sendingMaliciousRERR) {
         int q = (rand()%100);
         // Malicious RERR's probability from 0 -> 100.
         if (q < sendMaliciousRERR) {
                 /*********************************************************
                  Craft the malicious RERR packet here from the CBR packet.
                  This says that the next link from this node is broken and
                  sends back a RERR.
                  *********************************************************/
                 createMaliciousRERR(p);
         }
 } else if (cmh->ptype() == PT_DSR && modifyRouteHeader) {
         /*****************************************************************
          maliciousAction is by default -1 but may be set at the ns-2 tcl
          script. If maliciousAction is set then the tcl script will
          determine what malicious action is elicited by a malicious node.
          *****************************************************************/
         int r;
         if (maliciousChoice > -1) {
                 r = maliciousChoice;
         } else {
                 int r = rand()%3;
         }
         /******************************************************************
          Partition r 3 times for: path addition, path deletion, and route id
          modification.
          ******************************************************************/
         if (r == 0) { // Path addition.
                 if (!p.route.full()) {
                         ID id = ID(rand()%God::instance()->nodes(), ::IP);
                         p.route.appendToPath(id);
                         p.route.resetIterator();
                         cmh->size() -= srh->size();
                         p.route.fillSR(srh);
                 }
         } else if (r == 1) { // Path deletion.
                 /**********************************************************
                  Delete the first node. Note that this must not violate the
                  assmption that this node net_id must remain on the route
```

```
                since the node adds its ID to the route at recv before it
                transmits the packet at sendOutPacketWithRoute.
                ******************************************************/
               int len = p.route.length();
               /*******************************************************
                Only interested in routes that contain 3 or more IDs. We may
                then delete any other node except for the the last 2 IDs.
                *******************************************************/
               if (len <= 1) {
                       // This node is originating packet/next to originator.
                       return;
               } else {
                       // Remove from the route - startIndex to finishIndex.
                       int routeLength = p.route.length();
                       int finishIndex = routeLength - 2;
                       int startIndex = (rand()%finishIndex) + 1;
                       // Remove the first node from the source route.
                       p.route.removeSection(startIndex, finishIndex);
                       p.route.resetIterator();
                       cmh->size() -= srh->size();
                       p.route.fillSR(srh);
               }
       } else if (r == 2) { // Route id modification.
               /*******************************************************
                The route id field of the source routing header is modified.
                The route request sequence number field of the source
                routing header is modified.
                *******************************************************/
               if (srh->route_request()) {
                       ++srh->rtreq_seq();
               } else if (srh->route_reply()) {
                       ++srh->rtrep_seq();
               } else if (srh->route_error()) {
                       ++srh->rterr_seq();
               }
       } else {
               // Should not be here.
       }
       /***********************************************************
        Disabled modification of source and destination as they require
        crypto mechanisms to defend FABRICATION ATTACKS. Current scope
        is confined to modification attacks. Although generation of
        false RERR is an exception to the scope. Generation of false
       RERR is a type of FABRICATION ATTACK. Technically crypto
        solution is required but by forcing to generate RERR 3 times it
```

```
                    becomes detectable. FABRICATION of source or destination address
                    later, once we have studied the impact of the above attacks.

                    else if (r >= 60 && r < 80) { // Source modification.
                            // The source id of the source routing packet is modified.
                            int numberNodes = God::instance()->nodes();
                            int srcAddr = rand()%numberNodes;
                            while (srcAddr == iph->saddr())
                                    srcAddr = rand()%numberNodes;
                            ID srcID = ID(srcAddr, ::IP);
                            p.src = srcID; // SRPacket src ID
                            iph->saddr() = srcAddr; // IP header src addr
                    } else { // Destination modification.
                            // The destination id of source routing packet is modified.
                            int numberNodes = God::instance()->nodes();
                            int destAddr = rand()%numberNodes;
                            while (destAddr == iph->daddr())
                                    destAddr = rand()%numberNodes;
                            ID destID = ID(destAddr, ::IP);
                            p.dest = destID; // SRPacket dest ID
                            iph->daddr() = destAddr; // IP header dest addr
                    }
                    ***************************************************************/
        }
}

// DSRAgent OTcl linkage
static class DSRAgentClass : public TclClass {
public:
        DSRAgentClass() : TclClass("Agent/DSRAgent") {}
        TclObject* create(int, const char*const*) {
                return (new DSRAgent);
        }
        // To vary initial reputation for a ReputationElement.
        virtual void bind();
        virtual int method(int argc, const char*const* argv);
} class_DSRAgent;

// Binding for access from OTcl
void DSRAgentClass::bind() {
        TclClass::bind();
        add_method("setReputationStore");
        add_method("printReputationStore");
        add_method("setPacketExpiration");
        add_method("printPacketExpiration");
        add_method("setObservedVisitor");
        add_method("printObservedVisitor");
        add_method("setDirectVisitor");
        add_method("printDirectVisitor");
        add_method("setZoneIntervalBounds");
        add_method("printNumberRERRPackets");
}

// Parameters that can be set from OTcl
int DSRAgentClass::method(int ac, const char*const* av) {
        Tcl& tcl = Tcl::instance();
        int argc = ac - 2;
        const char*const* argv = av + 2;
        if (argc == 2) {
                if (strcmp(argv[1], "printReputationStore") == 0) {
                        tcl.evalf("puts \" ReputationStore::startingReputation =
                                %f\"", ReputationStore::startingReputation);
                        tcl.evalf("puts \" ReputationStore::expDecay = %f\"",
                                ReputationStore::expDecay);
                        tcl.evalf("puts \" ReputationStore::zoneInterval = %f\"",
                                ReputationStore::zoneInterval);
                        tcl.evalf("puts \" ReputationStore::threshold = %f\"",
                                ReputationStore::threshold);
                        return (TCL_OK);
                }
                if (strcmp(argv[1], "printPacketExpiration") == 0) {
                        tcl.evalf("puts \" PacketStore::packetExpiration = %f\"",
                                PacketStore::packetExpiration);
                        return (TCL_OK);
                }
                if (strcmp(argv[1], "printObservedVisitor") == 0) {
                        ObservedVisitor* v = ObservedVisitor::get_instance();
                        tcl.evalf("puts \" ObservedVisitor::badQID = %f\"",
                                v->get_badQID());
                        tcl.evalf("puts \" ObservedVisitor::badPath = %f\"",
                                v->get_badPath());
                        tcl.evalf("puts \" ObservedVisitor::badDest = %f\"",
                                v->get_badDest());
                        tcl.evalf("puts \" ObservedVisitor::badSrc = %f\"",
                                v->get_badSrc());
                        tcl.evalf("puts \" ObservedVisitor::RERRBonus = %f\"",
                                v->get_RERRBonus());
                        tcl.evalf("puts \" ObservedVisitor::RERRPenalty = %f\"",
                                v->get_RERRPenalty());
                        tcl.evalf("puts \" ObservedVisitor::penaltyReputation = %f\"",
                                v->get_penaltyReputation());
                        return (TCL_OK);
                }
                if (strcmp(argv[1], "printDirectVisitor") == 0) {
                        DirectVisitor* v = DirectVisitor::get_instance();
                        tcl.evalf("puts \" DirectVisitor::bonusReputation =  %f\"",
                                v->get_bonusReputation());
                        tcl.evalf("puts \" DirectVisitor::badQID = %f\"",
                                v->get_badQID());
                        tcl.evalf("puts \" DirectVisitor::badPath = %f\"",
                                v->get_badPath());
                        tcl.evalf("puts \" DirectVisitor::badDest = %f\"",
                                v->get_badDest());
                        tcl.evalf("puts \" DirectVisitor::badSrc = %f\"",
                                v->get_badSrc());
                        tcl.evalf("puts \" DirectVisitor::RERRPenalty = %f\"",
                                v->get_RERRPenalty());
                        tcl.evalf("puts \" DirectVisitor::penaltyReputation = %f\"",
                                v->get_penaltyReputation());
                        return (TCL_OK);
                }
        } else if (argc == 3) {
                if (strcmp(argv[1], "setPacketExpiration") == 0) {
                        PacketStore::packetExpiration = atof(argv[2]);
                        return (TCL_OK);
                }
                if (strcmp(argv[1], "setNumberRERRPackets") == 0) {
                        DSRAgent::numberRERRPackets = atoi(argv[2]);
                        return (TCL_OK);
                }
        } else if (argc == 4) {
                if (strcmp(argv[1], "setZoneIntervalBounds") == 0) {
                        PlugInTrust::zoneIntervalLowerBound = atof(argv[2]);
                        PlugInTrust::zoneIntervalUpperBound = atof(argv[3]);
                        return (TCL_OK);
                }
        } else if (argc == 6) {
                if (strcmp(argv[1], "setReputationStore") == 0) {
                        ReputationStore::startingReputation = atof(argv[2]);
                        ReputationStore::expDecay = atof(argv[3]);
                        ReputationStore::zoneInterval = atof(argv[4]);
                        ReputationStore::threshold = atof(argv[5]);
                        return (TCL_OK);
                }
        } else if (argc == 9) {
                if (strcmp(argv[1], "setObservedVisitor") == 0) {
                        ObservedVisitor* v = ObservedVisitor::get_instance();
                        v->set_badQID(atof(argv[2]));
                        v->set_badPath(atof(argv[3]));
                        v->set_badDest(atof(argv[4]));
                        v->set_badSrc(atof(argv[5]));
                        v->set_RERRBonus(atof(argv[6]));
                        v->set_RERRPenalty(atof(argv[7]));
                        v->set_penaltyReputation(atof(argv[8]));
                        return (TCL_OK);
                }
                if (strcmp(argv[1], "setDirectVisitor") == 0) {
                        DirectVisitor* v = DirectVisitor::get_instance();
                        v->set_bonusReputation(atof(argv[2]));
                        v->set_badQID(atof(argv[3]));
                        v->set_badPath(atof(argv[4]));
                        v->set_badDest(atof(argv[5]));
                        v->set_badSrc(atof(argv[6]));
                        v->set_RERRPenalty(atof(argv[7]));
                        v->set_penaltyReputation(atof(argv[8]));
                        return (TCL_OK);
                }
        } return TclClass::method(ac, av);
}

// Initialising Variables for SMRTI
DSRAgent::DSRAgent(): Agent(PT_DSR), request_table(128), route_cache(NULL),
send_buf_timer(this), flow_table(), ars_table() {
        // This is the overall threshold across which a decision is made.
        bind("threshold", SMRTI.get_thresholdAddr());
        bind("recTMD", SMRTI.get_recTMDAddr());
        bind("recTMO", SMRTI.get_recTMOAddr());
        bind("recTMR", SMRTI.get_recTMRAddr());
        bind("recTMP", SMRTI.get_recTMPAddr());
        bind("recommendedBonus", SMRTI.get_recommendedBonusAddr());
        bind("recommendedPenalty", SMRTI.get_recommendedPenaltyAddr());
        bind("malicious", &malicious);
        bind("smrti", &smrti);
        bind("maliciousChoice", &maliciousChoice);
        bind("sendMaliciousRERR", &sendMaliciousRERR);
        bind("sendingMaliciousRERR", &sendingMaliciousRERR);
        bind("modifyRouteHeader", &modifyRouteHeader);
        bind("collusion", &collusion);
        bind("selectiveMisbehaviour", &selectiveMisbehaviour);
        int c;
        route_request_num = 1;
        route_reply_num = 1;
        route_error_num = 1;
        .
        .
        .
}

// Registering the commands with DSRAgent
int DSRAgent::command(int argc, const char*const* argv) {
        TclObject *obj;
        if (argc == 2) {
                if (strcasecmp(argv[1], "dumpDSRAgent") == 0) {
                        printf("DSRAgent::dumpDSRAgentState");
                        dumpDSRAgentState();
                        return TCL_OK;
                }
                if (strcasecmp(argv[1], "dumpPlugInTrust") == 0) {
                        printf("DSRAgent::dumpPlugInTrustState\n");
                        dumpPlugInTrustState();
                        return TCL_OK;
                }
                .
                .
                .
        } else if(argc == 3) {
                if (strcasecmp(argv[1], "caseWeightFile") == 0) {
                        std::string filename(argv[2]);
                        SMRTI.set_caseWeightFile(filename);
                        SMRTI.loadCaseWeightTable();
                        return TCL_OK;
                }
                .
                .
                .
        } else if (argc == 6) {
                if (strcmp(argv[1], "setReputation") == 0) {
                        SMRTI.get_directReputationTable()->update(atoi(argv[2]),
                                new ReputationElement(Scheduler::instance().clock(),
                                        atof(argv[3])));
                        SMRTI.get_observedReputationTable()->update(atoi(argv[2]),
                                new ReputationElement(Scheduler::instance().clock(),
                                        atof(argv[4])));
                        SMRTI.get_recommendedReputationTable()->update(atoi(argv[2]),
                                new ReputationElement(Scheduler::instance().clock(),
                                        atof(argv[5])));
                        return (TCL_OK);
                }
        } return Agent::command(argc, argv);
}

// Handle packets with MAC destination address of this host, or MAC broadcast addr
void DSRAgent::recv(Packet* packet, Handler*) {
        .
        .
        .
        else if (srh->valid() == 1) {
                if (srh->route_error() && p.src.getNSAddr_t() == cmh->prev_hop()) {
                        int RERRCount = SMRTI.countDuplicatePackets
                                (&p, SMRTI.get_observedRERRTable());
                        /**********************************************************
                         This packet has been tapped. So we need to have 3 matching
                         RERR packets in the observedRERRTable.
                         **********************************************************/
                        if (RERRCount != numberRERRPackets) {
                                // Drop, since we don't handle more than 1 RERR.
                                Packet::free(p.pkt);
                                p.pkt = 0;   // Drop silently.
                                return;
                        }
                }
                .
                .
                .
}

// Handle a packet destined to us
void DSRAgent::handlePacketReceipt(SRPacket& p) {
        hdr_cmn *cmh =  hdr_cmn::access(p.pkt);
        hdr_sr *srh =  hdr_sr::access(p.pkt);
        if (cmh->ptype() == PT_DSR) {
                /**********************************************************
                 Once the SRC of the RREQ receives the RREP, it strips off
                 the path from the RREP to capture the recommendations for
                 the nodes lying in the path except for the previous
                 forwarding node. When the DEST of the RREQ receives the
                 RREQ with complete path, then the recommendations for the
                 nodes lying in the path is computed except for the
                 previous forwarding node
                 **********************************************************/
                if (smrti) {
                        SMRTI.captureRecommendedReputation(&p);
                }
        }
        /*******************************************************
         At present, silently drop the HELLO packet since we have
         not determined how it is to be processed at the dest.
         *******************************************************/
        if (srh->hello()) {
                Packet::free(p.pkt);
                p.pkt = 0;
                return;
        }
        .
        .
        .
}

/*******************************************************************
 See if we can reply to this route request from our cache. If so, do
 it and return true, otherwise, return false.
 *******************************************************************/
bool DSRAgent::replyFromRouteCache(SRPacket &p) {
        .
        .
        .
        // Increment the Reply ID to keep track of the responded replies
        srh->rtrep_seq() = route_reply_num++;
        .
        .
        .
}

/**************************************************************
 Take packet and send it out, packet must a have a route in it.
 Return value is not very meaningful. If fresh is true then
 reset the path before using it, if fresh is false then our
 caller wants us to use a path with the index set as it is.
 **************************************************************/
void DSRAgent::sendOutPacketWithRoute(SRPacket& p, bool fresh, Time delay) {
        .
        .
        .
        // Set the received node ID.
        cmnh->prev_hop() = net_id.getNSAddr_t();
```

```
    if (cmnh->ptype() == PT_DSR && smrti) {
        bool forward = SMRTI.isPacketTrusted(&p);
            if (forward) {
                SMRTI.captureRecommendedReputation(&p);
                if (srh->route_request()) {
                    SMRTI.storePacket(&p, SMRTI.get_directRREQTable(), 0);
                } else if (srh->route_reply()) {
                    /**************************************************
                     Although not relevant, it is still better to put the
                     HELLO packet in the RERR table even it will never be
                     used. Maybe it will be used in the future.
                     **************************************************/
                    if (srh->hello()) {
                        SMRTI.storePacket(&p,
                                SMRTI.get_directRERRTable(), 0);
                    } else {
                        SMRTI.storePacket(&p,
                                SMRTI.get_directRREPTable(), 0);
                    }
                } else if (srh->route_error()) {
                    SMRTI.storePacket(&p, SMRTI.get_directRERRTable(), 0);
                }
            } else {
                if (verbose_srr) {
                    trace( "SRR %.5f _%s_ dropped %s #rreq%d #rrep%d
                    (not trusted)", Scheduler::instance().clock(),
                    net_id.dump(), p.src.dump(), srh->rtreq_seq(),
                    srh->rtrep_seq());
                }
                // We must remove snooped RREQ, RREP, or RERR from the cache
                if (srh->route_request() || srh->route_reply() ||
                srh->route_error()) {
                    route_cache->noticeDeadLink(ID(oldPrevHop, ::IP),
                    p.src, Scheduler::instance().clock());
                }
                Packet::free(p.pkt);
                p.pkt = 0;
                return;
            }
    } else if ((cmnh->ptype() == PT_DSR || cmnh->ptype() == PT_CBR) &&
            malicious && net_id != p.src) {
            /****************************************************
             All exit points for a packet are sendOutPacketWithRoute.
             So sendOutPacketWithRoute will be where the packets are
             finally maliciously modified. This will also ensure that
```

```
                 the PlugInTrust module is not interfered with. Ensure
                 that only DSR and CBR packets are modified.
                 ********************************************************/
                maliciousAction(p);
        } // Jitter the packets.
        if (srh->route_request() || srh->route_reply() && p.jitter) {
                delay += Random::uniform(RREQ_JITTER);
        }
        /*************************************************************
         If this node is initiating a RERR then we need to replicate the
         RERR packet twice.
         *************************************************************/
        if (srh->route_error() && p.src == net_id) {
                replicateRERR(p, numberRERRPackets);
        } else {
                Scheduler::instance().schedule(ll, p.pkt, delay);
        }
        p.pkt = NULL; // packet sent off
}

/************************************************************
 Turn p into a route request and launch it, max_prop of request
 is set as specified.
 ************************************************************/
void DSRAgent::sendOutRtReq(SRPacket &p, int max_prop) {
        hdr_sr *srh =  hdr_sr::access(p.pkt);
        assert(srh->valid());
        srh->route_request() = 1;
        /***********************************************************
         route_request_num is post incremented after being assigned as
         the new rtreq_seq identifier.
         ***********************************************************/
        srh->rtreq_seq() = route_request_num++;
        srh->max_propagation() = max_prop;
        p.trueRoute.reset();
        p.trueRoute.appendToPath(net_id);
        .
        .
        .
}

/***************************************************************
 Take the route in p, add us to the end of it and return the route
 to the sender of p
 ***************************************************************/
```

```
void DSRAgent::returnSrcRouteToRequestor(SRPacket &p) {
        // Add a new sequence number for route reply
        new_srh->rtrep_seq() = route_reply_num++;
    .
        .
        .
        // The packet will be jittered through sendOutPacketWithRoute.
        p_copy.jitter = true;
        sendOutPacketWithRoute(p_copy, true);
}

/*************************************************************
 Process packets that are promiscously listened to from the MAC
 layer tap. Do not change or free packet.
 *************************************************************/
void DSRAgent::tap(const Packet *packet) {
        hdr_sr *srh = hdr_sr::access(packet);
        hdr_ip *iph = hdr_ip::access(packet);
        hdr_cmn *cmh =  hdr_cmn::access(packet);
        if (!dsragent_use_tap) return;
        if (!srh->valid()) return;      // can't do anything with it
        if (!srh->num_addrs()) {
                processFlowARS(packet);
                return;
        }
        SRPacket p((Packet *) packet, srh);
        p.dest = ID((Address::instance().get_nodeaddr(iph->daddr())),::IP);]
        p.src = ID((Address::instance().get_nodeaddr(iph->saddr())),::IP);
        /*************************************************************
         Direct Reputation is captured and if it is successful then quit
         as we are not concerned of any other tasks. Try to capture
         Direct Reputation. On failure try to capture observed
         reputation. Then store the packet for observed reputation. Even
         if the received packet was used in direct reputation, store it
         as it may be useful in observed reputation if some neighbours
         re-transmits it. If the above condition fails then the recieved
         packet may be the first packet to be stored for observed
         reputation or the second packet for the observed reputation to
         be captured and then stored for the recursive process. Unlike
         the above we dont return anything; instead we return after
         storing the packet irrespective of the analysis!
         *************************************************************/
        if (smrti) { // Does this node have its trust module enabled?
                bool directRep = SMRTI.captureDirectReputation(&p);
                if (!directRep) {
```

```
            SMRTI.captureObservedReputation(&p);
        }
        /***************************************
         Store the observed packets on entering
         PlugInTrust::captureObservedReputation.
         ***************************************/
        if (srh->route_request()) {
                SMRTI.storePacket(&p, SMRTI.get_observedRREQTable(),
                        cmh->prev_hop());
        } else if (srh->route_reply()) {
                /**************************************************
                 A HELLO packet is specialised type of RREP packet
                 that resides in the observedRERRTable.
                 **************************************************/
                if (srh->hello()) {
                        SMRTI.storePacket(&p, SMRTI.get_observedRERRTable(),
                                p.dest.getNSAddr_t());
                } else {
                        SMRTI.storePacket(&p, SMRTI.get_observedRREPTable(),
                                cmh->prev_hop());
                }
        } else if (srh->route_error()) {
                SMRTI.storePacket(&p, SMRTI.get_observedRERRTable(),
                        cmh->prev_hop());
        }
        // Is dsragent_dumpRepTables enabled?
        if (dsragent_dumpReputationTables)      {
                /****************************************************
                 Write out the contents of the direct, observed and
                 recommended reputation tables indexed by this
                 nodes net_id to the trace file. This trace is
                 required to check the expected output of the trust
                 module. The data that is required to be traced
                 from the reputation tables are: node id, reputation,
                 timestamp.
                        R 2.55975 -id 1 -tid 0 D -t 2.559748 -r
                        0.600000 O -t 2.559748 -r 0.600000 R -t
                        2.559748 -r 0.600000
                 where the leading R is Reputation, -id is the id of
                 the node indexing the ReputationStores of this node
                 ****************************************************/
                trace("R %.5f -id %d -tid %d %s", Scheduler::
                instance().clock(), net_id.getNSAddr_t(), cmh->prev_hop(),
                SMRTI.dumpReputationTables(cmh->prev_hop()));
        }
```

```
                /*************************************************************
                 Write out the contents of the direct and observed packet tables
                 indexed by the overheard node's id to the trace file.
                 Is dsragent_dumpPacketTables enabled?
                 *************************************************************/
                if (dsragent_dumpPacketTables) {
                        /********************************************************
                         The data that is required to be traced from the reputation
                         tables are: packet type, source address, destination
                         address, query identifier, path and for observed
                         reputation tables - the node id of the node the packet was
                         overheard from.
                         ********************************************************/
                        trace("%.5f -id %d -key %d directRREQTable %s", Scheduler::
                        instance().clock(), net_id.getNSAddr_t(), 0,
                        SMRTI.dumpPacketTable(SMRTI.get_directRREQTable(), 0));

                        trace("%.5f -id %d -key %d directRREPTable %s", Scheduler::
                        instance().clock(), net_id.getNSAddr_t(), 0,
                        SMRTI.dumpPacketTable(SMRTI.get_directRREPTable(), 0));

                        int key;
                        if (p.src.getNSAddr_t() == cmh->prev_hop()) {
                                key = cmh->prev_hop();
                        } else {
                                key = p.trueRoute[p.trueRoute.
                                        prevHopIndex(cmh->prev_hop())].getNSAddr_t();
                        }
                        trace("%.5f -id %d -key %d observedRREQTable %s", Scheduler::
                        instance().clock(), net_id.getNSAddr_t(), key,
                        SMRTI.dumpPacketTable(SMRTI.get_observedRREQTable(), key));

                        trace("%.5f -id %d -key %d observedRERRTable %s", Scheduler::
                        instance().clock(), net_id.getNSAddr_t(), key,
                        SMRTI.dumpPacketTable(SMRTI.get_observedRERRTable(), key));
                }
        }
        /*************************************************************
         A HELLO packet may need to be created here if the originator of the
         RERR packet is the previous hop and the route that was broken was
         supposedly from the prev_hop node to this node. Note that Non-smrti
         nodes also send this packet.
         *************************************************************/
        if (srh->route_error() && (p.src.getNSAddr_t() == cmh->prev_hop()) &&
        (srh->down_links()[0].from_addr == p.src.getNSAddr_t()) &&
        (srh->down_links()[0].to_addr == net_id.getNSAddr_t()) && smrti) {
                if (error_table.get(p.src) >= srh->rterr_seq()) {
                        return;
                } else {
                        /*************************************************************
                         Send a HELLO to record the rterr_seq number. This is so we
                         do not send more than 1 HELLO for each RERR overheard.
                         *************************************************************/
                        error_table.insert(p.src, p.src, srh->rterr_seq());
                }
                // Craft the HELLO packet here. Copy an existing RREP packet
                SRPacket HELLO;
                HELLO.src = net_id;
                HELLO.dest = p.src;
                HELLO.route.appendToPath(net_id);
                HELLO.route.appendToPath(p.src);
                HELLO.trueRoute.appendToPath(net_id);
                HELLO.trueRoute.appendToPath(p.src);
                route_cache->addRoute(HELLO.route,
                        Scheduler::instance().clock(), net_id);
                HELLO.pkt = allocpkt();
                hdr_sr* new_srh = hdr_sr::access(HELLO.pkt);
                new_srh->init();
                for (int i = 0 ; i < HELLO.route.length() ; i++) {
                        HELLO.route[i].fillSRAddr(new_srh->reply_addrs()[i]);
                        HELLO.trueRoute[i].fillSRAddr(new_srh->reply_true_addrs()[i]);
                }
                new_srh->route_reply_len() = HELLO.route.length();
                new_srh->route_request() = 0;
                new_srh->route_reply() = 1;
                new_srh->route_error() = 0;
                new_srh->hello() = 1;
                new_srh->rterr_seq() = srh->rterr_seq();
                hdr_ip* new_iph =  hdr_ip::access(HELLO.pkt);
                new_iph->saddr() = Address::instance().
                        create_ipaddr(HELLO.src.addr, RT_PORT);
                new_iph->sport() = RT_PORT;
                new_iph->daddr() = Address::instance().
                        create_ipaddr(HELLO.dest.addr, RT_PORT);
                new_iph->dport() = RT_PORT;
                new_iph->ttl() = 255;
                hdr_cmn* new_cmnh =  hdr_cmn::access(HELLO.pkt);
                new_cmnh->ptype() = PT_DSR;
                new_cmnh->size() = IP_HDR_LEN;
                if (verbose_srr) {
```

```
                              trace("SRR %.9f _%s_ HELLO-packet-sent %s -> %s #%d
                              (len %d) %s", Scheduler::instance().clock(), net_id.dump(),
                              HELLO.src.dump(), HELLO.dest.dump(), new_srh->rterr_seq(),
                              HELLO.route.length(), HELLO.route.dump());
                    }
                    sendOutPacketWithRoute(HELLO, true);
                    // Decrement directReputationStore entry of malicious RERR src.
                    double reputation = ((ReputationElement*)SMRTI.
                              get_directReputationTable()->retrieve(cmh->
                                        prev_hop()))->get_reputation();
                    double RERRPenalty = DirectVisitor::get_instance()->
                              get_RERRPenalty();
                    SMRTI.get_directReputationTable()->update(cmh->prev_hop(),
                              new ReputationElement(Scheduler::instance().clock(),
                                        reputation + RERRPenalty));
          } return; // Not interested on below func, so return!
          /**************************************************************
           Our code should happen before eveything for the following reasons.
           Consider two packets are sent by node-2 towards node-1 in two
           paths as 2-0-1 and 2-4-0-1. In this case, we can see that node-2
           has node-0 and node-4 as neighbors. So if a packet is travelling
           thro the path 2-4-0-1; then node-2 can sniff the first
           re-transmission of node-4 followed by node-1's re-transmission
           later. This enables node-2 to analyse the two packets and decide
           the observed reputation of node-0. Remaining code eliminates that
           possibility :( As those function concentrates to shorten the
           routes and learn new valid routes, which is not our prime
           objective, we can safely discard the option. Also, it is totally
           opposite to our design plan for direct reputation.
           ***************************************************************/
          .
          .
          .
}

/*********************************************************************
 Mark our route cache reflect the failure of the link between srh[cur_addr]
 and srh[next_addr], and then create a route err message to send to the
 orginator of the pkt (srh[0])
 *********************************************************************/
void DSRAgent::xmitFailed(Packet *pkt, const char* reason) {
          .
          .
          .
          /***************************************************************
          The need to uniquely identify RERR is required to match a HELLO packet
          which responds with the same RERR id.
          ***********************************************************************/
    srh->rterr_seq() = route_error_num++;
        // Send out the Route Error message
        sendOutPacketWithRoute(p, true);
}
```

Listing B.12: Path.cc

```
// Overlaod the Path != operation, for Direct, Observed Visitor
bool Path::operator!=(const Path &rhs) {
        return !operator==(rhs);
}

/**************************************************************
 These functions return the index to a path entry. If the path
 index returned is -1 then the operation failed.
 **************************************************************/
int Path::lastMinusOneIndex() {
        if (len > 1) {
                return len - 2;
        } return -1;
}

int Path::firstIndex() {
        if (len) {
                return 0;
        } return -1;
}

int Path::lastIndex() {
        if (len) {
                return len - 1;
        } return -1;
}

int Path::nextHopIndex(const ID& id) {
        if (len) {
                for (int i = 0; i < len; ++i) {
                        if (path[i] == id) {
                                ++i;
                                if (i != len) {
                                        return i;
                                } else {
                                        break;
                                }
                        }
                }
        } return -1;
}

int Path::prevHopIndex(const ID& id) {
        if (len) {
                for (int i = len - 1; i > -1; --i) {
                        if (path[i] == id) {
                                --i;
                                if (i != -1) {
                                        return i;
                                } else {
                                        break;
                                }
                        }
                }
        } return -1;
}

int Path::prevHopIndex(unsigned long id) {
        if (len) {
                for (int i = len - 1; i > -1; --i) {
                        if (path[i].getNSAddr_t() == id) {
                                --i;
                                if (i != -1) {
                                        return i;
                                } else {
                                        break;
                                }
                        }
                }
        } return -1;
}

int Path::findIndex(const ID& id) const {
// rtn true iff id or MAC_id is in path
        for (int c = 0; c < len ; c++)
                if (path[c] == id)
                        return c;
        return -1;
}
```

Listing B.13: PlugInTrust.cc

```
bool PlugInTrust::captureDirectReputation (SRPacket* p) {
        return directReputation->captureReputation(p);
}

bool PlugInTrust::captureObservedReputation (SRPacket* p) {
        return observedReputation->captureReputation(p);
}

bool PlugInTrust::captureRecommendedReputation (SRPacket* p) {
        return recommendedReputation->captureReputation(p);
}

bool PlugInTrust::isPacketTrusted (SRPacket* p) {
        unsigned long caseNum, src, prev, next, dest;
        caseNum = src = prev = next = dest = 0xffffffff;
        hdr_sr *srh = hdr_sr::access(p->pkt);
        hdr_cmn *cmnh = hdr_cmn::access(p->pkt);
        if (srh->route_request()) {
                if (p->src == net_id) { // Case-1: Initiator of RREQ
                        caseNum = 1;
                        dest = p->dest.getNSAddr_t();
                } else { // Case2/3/4: First node after src/interm/prev to dest
                        caseNum = 2;
                        src = p->src.getNSAddr_t();
                        prev = cmnh->prev_hop();
                        dest = p->dest.getNSAddr_t();
                }
        } else if (srh->route_reply() || srh->route_error()) {
                if (p->src == net_id) {
                        // Case5: Src of RREP/RERR including sub-case no INTERM
                        caseNum = 5;
                        next = cmnh->next_hop();
                        dest = p->dest.getNSAddr_t();
                } else {
                        // Case6/7/8: Next to RREP/RERR src, interm, prev to RREP dest
                        caseNum = 6;
                        src = p->src.getNSAddr_t();
                        prev = cmnh->prev_hop();
                        next = cmnh->next_hop();
                        dest = p->dest.getNSAddr_t();
                }
        }
        WeightElement* we = (WeightElement*)caseWeightTable->retrieve(caseNum);
        WeightElement* weightElement = we->clone();
        weightElement->set_sourceAddr(src);
        weightElement->set_prevHopAddr(prev);
        weightElement->set_nextHopAddr(next);
        weightElement->set_destinationAddr(dest);
        double eT = evaluateTrust(weightElement);
        return (eT >= threshold);
}

double PlugInTrust::trustMetric (Weight* w, unsigned long n) {
        return w->get_directReputation() *
                ((ReputationElement*)directReputationTable->retrieve(n))->
                get_reputation() + w->get_observedReputation() *
                ((ReputationElement*)observedReputationTable->retrieve(n))->
                get_reputation() + w->get_recommendedReputation() *
                ((ReputationElement*)recommendedReputationTable->retrieve(n))->
                get_reputation();
}

double PlugInTrust::evaluateTrust (WeightElement* w) {
        return (w->get_source())->get_participantWeight() *
                trustMetric(w->get_source(), w->get_sourceAddr()) +
                (w->get_prevHop())->get_participantWeight() *
                trustMetric(w->get_prevHop(), w->get_prevHopAddr()) +
                (w->get_nextHop())->get_participantWeight() *
                trustMetric(w->get_nextHop(), w->get_nextHopAddr()) +
                (w->get_destination())->get_participantWeight() *
                trustMetric(w->get_destination(), w->get_destinationAddr());
}

void PlugInTrust::storePacket (SRPacket* p, Table* t, unsigned long n) {
        hdr_sr* srh = hdr_sr::access(p->pkt);
        PacketElement* e;
        double time = Scheduler::instance().clock();
        unsigned long src = p->src.getNSAddr_t();
        unsigned long dest = p->dest.getNSAddr_t();
        Path route = p->route.copy();
        if (srh->route_request()) {
                // We do not consider the Gratuitous RREQ.
                if (srh->rtreq_seq() <= 0) {
                        return;
                } e = new RREQElement(time, RREQ, src, dest, srh->rtreq_seq(), route);
        } else if (srh->route_reply()) {
                if (srh->hello()) {
                        e = new RREPElement
                                (time, HELLO, src, dest, srh->rterr_seq(), route);
                } else { // We do not consider the Gratuitous RREP.
                        if (srh->rtrep_seq() <= 0) {
                                return;
                        }
                        e = new RREPElement
                                (time, RREP, src, dest, srh->rtrep_seq(), route);
                }
        } else if (srh->route_error()) {
                if (srh->rterr_seq() <= 0) {
                        return;
                } e = new RERRElement(time, RERR, src, dest, srh->rterr_seq(), route);
        } t->insert(n, e);
}

const char* PlugInTrust::dumpReputationTables (unsigned long n) const {
        std::string s;
        s += "D" + std::string(directReputationTable->retrieve(n)->dump());
        s += "O" + std::string(observedReputationTable->retrieve(n)->dump());
        s += "R" + std::string(recommendedReputationTable->retrieve(n)->dump());
        return s.c_str();
}

const char* PlugInTrust::dumpPacketTable (Table* t, unsigned long n) const {
        PrintVisitor* p = PrintVisitor::get_instance();
        p->clearOutput();
        t->traverse(n, *p);
        return p->get_output().c_str();
}

int PlugInTrust::countDuplicatePackets (SRPacket* p, Table* t) const {
        PacketCountVisitor* c = PacketCountVisitor::get_instance();
        c->init(p, 0, net_id);
        hdr_sr* srh = hdr_sr::access(p->pkt);
        hdr_cmn* cmnh = hdr_cmn::access(p->pkt);
        observedRERRTable->traverse(cmnh->prev_hop(), *c);
        int count = c->get_count();
        c->set_count(0);
        return count;
}

bool PlugInTrust::loadCaseWeightTable () {
        /*************************************************
         1. Reads case weights for a DSRAgent.
         2. Returns true if successful and false otherwise.
         *************************************************/
        double sD, sO, sR, sP, pD, pO, pR, pP,
                nD, nO, nR, nP, dD, dO, dR, dP;
        std::ifstream fin(caseWeightFile.c_str());
        if (fin.fail()) {
                return false;
        }
        for (int i = 1; i <= 8; ++i) {
                fin >> sD; fin >> sO; fin >> sR; fin >> sP;
                fin >> pD; fin >> pO, fin >> pR; fin >> pP;
                fin >> nD; fin >> nO; fin >> nR; fin >> nP;
                fin >> dD; fin >> dO; fin >> dR; fin >> dP;
                Weight* src = new Weight(sD, sO, sR, sP);
                Weight* prevHop = new Weight(pD, pO, pR, pP);
                Weight* nextHop = new Weight(nD, nO, nR, nP);
                Weight* dest = new Weight(dD, dO, dR, dP);
                WeightElement* weightElement =
                        new WeightElement(src, prevHop, nextHop, dest);
                caseWeightTable->insert(i, weightElement);
        }
        WeightElement* nullWeightElement = new WeightElement
                (new Weight(), new Weight(), new Weight(), new Weight());
        caseWeightTable->insert(0xffffffff, nullWeightElement);
        fin.close();
        return true;
}
```

Listing B.14: DirectReputation.cc

```
bool DirectReputation::captureReputation (SRPacket* p) {
        DirectVisitor* d = DirectVisitor::get_instance();
        PlugInTrust* smrti = get_plugInTrust();
        ID net_id = smrti->get_net_id();
        d->init(p, smrti->get_directReputationTable(), net_id);
        hdr_sr *srh = hdr_sr::access(p->pkt);
        hdr_cmn* cmnh = hdr_cmn::access(p->pkt);
        double time = Scheduler::instance().clock();
        /*****************************************************************
         The trueRoute length must be greater than 2 so that it can contain
         the prev_hop's id and the id of the prev_hop's prev_hop.
         *****************************************************************/
        if (p->trueRoute.length() < 2) {
                return false;
        }
        /*****************************************************************
         Since the last hop node is assumed to not remove itself from the
         trueRoute and therefore it can add a hop on to the end of the
```

```
          trueRoute. The if statement needed to be changed to,
          if (srh->route_request() && net_id ==
          p->route[p->route.prevHopIndex(cmnh->prev_hop())]).
          ****************************************************************/
        if (srh->route_request()) {
                int prevHopIndex = p->trueRoute.prevHopIndex(cmnh->prev_hop());
                // Check route bounds.
                if (prevHopIndex < 0) {
                        return false;
                }
                /****************************************************************
                 The required change performs an update on the previous-hop rather
                 than the last index in the route which would consequently be
                 false if the path was modified by appending another hop.
                 ****************************************************************/
                if (net_id == p->trueRoute[prevHopIndex]) {
                        smrti->get_directRREQTable()->traverse(0, *d);
                        return true;
                }
        } else if (srh->route_reply() || srh->route_error()) {
                int nextHopIndex = p->trueRoute.nextHopIndex(net_id);
                // Check route bounds.
                if (nextHopIndex < 0) {
                        return false;
                }
                if (p->trueRoute.member(net_id) &&
                        p->trueRoute[nextHopIndex].getNSAddr_t()==cmnh->prev_hop()) {
                        if (srh->route_reply()) {
                                smrti->get_directRREPTable()->traverse(0, *d);
                        } else {
                                smrti->get_directRERRTable()->traverse(0, *d);
                        } return true;
                }
        } return false;
}
```

Listing B.15: DirectVisitor.cc

```
void DirectVisitor::packetMatch (PacketElement* e) {
        SRElement* s = 0;
        if ((s = dynamic_cast<SRElement*>(e))) {
                packetMatch(s);
        }
}

DirectVisitor* DirectVisitor::get_instance () {
        if (instance == 0) {
                instance = new DirectVisitor;
        } return instance;
}

void DirectVisitor::packetMatch (SRElement* e) {
        double update = 0;
        SRPacket* p = get_srPacket();
        hdr_sr* srh = hdr_sr::access(p->pkt);
        hdr_cmn* cmh = hdr_cmn::access(p->pkt);
        /************************************************************
         The path should not be removed, instead cmh->prev_hop should
         be added to the Path of the PacketElement e. This ensures
         that path addition and path deletions will be noticed  since
         the path should have been only appended by previous hop.
         ************************************************************/
        Path path = e->get_path().copy();
        if (srh->route_request()) {
                // The path length can never exceed MAX_SR_LEN
                if (path.length() >= MAX_SR_LEN) {
                        return;
                }
                ID id = ID(cmh->prev_hop(), ::IP);
                path.appendToPath(id);
        }
        if (p->src.getNSAddr_t() == e->get_sourceID() && p->dest.getNSAddr_t() ==
        e->get_destinationID() && p->route == path && ((srh->route_request() &&
        srh->rtreq_seq() == e->get_queryID()) || (srh->route_reply() &&
        srh->rtrep_seq() == e->get_queryID()) || (srh->route_error() &&
        srh->rterr_seq() == e->get_queryID()))) {
                update = bonusReputation;
        } else if (p->src.getNSAddr_t() == e->get_sourceID() && p->dest.getNSAddr_t()
        == e->get_destinationID() && p->route == path && ((srh->route_request() &&
        srh->rtreq_seq() != e->get_queryID()) || (srh->route_reply() &&
        srh->rtrep_seq() != e->get_queryID()) || (srh->route_error() &&
        srh->rterr_seq() != e->get_queryID()))) {
                update = badQID;
        } else if (p->src.getNSAddr_t() == e->get_sourceID() && p->dest.getNSAddr_t()
        == e->get_destinationID() && p->route != path && ((srh->route_request() &&
        srh->rtreq_seq() == e->get_queryID()) || (srh->route_reply() &&
        srh->rtrep_seq() == e->get_queryID()) || (srh->route_error() &&
        srh->rterr_seq() == e->get_queryID()))) {
                update = badPath;
        } else if (p->src.getNSAddr_t() == e->get_sourceID() && p->dest.getNSAddr_t()
        != e->get_destinationID() && p->route == path && ((srh->route_request() &&
        srh->rtreq_seq() == e->get_queryID()) || (srh->route_reply() &&
        srh->rtrep_seq() == e->get_queryID()) || (srh->route_error() &&
        srh->rterr_seq() == e->get_queryID()))) {
                update = badDest;
        } else if (p->src.getNSAddr_t() != e->get_sourceID() && p->dest.getNSAddr_t()
        == e->get_destinationID() && p->route == path && ((srh->route_request() &&
        srh->rtreq_seq() == e->get_queryID()) || (srh->route_reply() &&
        srh->rtrep_seq() == e->get_queryID()) || (srh->route_error() &&
        srh->rterr_seq() == e->get_queryID()))) {
                update = badSrc;
        } else { // none of the conditions met so do nothing
                return;
        }
        if (update == 0.0) {
                return;
        }
        double reputation = ((ReputationElement*)get_table()->
                retrieve(cmh->prev_hop()))->get_reputation();
        get_table()->update(cmh->prev_hop(),
                new ReputationElement(Scheduler::instance().clock(),
                        reputation + update));
}
```

Listing B.16: ObservedReputation.cc

```
bool ObservedReputation::captureReputation (SRPacket* p) {
        ObservedVisitor* o = ObservedVisitor::get_instance();
        PlugInTrust* smrti = get_plugInTrust();
        ID net_id = smrti->get_net_id();
        o->init(p, smrti->get_observedReputationTable(), net_id);
        hdr_sr* srh = hdr_sr::access(p->pkt);
        hdr_cmn* cmnh = hdr_cmn::access(p->pkt);
        double time = Scheduler::instance().clock();
        // Enter only if this node is next to the src of RERR or HELLO.
        if ((srh->hello() || srh->route_error()) &&
        p->src.getNSAddr_t() == cmnh->prev_hop()) {
                smrti->get_observedRERRTable()->traverse(cmnh->prev_hop(), *o);
                o->set_HELLOCount(0);
                o->set_RERRCount(0);
        } else if (srh->route_request() || srh->route_reply() ||
        srh->route_error()) {
                /************************************************************
                 Next hop from RREQ/RREP originator so can't capture observed
                 reputation. For RERR, the work has been done earlier, so ok.
                 ************************************************************/
                if (p->src.getNSAddr_t() == cmnh->prev_hop()) {
                        return false;
                }
                int prevHopIndex = p->trueRoute.prevHopIndex(cmnh->prev_hop());
                // To make sure that we never go off the end of a route.
                if (prevHopIndex < 0) {
                        return false;
                }
                unsigned long key = p->trueRoute[prevHopIndex].getNSAddr_t();
                if (srh->route_request() &&
                !smrti->get_observedRREQTable()->retrieveList(key).empty()) {
                        smrti->get_observedRREQTable()->traverse(key, *o);
                } else if (srh->route_reply() &&
                !smrti->get_observedRREPTable()->retrieveList(key).empty()) {
                        smrti->get_observedRREPTable()->traverse(key, *o);
                } else if (srh->route_error() &&
                !smrti->get_observedRERRTable()->retrieveList(key).empty()) {
                        smrti->get_observedRERRTable()->traverse(key, *o);
                }
        } return true;
}
```

Listing B.17: ObservedVisitor.cc

```
void ObservedVisitor::packetMatch (PacketElement* e) {
        SRElement* s = 0;
        if ((s = dynamic_cast< SRElement* >(e))) {
                packetMatch(s);
        }
}

void ObservedVisitor::packetMatch (SRElement* e) {
        double update = 0.0;
        Path path = e->get_path().copy();
        SRPacket* p = get_srPacket();
        hdr_sr* srh = hdr_sr::access(p->pkt);
        hdr_cmn* cmnh = hdr_cmn::access(p->pkt);
        if (srh->hello() || srh->route_error()) {
                if (srh->rterr_seq() == e->get_queryID()) {
                        if (e->get_packetType() == HELLO) {
                                ++HELLOCount;
                        } else {
                                ++RERRCount;
                        }
                        if (srh->route_error() && RERRCount == 2) {
                                set_match(true);
                                update = RERRBonus;
                        }
                        if ((srh->route_error() && RERRCount == 2 && HELLOCount == 1)
                        || (srh->hello() && RERRCount == 3)) {
                                set_match(true);
                                update = RERRPenalty;
                        }
                }
        } else if (srh->route_request() || srh->route_reply()) {
                if (srh->route_request()) {
                        /********************************************************
                         Adding the node ID to the end of the stored packet route
                         is natural since it is what a node should have done in
                         first place rather than maliciously modify the packet.
                         ********************************************************/
                        if (path.length() >= MAX_SR_LEN) {
                                return;
                        }
                        ID id = ID(cmnh->prev_hop(), ::IP);
                        path.appendToPath(id);
                }
                if (p->src.getNSAddr_t() == e->get_sourceID() && p->dest.getNSAddr_t()
                == e->get_destinationID() && p->route == path &&
                ((srh->route_request() && srh->rtreq_seq() != e->get_queryID()) ||
                (srh->route_reply() && srh->rtrep_seq() != e->get_queryID())))  {
                                update = badQID;
                } else if (p->src.getNSAddr_t() == e->get_sourceID() &&
                p->dest.getNSAddr_t() == e->get_destinationID() && p->route != path &&
                ((srh->route_request() && srh->rtreq_seq() == e->get_queryID()) ||
                (srh->route_reply() && srh->rtrep_seq() == e->get_queryID()))) {
                        update = badPath;
                } else if (p->src.getNSAddr_t() == e->get_sourceID() &&
                p->dest.getNSAddr_t() != e->get_destinationID() && p->route == path &&
                ((srh->route_request() && srh->rtreq_seq() == e->get_queryID()) ||
                (srh->route_reply() && srh->rtrep_seq() == e->get_queryID()))) {
                        update = badDest;
                } else if (p->src.getNSAddr_t() != e->get_sourceID() &&
                p->dest.getNSAddr_t() == e->get_destinationID() && p->route == path &&
                ((srh->route_request() && srh->rtreq_seq() == e->get_queryID()) ||
                (srh->route_reply() && srh->rtrep_seq() == e->get_queryID()))) {
                        update = badSrc;
                }
        } else { // none of the conditions met so do nothing
                return;
        }
        if (update == 0.0) {
                return;
        }
        double reputation = ((ReputationElement*)get_table()->
                retrieve(cmnh->prev_hop()))->get_reputation();
        get_table()->update(cmnh->prev_hop(), new ReputationElement(Scheduler::
                instance().clock(), reputation + update));
}
```

Listing B.18: RecommendedReputation.cc

```
bool RecommendedReputation::captureReputation (SRPacket* p) {
        PlugInTrust* smrti = get_plugInTrust();
        hdr_sr* srh = hdr_sr::access(p->pkt);
        ID net_id = smrti->get_net_id();
        if (p->src == net_id) { // RREQ Case1; RREP Case5.
                return false;
        }
        hdr_cmn* cmnh = hdr_cmn::access(p->pkt);
        /*****************************************************
         We do not want to update recommended reputation for packets
         that are not source routing packets.
         *****************************************************/
        if (cmnh->ptype() != PT_DSR) {
                return false;
        }
        /**********************************************************************
         captureRecommendedReputation is called from DSRAgent::sendOutPacketWithRoute.
         Hence the current node's ID "net_id" will be in the path. This applies for
         the source of initiator (case1), first node after the src of RREQ (case2),
         intermediate node (case3), the node prev to the destination of RREQ (case4),
         src of the RREP (case5), the node next to src of RREP (case6), intermediate
         node in RREP (case7) and the node prev to the dest of the RREP (case8). The
         dest of the RREP (which from now onwards can be called as case-RREP-END) is
         also included in the path, though it is called DSRAgent::handlePacketReceipt
         before DSRAgent::acceptRouteReply. The dest of RREQ (which from now onwards
         as case-RREQ-END) unlike the RREP dest is not added to the path, when called
         from DSRAgent::handlePacketReceipt before DSRAgent::returnSrctoRequestor.
         **********************************************************************/
        Path path = p->route.copy();
        if (srh->route_request()) {
                if (p->dest == net_id) {
                        /***************************************************
                         This is the only case where the current node's net_id
                         has not been added to the end of the source route.
                         ***************************************************/
                        path.appendToPath(net_id);
                }
        }
        // Cannot update recommended reputation with only 2 mobile nodes.
        if (path.length() < 3) {
                return false;
        }
        double update;
        double reputation;
        double time = Scheduler::instance().clock();
        for (int i = path.findIndex(net_id) - 1; i >= 1; --i) {
                double recommendersTM = smrti->trustMetric(new Weight(smrti->
                        get_recTMD(), smrti->get_recTMO(), smrti->get_recTMR(),
                                smrti->get_recTMP()), path[i].getNSAddr_t());
                reputation = ((ReputationElement*)smrti->
                        get_recommendedReputationTable()->retrieve
                                (path[i-1].getNSAddr_t()))->get_reputation();
                if (recommendersTM >= smrti->get_threshold()) {
                        update = std::min(reputation + recommendersTM*smrti->
                                get_recommendedBonus(), smrti->
                                        get_zoneIntervalUpperBound());
                } else {
                        update = std::max(reputation + recommendersTM*smrti->
                                get_recommendedPenalty(), smrti->
                                        get_zoneIntervalLowerBound());
                }
                smrti->get_recommendedReputationTable()->update
                        (path[i - 1].getNSAddr_t(), new
                                ReputationElement(time, update));
        } return true;
}
```

Listing B.19: ReputationStore.cc

```
ReputationStore::~ReputationStore() {
        for (std::map<unsigned long, TableElement*>::iterator reputationElement
        = reputationStore.begin(); reputationElement != reputationStore.end();
        ++reputationElement) {
                delete (*reputationElement).second;
        }
}

TableElement* ReputationStore::find (unsigned long n) {
        double time = Scheduler::instance().clock();
        if (reputationStore[n] == 0) {
                reputationStore[n] = new ReputationElement(time,
                        ReputationStore::startingReputation);
        }
        // Reputation update is performed here.
        double reputation = ((ReputationElement*)reputationStore[n])->
                get_reputation();
        if (fmod(reputation, expDecay) != 0) {
                double timestamp = reputationStore[n]->get_timestamp();
                double rc;
                if (reputation >= ReputationStore::threshold) {
                        rc = std::max(reputation - expDecay*(time - timestamp),
                                zoneInterval*(floor(reputation/zoneInterval)));
                } else {
                        rc = std::min(reputation + expDecay*(time - timestamp),
                                zoneInterval*(ceil(reputation/zoneInterval)));
                } modify(n, new ReputationElement(time, rc));
        } return reputationStore[n];
}

void ReputationStore::modify (unsigned long n, TableElement* e) {
        delete reputationStore[n];
        reputationStore[n] = e;
}

void ReputationStore::iterate (unsigned long n, Visitor& v) {
        for (std::map<unsigned long, TableElement*>::iterator reputationElement
        = reputationStore.begin(); reputationElement != reputationStore.end();
        ++reputationElement) {
                (*reputationElement).second->accept(v);
        }
}
```

Listing B.20: ReputationElement.cc

```
void ReputationElement::accept (Visitor& v) {
        v.visit(this);
}

const char* ReputationElement::dump () const {
        static char buf[50];
        char* ptr = buf;
        // An example dump: -t 2.366667 -r 0.677777
        sprintf(ptr, "-t %f -r %f", get_timestamp(), reputation);
        return ptr;
}
```

Listing B.21: PacketStore.cc

```
PacketStore::~PacketStore() {
        for (std::map<unsigned long, std::list<TableElement*>>::iterator m_iter
        = packetStore.begin(); m_iter != packetStore.end(); ++m_iter) {
                for (std::list<TableElement*>::iterator l_iter =
                m_iter->second.begin(); l_iter != m_iter->second.end(); ++m_iter) {
                        delete *l_iter;
                }
        }
}

void PacketStore::iterate (unsigned long n, Visitor& v) {
        std::list<TableElement*>::iterator packetElement = packetStore[n].begin();
        while (packetElement != packetStore[n].end()) {
                if ((*packetElement)->get_purge()) {
                        packetElement = packetStore[n].erase(packetElement);
                        continue;
                }
                (*packetElement)->accept(v);
                if ((Scheduler::instance().clock() - (*packetElement)->
                get_timestamp()) >= PacketStore::packetExpiration) {
                        packetElement = packetStore[n].erase(packetElement);
                } else {
                        ++packetElement;
                }
        }
}

std::set<unsigned long> PacketStore::indexes () {
        std::set<unsigned long> indexes;
        std::map<unsigned long, std::list<TableElement*>>::iterator packetList =
                packetStore.begin();
        while (packetList != packetStore.end()) {
                indexes.insert(packetList->first);
                ++packetList;
        } return indexes;
}
```

Listing B.22: SRElement.cc

```
void SRElement::accept (Visitor& v) {
        v.visit(this);
}

const char* SRElement::dump () const {
        static char buf[200];
        char* ptr = buf;
        sprintf(ptr, "P -T %d -s %d -d %d -q %d -p %s",
                packetType,     sourceID, destinationID, queryID, path.dump());
        return ptr;
}
```

Listing B.23: WeightStore.cc

```
WeightStore::~WeightStore() {
        for (std::map<unsigned long, TableElement*>::iterator weightElement
        = weightStore.begin(); weightElement != weightStore.end();
        ++weightElement) {
                delete (*weightElement).second;
        }
}

void WeightStore::add (unsigned long n, TableElement* e) {
        weightStore.insert(std::map<unsigned long, TableElement*>::value_type(n, e));
}

void WeightStore::modify (unsigned long n, TableElement* e) {
        delete weightStore[n];
        weightStore[n] = e;
}

void WeightStore::iterate (unsigned long n, Visitor& v) {
        for (std::map<unsigned long, TableElement*>::iterator weightElement
        = weightStore.begin(); weightElement != weightStore.end();
        ++weightElement) {
                (*weightElement).second->accept(v);
        }
}
```

Listing B.24: PacketCountVisitor.cc

```
void PacketCountVisitor::packetMatch (SRElement* e) {
        SRPacket* p = get_srPacket();
        hdr_sr* srh = hdr_sr::access(p->pkt);
        if (srh->route_error()) {
                if (srh->rterr_seq() == e->get_queryID()) {
                        ++count;
                }
        }
}
```

Listing B.25: PrintVisitor.cc

```
void PrintVisitor::visit (PacketElement* e) {
        if (dynamic_cast<SRElement*>(e)) {
                output = std::string(output + "\n" + e->dump());
        } else if (dynamic_cast<MACPacketElement*>(e)) { }
}
```

C
Analysis Scripts

Following the implementation of the fellowship and SMRTI models, we conducted detailed simulations involving several scenarios. The results of those simulations that were in the form of trace logs were then processed to compute the performance metrics. Finally, those metrics were represented using graphs for the purpose of analysis. All these processes were automated and accomplished through several scripts. In the following, we present few scripts that form the baseline for most of the automation.

- **CbrGen.py (Listing C.1)** is a customised tool that generates the CBR traffic for different scenarios depending on the varying degree of nodes, CBR traffic flows, their connection durations and the simulation period. This tool replaces the default CBR generator provided with the NS2 to meet the requirement of dynamically changing the CBR traffic flows between different pairs of source and destination mobile nodes. Note that the default CBR traffic generator is tailored to generate the CBR flows between static pairs of source and destination mobile nodes.

- **NS2Scenario.tcl (Listing C.2)** presents an example simulation scenario for evaluating the SMRTI model.
- **NS2Batch.sh (Listing C.3)** presents an example automated process of running multiple simulations for the SMRTI model using a base scenario but by varying a single parameter such as the total count of malicious nodes in the network.
- **TraceParser.py (Listing C.4)** processes the huge volume of trace logs generated by the simulations and contains modules that can compute the required performance metrics.
- **Latency.py (Listing C.5)** relies on *TraceParser.py* to compute the latency metrics for the CBR flows and used as a template for all simulation scenarios involving the fellowship and SMRTI models.
- **LatencyGraph.py (Listing C.6)** collects the output generated by the *Latency.py* for related simulations to draw a graph and used as a template for various simulation scenarios involving the fellowship and SMRTI models.
- **PacketDeliveryRatio.py (Listing C.7)** relies on *TraceParser.py* to compute the PDR metrics for the CBR flows and used as a template for all simulation scenarios involving the fellowship and SMRTI models.
- **PacketDeliveryRatioGraph.py (Listing C.8)** collects the output generated by the *PacketDeliveryRatio.py* for related simulations to draw a graph and used as a template for various simulation scenarios involving the fellowship and SMRTI models.
- **SuccessfulRouteDiscovery.py (Listing C.9)** uses *TraceParser.py* to compute the SRD metrics for the DSR's control messages and used as a template for all simulation scenarios involving the SMRTI model.

- **SuccessfulRouteDiscoveryGraph.py (Listing C.10)** collects the output of *SuccessfulRouteDiscovery.py* for related simulations to draw a graph and used as a template for various simulation scenarios involving the SMRTI model.
- **CompressNamTr.py (Listing C.11)** is an utility that compresses all NAM-oriented trace logs and archives them as scheduled.

Listing C.1: CbrGen.py

```
#!/usr/bin/python

class Connection:
    def __init__(self):
        self._src = -1
        self._dst = -1
        self._startTime = 0.0
        self._finishTime = -1.0

if __name__ == '__main__':
    import random
    import sys

if len(sys.argv) != 6:
    print "./cbrgen.py <number connections> <number nodes> <min connection time>
        <max connection time> <max simulation time>"
    sys.exit(1)

# 1. Setup.
currentTime = 0
udpCount = 0

# Create list of connections.
maximumConnections = int(sys.argv[1])
connections = []
for i in range(maximumConnections):
    connections.append(Connection())

# Create set of free node addresses.
numberNodes = int(sys.argv[2])
freeNodeList = []
for i in range(numberNodes):
    freeNodeList.append(i)
freeNodeSet = set(freeNodeList)
```

```
# Set max and min connection times.
minConnectionTime = int(sys.argv[3])
maxConnectionTime = int(sys.argv[4])
maxSimulationTime = int(sys.argv[5])
currentTime = maxSimulationTime

print '# nodes: %d, max conn: %d, send rate: 4.0, seed: 1' % (numberNodes,
        maximumConnections)

# Set up initial connections.
for connection in connections:
# Choose src from freeNodeSet.
    while 1:
        src = random.randint(0, numberNodes - 1)
        if src in freeNodeSet:
            # Remove src from freeNodeSet.
            freeNodeSet.remove(src)
            break
        else:
            continue
    connection._src = src
    # Choose dst from freeNodeSet.
    while 1:
        dst = random.randint(0, numberNodes - 1)
        if dst != src:
            break
        else:
            continue
    connection._dst = dst
    # Choose length of connection time.
    connectionLength = random.randint(minConnectionTime, maxConnectionTime)
    # Choose finish time.
    finishTime = connection._startTime + connectionLength
    if finishTime > maxSimulationTime:
        finishTime = maxSimulationTime
    connection._finishTime = finishTime
    # Find the lowest connection time.
    if finishTime < currentTime:
        currentTime = finishTime
    # Increment cbr

for connection in connections:
    print '# %d connecting to %d at time %f' % (connection._src, connection._dst,
        connection._startTime)
    print 'set udp_(%d) [new Agent/UDP]' % (udpCount)
    print '$ns_ attach-agent $node_(%d) $udp_(%d)' % (connection._src, udpCount)
    print 'set null_(%d) [new Agent/Null]' % (udpCount)
    print '$ns_ attach-agent $node_(%d) $null_(%d)' % (connection._dst, udpCount)
    print 'set cbr_(%d) [new Application/Traffic/CBR]' % (udpCount)
    print '$cbr_(%d) set packetSize_ 512' % (udpCount)
    print '$cbr_(%d) set interval_ 4.0' % (udpCount)
    print '$cbr_(%d) set random_ 1' % (udpCount)
    print '$cbr_(%d) set maxpkts_ 10000' % (udpCount)
    print '$cbr_(%d) attach-agent $udp_(%d)' % (udpCount, udpCount)
    print '$ns_ connect $udp_(%d) $null_(%d)' % (udpCount, udpCount)
    print '$ns_ at %f \"$cbr_(%d) start\"' % (connection._startTime, udpCount)
    print '$ns_ at %f \"$cbr_(%d) stop\"' % (connection._finishTime, udpCount)
    udpCount += 1

# Setup is complete then we need to start recycling connections.
# Stop condition
while currentTime < maxSimulationTime:
    # Current time holds the lowest connection time.
    for connection in connections:
        # Find the connections with the lowest finishTime.
        if connection._finishTime == currentTime and connection._finishTime !=
        maxSimulationTime:
            # Add the src node back to the freeNodeSet
            freeNodeSet.add(connection._src)
            # Process the next node.
            # Choose src from freeNodeSet.
            while 1:
                src = random.randint(0, numberNodes - 1)
                if src in freeNodeSet:
                    # Remove src from freeNodeSet.
                    freeNodeSet.remove(src)
                    break
                else:
                    continue
            connection._src = src
            # Choose dst from freeNodeSet.
            while 1:
                dst = random.randint(0, numberNodes - 1)
                if dst != src:
                    break
                else:
                    continue
            connection._dst = dst
            # Set start time.
            connection._startTime = connection._finishTime
            # Choose length of connection time.
            connectionLength = random.randint(minConnectionTime, maxConnectionTime)
            # Choose finish time.
            finishTime = connection._startTime + connectionLength
            if finishTime > maxSimulationTime:
                finishTime = maxSimulationTime
            connection._finishTime = finishTime
            # Find the lowest connection time.
            currentTime = finishTime
            # Increment cbr
            udpCount += 1
            print '# %d connecting to %d at time %f' % (connection._src,
                connection._dst, connection._startTime)
            print 'set udp_(%d) [new Agent/UDP]' % (udpCount)
            print '$ns_ attach-agent $node_(%d) $udp_(%d)' % (connection._src,
                udpCount)
            print 'set null_(%d) [new Agent/Null]' % (udpCount)
            print '$ns_ attach-agent $node_(%d) $null_(%d)' % (connection._dst,
                udpCount)
            print 'set cbr_(%d) [new Application/Traffic/CBR]' % (udpCount)
            print '$cbr_(%d) set packetSize_ 512' % (udpCount)
            print '$cbr_(%d) set interval_ 4.0' % (udpCount)
            print '$cbr_(%d) set random_ 1' % (udpCount)
            print '$cbr_(%d) set maxpkts_ 10000' % (udpCount)
            print '$cbr_(%d) attach-agent $udp_(%d)' % (udpCount, udpCount)
            print '$ns_ connect $udp_(%d) $null_(%d)' % (udpCount, udpCount)
            print '$ns_ at %f \"$cbr_(%d) start\"' % (connection._startTime, udpCount)
            print '$ns_ at %f \"$cbr_(%d) stop\"' % (connection._finishTime, udpCount)

    # Then find the next lowest connection time by iterating through the list.
    for connection in connections:
        if connection._finishTime < currentTime:
            currentTime = connection._finishTime
```

Listing C.2: NS2Scenario.tcl

```
if {$argc != 8} {
        puts "Arguments must be in this order: "
        puts "/t<connection pattern file name>"
        puts "/t<scenario file name>"
        puts "/t<trace file>"
        puts "/t<nam file>"
        puts "/t<smrti flag>"
        puts "/t<malicious flag>"
        puts "/t<num malicious nodes>"
        puts "/t<case weights>"
        exit
}

set scenFile [lindex $argv 0]
set cbrFile [lindex $argv 1]
set traceFile [lindex $argv 2]
set namFile [lindex $argv 3]
set smrtiFlag [lindex $argv 4]
set maliciousFlag [lindex $argv 5]
set numMaliciousNodes [lindex $argv 6]
set caseWeights [lindex $argv 7]

# ======================================================================
# Default Script Options
# ======================================================================

set opt(chan)    Channel/WirelessChannel
set opt(prop)    Propagation/TwoRayGround
set opt(netif)   Phy/WirelessPhy
set opt(mac)     Mac/802_11
set opt(ifq)     CMUPriQueue
set opt(ll)      LL
set opt(ant)     Antenna/OmniAntenna
set opt(x)               1200     ; # X dimension of the topography
set opt(y)               1200     ; # Y dimension of the topography
set opt(cp)              $cbrFile
set opt(sc)              $scenFile
set opt(ifqlen)          50       ; # max packet in ifq
set opt(nn)              100      ; # number of nodes
set opt(seed)            0.0      ; # for random generator
set opt(stop)            300      ; # simulation time
set opt(tr)              $traceFile       ; # trace file
set opt(nam)             $namFile         ; # nam file
set opt(rp)              DSR      ; # routing protocol script

# ======================================================================

LL set mindelay_                50us
LL set delay_                   25us
LL set bandwidth_               0         ; # not used
Agent/Null set sport_           0
Agent/Null set dport_           0
Agent/CBR set sport_            0
Agent/CBR set dport_            0
Agent/TCPSink set sport_        0
```

```
Agent/TCPSink set dport_        0
Agent/TCP set sport_            0
Agent/TCP set dport_            0
Agent/TCP set packetSize_       1460

# unity gain, omni-directional antennas
# set up the antennas to be centered in the node and 1.5 meters above it
Antenna/OmniAntenna set X_ 0
Antenna/OmniAntenna set Y_ 0
Antenna/OmniAntenna set Z_ 1.5
Antenna/OmniAntenna set Gt_ 1.0
Antenna/OmniAntenna set Gr_ 1.0

# Initialize the SharedMedia interface with parameters to make
# it work like the 914MHz Lucent WaveLAN DSSS radio interface
# Phy/WirelessPhy set Pt_ 7.214e-3       ;# For 100m transmission range.
# Phy/WirelessPhy set Pt_ 0.2818         ;# For 250m transmission range.
# Phy/WirelessPhy set Pt_ 8.5872e-4      ;# For 40m transmission range.

Phy/WirelessPhy set CPThresh_ 10.0
Phy/WirelessPhy set CSThresh_ 1.559e-11
Phy/WirelessPhy set RXThresh_ 3.652e-10
Phy/WirelessPhy set Rb_ 2*1e6
Phy/WirelessPhy set Pt_ 0.2818          ;# For 250m transmission range.
Phy/WirelessPhy set freq_ 914e+6
Phy/WirelessPhy set L_ 1.0

# ======================================================================
# Main Program
# ======================================================================
# getopt $argc $argv
# Initialize Global Variables
set ns_ [new Simulator]

$ns_ use-newtrace
set tracefd     [open $opt(tr) w]
set nf [open $opt(nam) w]
$ns_ trace-all $tracefd
$ns_ namtrace-all-wireless $nf $opt(x) $opt(y)

set topo        [new Topography]
$topo load_flatgrid $opt(x) $opt(y)

# Create God
set god_ [create-god $opt(nn)]
```

```
# For compatibility and to avoid warning
set chan1 [new $opt(chan)]

# define how node should be created
# global node setting
$ns_ node-config        -adhocRouting $opt(rp) \
                        -llType $opt(ll) \
                        -macType $opt(mac) \
                        -ifqType $opt(ifq) \
                        -ifqLen $opt(ifqlen) \
                        -antType $opt(ant) \
                        -propType $opt(prop) \
                        -phyType $opt(netif) \
                        -channel $chan1 \
                        -topoInstance $topo \
                        -wiredRouting OFF \
                        -agentTrace ON \
                        -routerTrace ON \
                        -macTrace OFF

# Global settings for the DSRAgents.
puts "Printing global settings for DSRAgent."
Agent/DSRAgent setReputationStore 0.51 0.002 0.25 0.5
Agent/DSRAgent printReputationStore
Agent/DSRAgent setPacketExpiration 0.5
Agent/DSRAgent printPacketExpiration
Agent/DSRAgent setObservedVisitor -0.2 -0.2 0.04 -0.2
Agent/DSRAgent printObservedVisitor
Agent/DSRAgent setDirectVisitor -0.2 -0.2 0.04 -0.2
Agent/DSRAgent printDirectVisitor

#  Create the specified number of nodes $opt(nn) and "attach" them
#  the channel.

puts "Configuring MobileNodes."
for {set i 0} {$i < $opt(nn)} {incr i} {
    set node_($i) [$ns_ node]
    $node_($i) random-motion 0          ; # disable random motion
    [$node_($i) get-dsragent] set threshold 0.5
    [$node_($i) get-dsragent] set recTMD 0.5
    [$node_($i) get-dsragent] set recTMO 0.3
    [$node_($i) get-dsragent] set recTMR 0.2
    [$node_($i) get-dsragent] set recTMP 1.0
    [$node_($i) get-dsragent] set recTMW 0.0
    [$node_($i) get-dsragent] caseWeightFile $caseWeights
    [$node_($i) get-dsragent] set smrti $smrtiFlag
    [$node_($i) get-dsragent] set malicious 0
    [$node_($i) get-dsragent] dumpPlugInTrust
    [$node_($i) get-dsragent] dumpDSRAgent
}

# n malicious
set startIndex [expr $opt(nn) - $numMaliciousNodes]
for {set i $startIndex} {$i < $opt(nn)} {incr i} {
    [$node_($i) get-dsragent] set smrti 0
    [$node_($i) get-dsragent] set malicious $maliciousFlag
}

# Source the Connection and Movement scripts
if { $opt(sc) == "" } {
        puts "*** NOTE: no scenario file specified."
        set opt(sc) "none"
} else {
        puts "Loading scenario file..."
        source $opt(sc)
        puts "Load complete..."
}

if { $opt(cp) == "" } {
        puts "*** NOTE: no connection pattern specified."
        set opt(cp) "none"
} else {
        puts "Loading connection pattern..."
        source $opt(cp)
}

# Define initial postion of nodes for NAM
for {set i 0} {$i < $opt(nn)} {incr i} {
        # 20 defines the node size in nam, must adjust it according to your scenario
        # The function is to be called once the mobility model is defined
        $ns_ initial_node_pos $node_($i) 20
}

# Tell all the nodes when the simulation ends
for {set i } {$i < $opt(nn) } {incr i} {
    $ns_ at $opt(stop).000000001 "$node_($i) reset";
}

# tell nam the simulation stop time
```

```
$ns_ at $opt(stop)       "$ns_ nam-end-wireless $opt(stop)"

$ns_ at $opt(stop).1 "puts \"NS EXITING...\" ; $ns_ halt"

proc stop_exit {} {
        exit
}
$ns_ at $opt(stop).2 "stop_exit"

proc stop {} {
    global ns_ tracefd  nf
    $ns_ flush-trace
    close $tracefd
    close $nf
}
$ns_ at $opt(stop) "stop"

puts $tracefd "M 0.0 nn $opt(nn) x $opt(x) y $opt(y) rp $opt(rp)"
puts $tracefd "M 0.0 sc $opt(sc) cp $opt(cp) seed $opt(seed)"
puts $tracefd "M 0.0 prop $opt(prop) ant $opt(ant)"

puts "Starting Simulation..."
$ns_ run
```

Listing C.3: NS2Batch.sh

```
#!/bin/sh

NS2=/home/ns2/ns-allinone-2.28/ns-2.28/ns
NUM_ARGS=$#
SCRIPT_NAME=$0

if [ ${NUM_ARGS} != 6 ]
then
        echo "${SCRIPT_NAME} <tcl script> <cbr basename> <scen basename>
                <num scenfiles> <trace basename> <nam basename>"
        echo "Exiting.."
        exit 0
fi

TCL_SCRIPT=$1
CBR_BASE=$2
SCEN_BASE=$3
NUM_SCEN=$4
TR_BASE=$5
NAM_BASE=$6
```

```
NUM_MALICIOUS=0

while [ $NUM_MALICIOUS -lt 100 ]
do

        for (( SUFFIX = 1 ; SUFFIX<= $NUM_SCEN ; SUFFIX++ ))
        do
                echo "$NS2 $TCL_SCRIPT ${SCEN_BASE}_${SUFFIX}.scen ${CBR_BASE}
                        _${NUM_MALICIOUS}.cbr ${TR_BASE}_${NUM_MALICIOUS}_${SUFFIX}
                        _dsr.tr ${NAM_BASE}_${NUM_MALICIOUS}_${SUFFIX}_dsr.nam
                        0 1 $NUM_MALICIOUS caseWeights"
                $NS2 $TCL_SCRIPT ${SCEN_BASE}_${SUFFIX}.scen ${CBR_BASE}
                        _${NUM_MALICIOUS}.cbr ${TR_BASE}_${NUM_MALICIOUS}_${SUFFIX}
                        _dsr.tr ${NAM_BASE}_${NUM_MALICIOUS}_${SUFFIX}_dsr.nam 0 1
                        $NUM_MALICIOUS caseWeights
                sleep 60
                tar -cvf ${TR_BASE}_${NUM_MALICIOUS}_${SUFFIX}_dsr.tar ${TR_BASE}_
                        ${NUM_MALICIOUS}_${SUFFIX}_dsr.tr
                gzip ${TR_BASE}_${NUM_MALICIOUS}_${SUFFIX}_dsr.tar
                rm ${TR_BASE}_${NUM_MALICIOUS}_${SUFFIX}_dsr.tr
                tar -cvf ${NAM_BASE}_${NUM_MALICIOUS}_${SUFFIX}_dsr.tar ${NAM_BASE}
                        _${NUM_MALICIOUS}_${SUFFIX}_dsr.nam
                gzip ${NAM_BASE}_${NUM_MALICIOUS}_${SUFFIX}_dsr.tar
                rm ${NAM_BASE}_${NUM_MALICIOUS}_${SUFFIX}_dsr.nam

                echo "$NS2 $TCL_SCRIPT ${SCEN_BASE}_${SUFFIX}.scen ${CBR_BASE}
                        _${NUM_MALICIOUS}.cbr ${TR_BASE}_${NUM_MALICIOUS}_${SUFFIX}
                        _smrti_caseWeights.tr ${NAM_BASE}_${NUM_MALICIOUS}_${SUFFIX}
                        _smrti_caseWeights.nam 1 1 $NUM_MALICIOUS caseWeights"
                $NS2 $TCL_SCRIPT ${SCEN_BASE}_${SUFFIX}.scen ${CBR_BASE}
                        _${NUM_MALICIOUS}.cbr ${TR_BASE}_${NUM_MALICIOUS}_${SUFFIX}
                        _smrti_caseWeights.tr ${NAM_BASE}_${NUM_MALICIOUS}_${SUFFIX}
                        _smrti_caseWeights.nam 1 1 $NUM_MALICIOUS caseWeights
                sleep 60
                tar -cvf ${TR_BASE}_${NUM_MALICIOUS}_${SUFFIX}_smrti_caseWeights.tar
                        ${TR_BASE}_${NUM_MALICIOUS}_${SUFFIX}_smrti_caseWeights.tr
                gzip ${TR_BASE}_${NUM_MALICIOUS}_${SUFFIX}_smrti_caseWeights.tar
                rm ${TR_BASE}_${NUM_MALICIOUS}_${SUFFIX}_smrti_caseWeights.tr
                tar -cvf ${NAM_BASE}_${NUM_MALICIOUS}_${SUFFIX}_smrti_caseWeights.tar
                        ${NAM_BASE}_${NUM_MALICIOUS}_${SUFFIX}_smrti_caseWeights.nam
                gzip ${NAM_BASE}_${NUM_MALICIOUS}_${SUFFIX}_smrti_caseWeights.tar
                rm ${NAM_BASE}_${NUM_MALICIOUS}_${SUFFIX}_smrti_caseWeights.nam

                echo "$NS2 $TCL_SCRIPT ${SCEN_BASE}_${SUFFIX}.scen ${CBR_BASE}
```

```
          _${NUM_MALICIOUS}.cbr ${TR_BASE}_${NUM_MALICIOUS}_${SUFFIX}
          _smrti_caseWeightsMinusObserved.tr ${NAM_BASE}_${NUM
          _MALICIOUS}_${SUFFIX}_smrti_caseWeightsMinusObserved.nam
          1 1 $NUM_MALICIOUS caseWeightsMinusObserved"
$NS2 $TCL_SCRIPT ${SCEN_BASE}_${SUFFIX}.scen ${CBR_BASE}
          _${NUM_MALICIOUS}.cbr ${TR_BASE}_${NUM_MALICIOUS}_${SUFFIX}
          _smrti_caseWeightsMinusObserved.tr ${NAM_BASE}_${NUM
          _MALICIOUS}_${SUFFIX}_smrti_caseWeightsMinusObserved.nam
          1 1 $NUM_MALICIOUS caseWeightsMinusObserved
sleep 60
tar -cvf ${TR_BASE}_${NUM_MALICIOUS}_${SUFFIX}
          _smrti_caseWeightsMinusObserved.tar ${TR_BASE}_${NUM
          _MALICIOUS}_${SUFFIX}_smrti_caseWeightsMinusObserved.tr
gzip ${TR_BASE}_${NUM_MALICIOUS}_${SUFFIX}
          _smrti_caseWeightsMinusObserved.tar
rm ${TR_BASE}_${NUM_MALICIOUS}_${SUFFIX}
          _smrti_caseWeightsMinusObserved.tr
tar -cvf ${NAM_BASE}_${NUM_MALICIOUS}_${SUFFIX}_smrti
          _caseWeightsMinusObserved.tar ${NAM_BASE}_${NUM_MALICIOUS}
          _${SUFFIX}_smrti_caseWeightsMinusObserved.nam
gzip ${NAM_BASE}_${NUM_MALICIOUS}_${SUFFIX}
          _smrti_caseWeightsMinusObserved.tar
rm ${NAM_BASE}_${NUM_MALICIOUS}_${SUFFIX}
          _smrti_caseWeightsMinusObserved.nam

echo "$NS2 $TCL_SCRIPT ${SCEN_BASE}_${SUFFIX}.scen ${CBR_BASE}
          _${NUM_MALICIOUS}.cbr ${TR_BASE}_${NUM_MALICIOUS}_${SUFFIX}
          _smrti_caseWeightsMinusRecommended.tr ${NAM_BASE}_${NUM
          _MALICIOUS}_${SUFFIX}_smrti_caseWeightsMinusRecommended.nam
          1 1 $NUM_MALICIOUS caseWeightsMinusRecommended"
$NS2 $TCL_SCRIPT ${SCEN_BASE}_${SUFFIX}.scen ${CBR_BASE}_${NUM
          _MALICIOUS}.cbr ${TR_BASE}_${NUM_MALICIOUS}_${SUFFIX}_smrti
          _caseWeightsMinusRecommended.tr ${NAM_BASE}_${NUM_MALICIOUS}
          _${SUFFIX}_smrti_caseWeightsMinusRecommended.nam 1 1
          $NUM_MALICIOUS caseWeightsMinusRecommended
sleep 60
tar -cvf ${TR_BASE}_${NUM_MALICIOUS}_${SUFFIX}_smrti
          _caseWeightsMinusRecommended.tar ${TR_BASE}_${NUM_MALICIOUS}
          _${SUFFIX}_smrti_caseWeightsMinusRecommended.tr
gzip ${TR_BASE}_${NUM_MALICIOUS}_${SUFFIX}
          _smrti_caseWeightsMinusRecommended.tar
rm ${TR_BASE}_${NUM_MALICIOUS}_${SUFFIX}
          _smrti_caseWeightsMinusRecommended.tr
tar -cvf ${NAM_BASE}_${NUM_MALICIOUS}_${SUFFIX}_smrti
          _caseWeightsMinusRecommended.tar ${NAM_BASE}_${NUM_MALICIOUS}
```

```
                    _${SUFFIX}_smrti_caseWeightsMinusRecommended.nam
            gzip ${NAM_BASE}_${NUM_MALICIOUS}_${SUFFIX}
                    _smrti_caseWeightsMinusRecommended.tar
            rm ${NAM_BASE}_${NUM_MALICIOUS}_${SUFFIX}
                    _smrti_caseWeightsMinusRecommended.nam
    done
    NUM_MALICIOUS=`expr $NUM_MALICIOUS + 10`
done
```

Listing C.4: TraceParser.py

```
#!/usr/bin/python

class PacketEvent:
        def __init__(self, sentTime, byteSize, uid):
                assert(sentTime >= 0)
                assert(byteSize > 0)
                assert(uid >= 0)
                self._sentTime = sentTime
                self._byteSize = byteSize
                self._uid = uid
                self._received = 0
                self._finishTime = None
                # A dictionary from time to drop reason for a packet.
                # Can have more than one reason.
                self._drops = {}

        def received(self, receivedTime):
                assert(receivedTime > self._sentTime)
                self._received = 1
                self._finishTime = receivedTime

        def dropped(self, dropTime, dropReason):
                assert(dropTime >= self._sentTime)
                assert(not self._drops.has_key(dropTime))
                self._drops[dropTime] = dropReason
                if (not self._received):
                        dropTimes = self._drops.keys()
                        dropTimes.sort()
                        self._finishTime = dropTimes[-1]

        def wasDropped(self):
                return self._received == 0 and len(self._drops) > 0

        def wasReceived(self):
                return self._received == 1
        def valid(self):
                if (self._received == 1):
                        assert(self._finishTime != None)
                        return 1
                assert(self._received == 0)
                if (len(self._drops) > 0):
                        assert(self._finishTime != None)
                        return 1
                return 0

        def finishTimeCompare(self, other):
                assert(self.valid() and other.valid())
                return cmp(self._finishTime, other._finishTime)

        def timeTaken(self):
                return float(self._finishTime - self._sentTime)

        def dump(self):
                print "SentT\t\tSize\tRecv?\tUid\tFinT\t"
                print "%s\t%d\t%d\t%s\t%s\t" % (self._sentTime, self._byteSize,
                        self._received, self._uid, self._finishTime)
                print "drops"
                print self._drops

class Flow:
        def __init__(self, saddr, sport, daddr, dport):
                self._packets = {}
                assert(not(saddr == daddr and sport == dport))
                self._saddr = saddr
                self._sport = sport
                self._daddr = daddr
                self._dport = dport
                self._startTime = None
                self._finishTime = None

        def send(self, sentTime, byteSize, uid):
                if self._packets.has_key(uid):
                        self._packets[uid]._sentTime = sentTime
                else:
                        self._packets[uid] = PacketEvent(sentTime, byteSize, uid)
                if (self._startTime == None or sentTime < self._startTime):
                        self._startTime = sentTime
                if (self._finishTime == None or sentTime > self._finishTime):
                        self._finishTime = sentTime
```

```
    def receive(self, receivedTime, uid):
        if not (self._packets.has_key(uid)):
            return
        assert(self._packets.has_key(uid))
        self._packets[uid].received(receivedTime)
        if (self._finishTime == None or receivedTime > self._finishTime):
            self._finishTime = receivedTime

    def drop(self, dropTime, dropReason, uid):
        assert(self._packets.has_key(uid))
        self._packets[uid].dropped(dropTime, dropReason)
        if (self._finishTime == None or dropTime > self._finishTime):
            self._finishTime = dropTime

    def sentPackets(self):
        return len(self._packets)

    def receivedPackets(self):
        receivedCount = 0
        for p in self._packets.values():
            if (p._received):
                receivedCount += 1
        return receivedCount

    def droppedPackets(self):
        dropCount = 0
        for p in self._packets.values():
            if (p.wasDropped()):
                dropCount += 1
        return dropCount

    def dump(self):
        packets = [p for p in self._packets.values() if p.valid()]
        packets.sort(PacketEvent.finishTimeCompare)
        for p in packets:
            p.dump()

    def startTime(self):
        return self._startTime

    def finishTime(self):
        return self._finishTime

    def validPackets(self):
```

```
                return [p for p in self._packets.values() if p.valid()]

        def packetsSentInRange(self, startTime, stopTime):
                return [p for p in self._packets.values() if
                        p.valid() and startTime <= p._finishTime and
                        p._finishTime <= stopTime ]

        def updateTimestamp(self, startTime, uid):
                assert(self._packets.has_key[uid])
                self._packets[uid]._sentTime = startTime

        def packetsReceivedInRange(self, startTime, stopTime):
                return [p for p in self._packets.values() if
                        p.valid() and p.wasReceived() and startTime <= p._finishTime
                        and p._finishTime <= stopTime ]



        def sumReceivedTimes(self):
                sumReceivedTimes = float(0.0)
                for p in self._packets.values():
                        if (p._received):
                                sumReceivedTimes += p.timeTaken()
                return sumReceivedTimes



def getFlows(lines):
        flows = {}
        for line in lines:
                data = line.split()
                if len(data) == 0:
                        continue
                if data[0] == 'Ssb' or data[0] == 'SSendFailure' or
                data[0] == 'M' or data[0] == 'N':
                        continue
                if data[0] == 'Sconfig' or data[0] == 'S' or
                data[0] == 'S$miss' or data[0] == 'SRR':
                        continue
                tags = {}
                for i in range(1, len(data), 2):
                        if i < len(data)-1:
                                tags[data[i]] = data[i+1]
                eventType = data[0]
                if tags.has_key('-P') and tags['-P'] == 'arp'
                and not tags.has_key('-Is'):
                        continue
        if not tags.has_key('-Is') or not tags.has_key('-Id'):
            print 'corrupted trace: '
            print line
            continue
        flowKey = "%s.%s" % (tags['-Is'], tags['-Id'])
        if (tags['-Nl'] == 'AGT'):
            if (eventType == 's'):
                if (not flows.has_key(flowKey)):
                    (saddr, sport) = tags['-Is'].split('.')
                    (daddr, dport) = tags['-Id'].split('.')
                    flows[flowKey] = Flow(int(saddr),
                        int(sport), int(daddr), int(dport))
                flows[flowKey].send(float(tags['-t']),
                    int(tags['-Il']) + 20, tags['-Ii'])
            elif (eventType == 'r'):
                flows[flowKey].receive(float(tags['-t']), tags['-Ii'])
        elif (eventType == 'd'):
            if (not flows.has_key(flowKey)):
                continue
            flows[flowKey].drop(float(tags['-t']), tags['-Nw'],
                tags['-Ii'])
    return flows.values()

def getSuccessRouteDiscoveryFlows(lines):
    flows = {}
    for line in lines:
        data = line.split()
        if len(data) == 0:
            continue
        eventType = data[0]
        if not (eventType == 's' or eventType == 'r'):
            continue
        tags = {}
        for i in range(1, len(data), 2):
            if i < len(data)-1:
                tags[data[i]] = data[i+1]
        if (not tags.has_key('-Hs') or not tags.has_key('-Ni')
            or not tags.has_key('-Nl') or not tags.has_key('-It')
            or not tags.has_key('-Is') or not tags.has_key('-Id')
            or not tags.has_key('-Il') or not tags.has_key('-P')
            or not tags.has_key('-Pq') or not tags.has_key('-Pp')
            or not tags.has_key('-Ps') or not tags.has_key('-Pe')):
            continue
        if (tags['-Nl'] != 'RTR'):
            continue
            if not (tags['-It'] == 'DSR' and tags['-P'] == 'dsr'):
                continue
            srcAddrPort = tags['-Is'].split('.')
            destAddrPort = tags['-Id'].split('.')
            srcDestPair = tags['-Pe'].split('->')
            sendFlowKey = "%s.%s" % (tags['-Is'], tags['-Id'])
            recvFlowKey = "%s.%s" % (tags['-Id'], tags['-Is'])
            if ((eventType == 's') and (tags['-Hs'] == tags['-Ni']) and
                (tags['-Ni'] == srcAddrPort[0]) and (tags['-Pq'] == '1')):
                if (not flows.has_key(sendFlowKey)):
                    flows[sendFlowKey]=Flow(int(srcAddrPort[0]),
                        int(srcAddrPort[1]), int(destAddrPort[0]),
                            int(destAddrPort[1]))
                flows[sendFlowKey].send(float(tags['-t']), int(tags['-Il']) +
                    20, tags['-Ps'])
            elif((eventType == 'r') and (tags['-Hs'] == tags['-Ni']) and
                (tags['-Ni'] == destAddrPort[0]) and (tags['-Pp'] == '1')
                and (srcDestPair[0] != srcDestPair[1]) and
                (srcAddrPort[0] == srcDestPair[1]) and
                (destAddrPort[0] == srcDestPair[0])):
                flows[recvFlowKey].receive(float(tags['-t']), tags['-Ps'])
    return flows.values()

def getLatencyFlows(lines):
    flows = {}
    for line in lines:
        data = line.split()
        if len(data) == 0:
            continue
        eventType = data[0]
        if not (eventType == 's' or eventType == 'r'):
            continue
        tags = {}
        for i in range(1, len(data), 2):
            if i < len(data)-1:
                tags[data[i]] = data[i+1]
        if (not tags.has_key('-Hs') or not tags.has_key('-Ni')
            or not tags.has_key('-Nl') or not tags.has_key('-It')
            or not tags.has_key('-Is') or not tags.has_key('-Id')
            or not tags.has_key('-Il') or not tags.has_key('-Pi')
            or not tags.has_key('-Pn')):
            continue
        if (tags['-Nl'] != 'RTR'):
            continue
        if not (tags['-It'] == 'cbr' and tags['-Pn'] == 'cbr'):
```

```
                        continue
                srcAddrPort = tags['-Is'].split('.')
                destAddrPort = tags['-Id'].split('.')
                flowKey = "%s.%s" % (tags['-Is'], tags['-Id'])
                if ((eventType == 's') and
                        (tags['-Hs'] == tags['-Ni']) and (tags['-Ni'] ==
                        srcAddrPort[0])):
                        if (not flows.has_key(flowKey)):
                                flows[flowKey] = Flow(int(srcAddrPort[0]),
                                        int(srcAddrPort[1]), int(destAddrPort[0]),
                                                int(destAddrPort[1]))
                        flows[flowKey].send(float(tags['-t']), int(tags['-Il']) +
                                20, tags['-Pi'])
                elif (eventType == 'r') and (tags['-Hs'] == tags['-Ni']) and
                (tags['-Ni'] == destAddrPort[0]):
                        flows[flowKey].receive(float(tags['-t']), tags['-Pi'])
        return flows.values()

def getThroughput(traceFile):
        try:
                f = open(traceFile, 'r')
        except IOError:
                print 'cannot open', traceFile
        else:
                flows = getFlows(f)
                f.close()
        received = 0.0
        sent = 0.0
        for flow in flows:
                received += float(flow.receivedPackets())
                sent += float(flow.sentPackets())
        return received/sent



def getLatency(traceFile):
        try:
                f = open(traceFile, 'r')
        except IOError:
                print 'cannot open', traceFile
        else:
                flows = getLatencyFlows(f)
                f.close()
        completedCBRTimes = float(0.0)
        completedCBRCount = float(0.0)
        for flow in flows:
                completedCBRTimes += float(flow.sumReceivedTimes())
                completedCBRCount += float(flow.receivedPackets())
        return completedCBRTimes/completedCBRCount


def getSuccessRouteDiscovery(traceFile):
        try:
                f = open(traceFile, 'r')
        except IOError:
                print 'cannot open', traceFile
        else:
                flows = getSuccessRouteDiscoveryFlows(f)
                f.close()
        received = 0.0
        sent = 0.0
        for flow in flows:
                received += float(flow.receivedPackets())
                sent += float(flow.sentPackets())
        return received/sent
```

Listing C.5: Latency.py

```
#!/usr/bin/python

def getLatency(file):
        tar=tarfile.open(file, 'r:gz')
        tar.extract(tar.next())
        traceFile=tar.getnames()[0]
        tar.close()
        latencyMsg = '''Latency of CBR flows in the file ''' + traceFile + '\n'
        sys.stdout.write(latencyMsg)
        latency = traceParser.getLatency(traceFile)
        latencyValueMsg = '''Latency = %f''' % (latency,) + '\n'
        sys.stdout.write(latencyValueMsg)
        os.remove(traceFile)
        sys.stdout.flush()

if __name__ == '__main__':
        import sys, os, tarfile, gzip, traceParser

        if len(sys.argv) != 2:
                print sys.argv[0] + "."
                sys.exit(0)

        introMsg='''Starting latency analysis.''' + '\n'
        saveout = sys.stdout
        fsock = open('latencyBatch1.log', 'w+')
        sys.stdout = fsock
        sys.stdout.write(introMsg)
        sys.stdout.flush()

        for root, dirs, files in os.walk(sys.argv[1]):
                for file in files:
                        if file.find('.gz') == -1:
                                continue
                        if file.find('trace') != -1:
                                splitStr=file.split('_')
                                if splitStr[3] == 'dsr.tar.gz':
                                        getLatency(file)
                                elif splitStr[3] == 'smrti':
                                        if splitStr[4] == 'caseWeights.tar.gz':
                                                getLatency(file)
                                        elif splitStr[4] ==
                                        'caseWeightsMinusRecommended.tar.gz':
                                                getLatency(file)
                                        elif splitStr[4] ==
                                        'caseWeightsMinusObserved.tar.gz':
                                                getLatency(file)
                                        else:
                                                sys.exit(0)
                                else:
                                        sys.exit(0)
```

Listing C.6: LatencyGraph.py

```
#!/usr/bin/python

if __name__ == '__main__':
        import csv, sys

        f = open('xLatencyBatch1.plt', 'w')
        f.write('set terminal x11' + '\n')
        f.write('set grid' + '\n')
        f.write('set key box' + '\n')
        f.write('set size 1, 1' + '\n')
        f.write('set xlabel \"MN (%)\"' + '\n')
        f.write('set ylabel \"Latecny (sec)\"' + '\n')
        f.write('set title \"Latency Vs Malicious Nodes(MN)\"' + '\n')
        f.write('set xrange [0:100]' + '\n')
        f.write('set yrange [0:0.1]' + '\n')
        f.write('set xtics 10' + '\n')
        f.write('set ytics 0.01' + '\n')
```

```
f.write('plot \"xLatencyBatch1.dat\" u 1:2 t "DSR" w lp lt 1 lw 2 pt 4 ps 2,
        \\' + '\n')
f.write('\t\"xLatencyBatch1.dat\" u 1:3 t "SMRTI" w lp lt 3 lw 2 pt 5 ps 2' +
        '\n')
f.write('set terminal png  notransparent interlace large size 800, 800 crop
        enhanced' + '\n')
f.write('set output \'oLatencyBatch1.png\'' + '\n')
f.write('replot' + '\n')
f.write('set terminal jpeg interlace large size 800, 800 crop enhanced' +
        '\n')
f.write('set output \'oLatencyBatch1.jpeg\'' + '\n')
f.write('replot' + '\n')
f.write('set terminal postscript eps enhanced color blacktext dashed dl 0.5 lw
        2 \"Helevetica\" 15' + '\n')
f.write('set size 1.5, 1.5' + '\n')
f.write('set output \'oLatencyBatch1.ps\'' + '\n')
f.write('replot' + '\n')
f.write('reset' + '\n')
f.close()

f = open('xLatencyBatch1.dat', 'w')
reader = csv.reader(open(sys.argv[1], 'rU'))
title1 = ''
title2 = ''
title3 = ''
xAxis = ''
yAxis1 = ''
yAxis2 = ''
for row in reader:
        if row == []:
                continue
        if row[0] == 'Malicious_Nodes':
                title1 = row[0]
                title2 = row[1]
                title3 = row[2]
                f.write('\n#' + title1 + '\t' + title2 + '\t' + title3 + '\n')
        elif row[0] == 'X-Axis':
                continue
        else:
                xAxis = row[0]
                yAxis1 = row[1]
                yAxis2 = row[2]
                f.write(xAxis + '\t' + yAxis1 + '\t' + yAxis2 + '\n')
f.close()
```

Listing C.7: PacketDeliveryRatio.py

```
#!/usr/bin/python

def getThroughput(file, bucket):
        tar=tarfile.open(file, 'r:gz')
        tar.extract(tar.next())
        traceFile=tar.getnames()[0]
        tar.close()
        index = int(splitStr[1])
        throughPutMsg = '''Analysing throughput of file''' + traceFile + '\n'
        sys.stdout.write(throughPutMsg)
        throughPut = traceParser.getThroughput(traceFile)
        throughPutValueMsg = '''Throughput = %f''' % (throughPut, ) + '\n'
        sys.stdout.write(throughPutValueMsg)
        bucket[index] += throughPut
        os.remove(traceFile)
        sys.stdout.flush()

if __name__ == '__main__':
        import sys, os, tarfile, gzip, traceParser

        if len(sys.argv) != 2:
                print sys.argv[0] + "."
                sys.exit(0)

        introMsg='''Starting throughput analysis.''' + '\n'
        saveout = sys.stdout
        fsock = open('packetDeliveryRatioBatch1.log', 'w+')
        sys.stdout = fsock
        sys.stdout.write(introMsg)
        sys.stdout.flush()

        dsr = {}
        smrti= {}
        noObserved = {}
        noRecommended = {}

        i = 0
        while i < 100:
                dsr[i] = float(0.0)
                smrti[i] = 0.0
                noObserved[i] = 0.0
                noRecommended[i] = 0.0
                i += 10
        for root, dirs, files in os.walk(sys.argv[1]):
                for file in files:
                        if file.find('.gz') == -1:
                                continue
                        if file.find('trace') != -1:
                                splitStr=file.split('_')
                                if splitStr[3] == 'dsr.tar.gz':
                                        getThroughput(file, dsr)
                                elif splitStr[3] == 'smrti':
                                        if splitStr[4] == 'caseWeights.tar.gz':
                                                getThroughput(file, smrti)
                                        elif splitStr[4] ==
                                        'caseWeightsMinusRecommended.tar.gz':
                                                getThroughput(file, noRecommended)
                                        elif splitStr[4] ==
                                        'caseWeightsMinusObserved.tar.gz':
                                                getThroughput(file, noObserved)
                                        else:
                                                sys.exit(0)
                                else:
                                        sys.exit(0)

        i = 0
        while i < 100:
                dsr[i] /= 10.0
                smrti[i] /= 10.0
                noObserved[i] /= 10.0
                noRecommended[i] /= 10.0
                i += 10

        i = 0
        while i < 100:
                finalThroughMsg = '''number of malicious nodes = %i dsr = %f
                        smrti = %f noObserved = %f noRecommended = %f'''
                        % (i*10, dsr[i], smrti[i], noObserved[i], noRecommended[i], )
                print finalThroughMsg
                i += 10
```

Listing C.8: PacketDeliveryRatioGraph.py

```
#!/usr/bin/python

if __name__ == '__main__':
        import csv, sys

        f = open('xPacketDeliveryRatioBatch1.plt', 'w')
```

```
f.write('set terminal x11' + '\n')
f.write('set grid' + '\n')
f.write('set key box' + '\n')
f.write('set size 1, 1' + '\n')
f.write('set xlabel \"MN (%)\"' + '\n')
f.write('set ylabel \"PDR (%)\"' + '\n')
f.write('set title \"Packet Delivery Ratio(PDR) Vs Malicious Nodes(MN)\"' +
        '\n')
f.write('set xrange [0:100]' + '\n')
f.write('set yrange [0:100]' + '\n')
f.write('set xtics 10' + '\n')
f.write('set ytics 10' + '\n')
f.write('plot \"xPacketDeliveryRatioBatch1.dat\" u 1:2 t "DSR" w lp lt 1 lw 2
        pt 4 ps 2, \\' + '\n')
f.write('\t\"xPacketDeliveryRatioBatch1.dat\" u 1:3 t "SMRTI" w lp lt 3 lw 2
        pt 5 ps 2' + '\n')
f.write('set terminal png  notransparent interlace large size 800, 800 crop
        enhanced' + '\n')
f.write('set output \'oPacketDeliveryRatioBatch1.png\'' + '\n')
f.write('replot' + '\n')
f.write('set terminal jpeg interlace large size 800, 800 crop enhanced' +
        '\n')
f.write('set output \'oPacketDeliveryRatioBatch1.jpeg\'' + '\n')
f.write('replot' + '\n')
f.write('set terminal postscript eps enhanced color blacktext dashed dl 0.5 lw
        2 \"Helevetica\" 15' + '\n')
f.write('set size 1.5, 1.5' + '\n')
f.write('set output \'oPacketDeliveryRatioBatch1.ps\'' + '\n')
f.write('replot' + '\n')
f.write('reset' + '\n')
f.close()

f = open('xPacketDeliveryRatioBatch1.dat', 'w')
reader = csv.reader(open(sys.argv[1], 'rU'))
title1 = ''
title2 = ''
title3 = ''
xAxis = ''
yAxis1 = ''
yAxis2 = ''
for row in reader:
        if row == []:
                continue
        elif row[0] == 'Malicious_Nodes':
                title1 = row[0]
```

```
                    title2 = row[1]
                    title3 = row[2]
                    f.write('\n#' + title1 + '\t' + title2 + '\t' + title3 + '\n')
                elif row[0] == 'X-Axis':
                    continue
                else:
                    xAxis = row[0]
                    yAxis1 = row[1]
                    yAxis2 = row[2]
                    f.write(xAxis + '\t' + yAxis1 + '\t' + yAxis2 + '\n')
        f.close()
```

Listing C.9: SuccessfulRouteDiscovery.py

```
#!/usr/bin/python

def getSuccessRouteDiscovery(file):
        tar=tarfile.open(file, 'r:gz')
        tar.extract(tar.next())
        traceFile=tar.getnames()[0]
        tar.close()
        index = int(splitStr[1])
        successRouteDiscoveryMsg = '''Analysing Percentage of Successful Route \
                Discoveries of file ''' + traceFile + '\n'
        sys.stdout.write(successRouteDiscoveryMsg)
        successPercentage = traceParser.getSuccessRouteDiscovery(traceFile)
        successRouteDiscoveryValueMsg = '''Success Percentage = %f '''
                % (successPercentage, ) + '\n'
        sys.stdout.write(successRouteDiscoveryValueMsg)
        os.remove(traceFile)
        sys.stdout.flush()

if __name__ == '__main__':
        import sys, os, tarfile, gzip, traceParser

        if len(sys.argv) != 2:
                print sys.argv[0] + "."
                sys.exit(0)

        introMsg='''Starting successRouteDiscovery analysis.''' + '\n'
        saveout = sys.stdout
        fsock = open('successRouteDiscoveryBatch1.log', 'w+')
        sys.stdout = fsock
        sys.stdout.write(introMsg)
        sys.stdout.flush()
        for root, dirs, files in os.walk(sys.argv[1]):
                for file in files:
                        if file.find('.gz') == -1:
                                continue
                        if file.find('trace') != -1:
                                splitStr=file.split('_')
                                if splitStr[3] == 'dsr.tar.gz':
                                        getSuccessRouteDiscovery(file)
                                elif splitStr[3] == 'smrti':
                                        if splitStr[4] == 'caseWeights.tar.gz':
                                                getSuccessRouteDiscovery(file)
                                        elif splitStr[4] ==
                                        'caseWeightsMinusRecommended.tar.gz':
                                                getSuccessRouteDiscovery(file)
                                        elif splitStr[4] ==
                                        'caseWeightsMinusObserved.tar.gz':
                                                getSuccessRouteDiscovery(file)
                                        else:
                                                sys.exit(0)
                                else:
                                        sys.exit(0)
```

Listing C.10: SuccessfulRouteDiscoveryGraph.py

```
#!/usr/bin/python

if __name__ == '__main__':
        import csv, sys

        f = open('xSuccessRouteDiscoveryBatch1.plt', 'w')
        f.write('set terminal x11' + '\n')
        f.write('set grid' + '\n')
        f.write('set key box' + '\n')
        f.write('set size 1, 1' + '\n')
        f.write('set xlabel \"MN (%)\"' + '\n')
        f.write('set ylabel \"SRD (%)\"' + '\n')
        f.write('set title \"SuccessRouteDiscovery(SRD) Vs Malicious Nodes(MN)\"'
                + '\n')
        f.write('set xrange [0:100]' + '\n')
        f.write('set yrange [0:100]' + '\n')
        f.write('set xtics 10' + '\n')
        f.write('set ytics 10' + '\n')
        f.write('plot \"xSuccessRouteDiscoveryBatch1.dat\" u 1:2 t "DSR" w lp lt 1
                lw 2 pt 4 ps 2, \\' + '\n')
        f.write('\t\"xSuccessRouteDiscoveryBatch1.dat\" u 1:3 t "SMRTI" w lp lt 3
                lw 2 pt 5 ps 2' + '\n')
        f.write('set terminal png  notransparent interlace large size 800, 800 crop
                enhanced' + '\n')
        f.write('set output \'oSuccessRouteDiscoveryBatch1.png\'' + '\n')
        f.write('replot' + '\n')
        f.write('set terminal jpeg interlace large size 800, 800 crop
                enhanced' + '\n')
        f.write('set output \'oSuccessRouteDiscoveryBatch1.jpeg\'' + '\n')
        f.write('replot' + '\n')
        f.write('set terminal postscript eps enhanced color blacktext dashed dl 0.5
                lw 2 \"Helevetica\" 15' + '\n')
        f.write('set size 1.5, 1.5' + '\n')
        f.write('set output \'oSuccessRouteDiscoveryBatch1.ps\'' + '\n')
        f.write('replot' + '\n')
        f.write('reset' + '\n')
        f.close()

        f = open('xSuccessRouteDiscoveryBatch1.dat', 'w')
        reader = csv.reader(open(sys.argv[1], 'rU'))
        title1 = ''
        title2 = ''
        title3 = ''
        xAxis = ''
        yAxis1 = ''
        yAxis2 = ''
        for row in reader:
                if row == []:
                        continue
                if row[0] == 'Malicious_Nodes':
                        title1 = row[0]
                        title2 = row[1]
                        title3 = row[2]
                        f.write('\n#' + title1 + '\t' + title2 + '\t' + title3 + '\n')
                elif row[0] == 'X-Axis':
                        continue
                else:
                        xAxis = row[0]
                        yAxis1 = row[1]
                        yAxis2 = row[2]
                        f.write(xAxis + '\t' + yAxis1 + '\t' + yAxis2 + '\n')
        f.close()
```

Listing C.11: CompressNamTr.py

```
#!/usr/bin/python


if __name__ == '__name__':
```

```
import os, sys, tarfile, gzip

if (len(sys.argv[0] + " <directory>":
        sys.exit(0)

for root, dirs, files in os.walk(sys.argv[1]):
        for file in files:
                if file.find('.gz') != −1:
                        continue
                if file.find('.nam') != −1 or file.find('.tr') != −1:
                        tarName = file + '.tar'
                        tar = tarfile.open(tarName, 'w')
                        tar.add(file)
                        tar.close()
                        file = open(tarName, 'r')
                        fileContents = file.read()
                        gzipName = tarName + '.gz'
                        gzipFile = gzip.GzipFile(gzipName, 'wb')
                        gzipFile.write(fileContents)
                        gzipFile.close()
                        os.remove(tarName)
```

D

Scalable Multi-service Group Key Management

D.1 Encryption-Decryption Process in SMG

Node $\mathcal{A}$ sets up the following parameters in order to achieve l different sensitivity levels for communication flows via intermediate nodes.

- A random number $r^{\mathcal{A}} \in \mathbb{Z}$,
- Computes $R^{\mathcal{A}} = r^{\mathcal{A}} P^{\mathcal{A}}$,
- Computes $x_{\mathcal{I}}^{\mathcal{A}} = \hat{e}(r^{\mathcal{A}} Q_{\mathcal{I}}^{\mathcal{A}}, P_{pub}^{\mathcal{A}})$; where $\hat{e}$ is the *Weil pairing mapping*,

Node $\mathcal{A}$ then computes the polynomial function $f^k(x)$ as given in (D.1), where k denotes the sensitivity level of a communication flow $CG_k^{\mathcal{A}}$.

$$
\begin{aligned}
f^k(x) &= \prod_{\mathcal{I}=1}^{\mathcal{M}} (x - x_{\mathcal{I}}^{\mathcal{A}}) \bmod p^{\mathcal{A}};\\
&\Updownarrow\\
\prod_{\mathcal{I}=1}^{\mathcal{M}} (x - x_{\mathcal{I}}^{\mathcal{A}}) &= \sum_{\mathcal{I}=0}^{\mathcal{M}} (a_{\mathcal{I}k}^{\mathcal{A}} x_{\mathcal{I}}^{\mathcal{A}}) \bmod p^{\mathcal{A}};\\
&\Updownarrow\\
m &= \sum_{\mathcal{I}=1} S_{\mathcal{I}k}^{\mathcal{A}}; \quad (1 \leqslant \mathcal{I} \leqslant n); \quad (1 \leqslant k \leqslant l);
\end{aligned}
\tag{D.1}
$$

From (D.1), the following can be obtained:

$$
\begin{aligned}
a_{0k}^{\mathcal{A}} &= \prod_{\mathcal{J}=1}^{\mathcal{M}} (-x_{\mathcal{J}}^{\mathcal{A}});\\
a_{1k}^{\mathcal{A}} &= \sum_{\mathcal{I}=1}^{\mathcal{M}} \prod_{\mathcal{J}=1, \mathcal{J}\neq\mathcal{I}}^{\mathcal{M}} (-x_{\mathcal{J}}^{\mathcal{A}});\\
&\vdots\\
a_{\mathcal{M}-2,k}^{\mathcal{A}} &= \sum_{\mathcal{I}=1}^{\mathcal{A}} \sum_{\mathcal{J}=\mathcal{I}+1}^{\mathcal{A}} (-x_{\mathcal{I}}^{\mathcal{A}})(-x_{\mathcal{J}}^{\mathcal{A}});\\
a_{\mathcal{M}-1,k}^{\mathcal{A}} &= \sum_{\mathcal{J}=1}^{\mathcal{M}} (-x_{\mathcal{J}}^{\mathcal{A}});\\
a_{\mathcal{M}k}^{\mathcal{A}} &= 1;
\end{aligned}
\tag{D.2}
$$

Since the set $\{a_{\mathcal{I}k}^{\mathcal{A}}\}$ satisfies $\sum_{\mathcal{I}=0}^{\mathcal{M}} (a_{\mathcal{I}k}^{\mathcal{A}} x_{\mathcal{J}k}^{\mathcal{A}}) = 0 \bmod p^{\mathcal{A}}$, where $\mathcal{J} = 1, \cdots, \mathcal{M}$; the set $\{a_{\mathcal{I}k}^{\mathcal{A}}\}$ is used to construct exponential functions as given in (D.3).

$$
\left\{a_{0k}^{\mathcal{A}} P^{\mathcal{A}},\, a_{1k}^{\mathcal{A}} P^{\mathcal{A}},\, a_{2k}^{\mathcal{A}} P^{\mathcal{A}},\, \cdots, a_{\mathcal{N}k}^{\mathcal{A}} P^{\mathcal{A}}\right\} \equiv \left\{P_{0k}^{\mathcal{A}},\, P_{1k}^{\mathcal{A}},\, P_{2k}^{\mathcal{A}},\, \cdots, P_{\mathcal{M}k}^{\mathcal{A}}\right\} \tag{D.3}
$$

D.1.1 Encryption

Note that in (D.4), $R_k^{\mathcal{A}} \in \mathbb{Z}$ and $D_k^{\mathcal{A}} \in \mathbb{G}_1^{\mathcal{A}}$.

$$\begin{aligned}
Auth_k^{\mathcal{A}} \leftarrow & \big(R^{\mathcal{A}},\, SK_k^{\mathcal{A}} \oplus H_2(D_k^{\mathcal{A}}),\, D_k^{\mathcal{A}} + R_k^{\mathcal{A}} P_{0k}^{\mathcal{A}},\, R_k^{\mathcal{A}} P_{1k}^{\mathcal{A}},\, \cdots, R_k^{\mathcal{A}} P_{\mathcal{M}k}^{\mathcal{A}}\big) \\
= & \big(R^{\mathcal{A}},\, C_k^{\mathcal{A}},\, C_{0k}^{\mathcal{A}},\, C_{1k}^{\mathcal{A}},\, \cdots, C_{\mathcal{M}k}^{\mathcal{A}}\big)
\end{aligned} \tag{D.4}$$

D.1.2 Decryption

$$\begin{aligned}
\hat{e}(S_{\mathcal{I}}^{\mathcal{A}}, R^{\mathcal{A}}) &= \hat{e}(s^{\mathcal{A}} Q_{\mathcal{I}}^{\mathcal{A}}, r^{\mathcal{A}} P^{\mathcal{A}}) = \hat{e}(r^{\mathcal{A}} Q_{\mathcal{I}}^{\mathcal{A}}, P_{pub}^{\mathcal{A}}) = x_{\mathcal{I}}^{\mathcal{A}} \\
C_{0k}^{\mathcal{A}} + \sum_{\mathcal{J}=1}^{\mathcal{M}} (x_{\mathcal{I}\mathcal{J}}^{\mathcal{A}} \cdot C_{\mathcal{J}k}^{\mathcal{A}}) &= D_k^{\mathcal{A}} + R_k^{\mathcal{A}} \big(a_{0k}^{\mathcal{A}},\, +a_{1k}^{\mathcal{A}} x_{\mathcal{I}}^{\mathcal{A}} + \cdots,\, +a_{\mathcal{M}k}^{\mathcal{A}} x_{\mathcal{M}\mathcal{I}}^{\mathcal{A}}\big) P^{\mathcal{A}} \\
&= D_k^{\mathcal{A}} \\
C_k^{\mathcal{A}} \oplus D_k^{\mathcal{A}} &= SK_k^{\mathcal{A}}
\end{aligned} \tag{D.5}$$

References

[1] Imad Aad, Jean-Pierre Hubaux, and Edward W. Knightly, *Denial of service resilience in ad hoc networks*, In Proceedings of the 10th Annual International Conference on Mobile Computing and Networking (MobiCom'04), 2004, pp. 202–215.

[2] Mehran Abolhasan, Tadeusz A. Wysocki, and Eryk Dutkiewicz, *A review of routing protocols for mobile ad hoc networks*, Ad Hoc Networks **2** (2004), no. 1, 1–22.

[3] William Joseph Adams, George C. Hadjichristofi, and Nathaniel J. Davis IV, *Calculating a node's reputation in a mobile ad hoc network*, In Proceedings of the 24th IEEE International Performance, Computing, and Communications Conference (IPCCC'05), April 2005, pp. 303–307.

[4] Sungjoon Ahn and A. Udaya Shankar, *Adapting to route-demand and mobility (arm) in ad-hoc network routing*, In Proceedings of the 9th International Conference on Network Protocols, November 2001, pp. 44–52.

[5] Marco D. Aime and Antonio Lioy, *Incremental trust: Building trust from past experience*, In Proceedings of the International Symposium on a World of Wireless, Mobile and Multimedia Networks, vol. 3, 2005, pp. 603–608.

[6] Ian F. Akyildiz, Xudong Wang, and Weilin Wang, *Wireless mesh networks: A survey*, Computer Networks **47** (2005), no. 4, 445–487.

[7] Mohammad Al-Shurman, Seong-Moo Yoo, and Seungjin Park, *Black hole attack in mobile ad hoc networks*, In Proceedings of the 42nd Annual South-East Regional Conference, 2004, pp. 96–97.

[8] Sathishkumar Alampalayam and Anup Kumar, *An adaptive and predictive security model for mobile ad hoc networks*, Wireless Personal Communications **29** (2004), no. 3-4, 263–281.

[9] Ammar Alkassar and Susanne Wetzel, *An untraceable coin-based incentive scheme for multi-hop networks*, In Proceedings of the 1st IEEE Workshop on Cryptography over Ad hoc Networks (WCAN'05), June 2005, pp. 323–329.

[10] Eitan Altman, Arzad Alam Kherani, Pietro Michiardi, and Refik Molva, *Noncooperative forwarding in ad-hoc networks*, In Proceedings of the 4th IFIP International Conferences on Networking (NETWORKING'05), Lecture Notes in Computer Science, vol. 3462, May 2005, pp. 486–498.

[11] Yi an Huang, Wei Fan, Wenke Lee, and Philip S. Yu, *Cross-feature analysis for detecting ad-hoc routing anomalies*, In Proceedings of the 23rd International Conference on Distributed Computing Systems (ICDCS'03), 2003, pp. 478–487.

[12] Yi an Huang and Wenke Lee, *A cooperative intrusion detection system for ad hoc networks*, In Proceedings of the 1st ACM Workshop on Security of Ad hoc and Sensor Networks (SASN'03), 2003, pp. 135–147.

[13] ______, *Attack analysis and detection for ad hoc routing protocols*, In Proceedings of the 7th International Symposium on Recent Advances in Intrusion Detection (RAID'04), Lecture Notes in Computer Science, vol. 3224, 2004, pp. 125–145.

[14] Luzi Anderegg and Stephan Eidenbenz, *Ad hoc-vcg: A truthful and cost-efficient routing protocol for mobile ad hoc networks with selfish agents*, In Proceedings of the 9th Annual International Conference on Mobile Computing and Networking, 2003, pp. 245–259.

[15] Farooq Anjum and Petros Mouchtaris, *Security for wireless ad hoc networks*, John Wiley & Sons, 2006.

[16] Farooq Anjum, Dhanant Subhadrabandhu, and Saswati Sarkar, *Signature based intrusion detection for wireless ad-hoc networks: A comparative study of various routing protocols*, In Proceedings of the IEEE Vehicular Technology Conference (VTC'03-Fall), vol. 3, October 2003, pp. 2152–2156.

[17] Farooq Anjum and Rajesh Talpade, *Lipad: Lightweight packet drop detection for ad hoc networks*, In Proceedings of the 60th IEEE Vehicular Technology Conference (VTC'04-Fall), September 2004, pp. 1233–1237.

[18] Geneviève Arboit, Claude Crépeau, Carlton R. Davis, and Muthucumaru Maheswaran, *A localized certificate revocation scheme for mobile ad hoc networks.*, Ad Hoc Networks **6** (2008), no. 1, 17–31.

[19] Patroklos G. Argyroudis and Donal O'Mahony, *Secure routing for mobile ad hoc networks*, IEEE Communications Surveys and Tutorials **7** (2005), no. 1-4, 2–21.

[20] Tuomas Aura and Silja Mäki, *Towards a survivable security architecture for ad-hoc networks*, In Proceedings of the 9th International Workshop on Security Protocols, Lecture Notes in Computer Science, vol. 2467, April 2002, pp. 74–79.

[21] Baruch Awerbuch, Reza Curtmola, David Holmer, Cristina Nita-rotaru, and Herbert Rubens, *Mitigating byzantine attacks in ad hoc wireless networks*, Technical report, Department of Computer Science, Johns Hopkins University, March 2004.

[22] Farag Azzedin and Muthucumaru Maheswaran, *Evolving and managing trust in grid computing systems*, In Proceedings of the IEEE Canadian Conference on Electrical and Computer Engineering (CCECE'02), 2002, pp. 1424–1429.

[23] Venkat Balakrishnan and Vijay Varadharajan, *Designing secure wireless mobile ad hoc networks*, In Proceedings of the 19th IEEE International Conference on Advanced Information Networking and Applications (AINA'05), March 2005, pp. 5–8.

[24] ______, *Fellowship in mobile ad hoc networks*, In Proceedings of the 1st IEEE International Conference on Security and Privacy in Communications Networks (SecureComm'05), September 2005, pp. 225–227.

[25] ______, *Packet drop attack: A serious threat to operational mobile ad hoc networks*, In Proceedings of the International Conference on Networks and Communication Systems (NCS'05), April 2005, pp. 89–95.

[26] Venkat Balakrishnan, Vijay Varadharajan, and Uday Kiran Tupakula, *Handbook of communication networks and distributed systems*, (Accepted for publication).

[27] ______, *Fellowship: Defense against flooding and packet drop attacks in manet*, In Proceedings of the 10th IEEE/IFIP Network Operations and Management Symposium (NOMS'06), April 2006, pp. 1–4.

[28] ______, *Subjective logic based trust model for mobile ad-hoc networks*, In Proceedings of the 4th ACM International Conference on Security and Privacy in Communication Networks (Se cureComm'08), September 2008.

[29] ______, *Handbook of wireless ad hoc and sensor networks*, ch. Trust Management in Mobile Ad hoc Networks, Springer (London), 2009.

[30] ______, *Mobile intelligence: Mobile computing and computational intelligence*, ch. SMRTI:: Secure MANET Routing with Trust Intrigue, John Wiley and Sons Ltd., 2010.

[31] Venkat Balakrishnan, Vijay Varadharajan, Uday Kiran Tupakula, and Phillip Lucs, *Team: Trust enhanced security architecture for mobile ad-hoc networks*, In Proceedings of the 15th IEEE International Conference on Networks (ICON'07), November 2007, pp. 182–187.

[32] ______, *Trust and recommendations in mobile ad hoc networks*, In Proceedings of the 3rd International Conference on Networking and Services (ICNS'07), June 2007, pp. 64–69.

[33] ______, *Trust enhanced secure mobile ad hoc network routing*, In Proceedings of the 21st IEEE International Conference on Advanced Information Networking and Applications Workshops(AINAW'07), May 2007, pp. 27–33.

[34] ______, *Trust integrated cooperation architecture for mobile ad-hoc networks*, In Proceedings of the 4th IEEE International Symposium on Wireless Communication Systems (ISWCS'07), October 2007, pp. 592–596.

[35] Venkat Balakrishnan, Vijay Varadharajan, Uday Kiran Tupakula, and Marie Elizabeth Gaup Moe, *Mitigating flooding attacks in mobile ad-hoc networks supporting anonymous communications*, In Proceedings of the 2nd IEEE International Conference on Wireless Broadband and Ultra Wideband Communication (AusWireless'07), August 2007, pp. 29–34.

[36] D. Balfanz, G. Durfee, N. Shankar, D. Smetters, J. Staddon, and H.-C. Wong, *Secure handshakes from pairing-based key agreements*, In Proceedings of the IEEE Symposium on Security and Privacy, May 2003, pp. 180–196.

[37] Dirk Balfanz, D. K. Smetters, Paul Stewart, and H. Chi Wong, *Talking to strangers: Authentication in ad-hoc wireless networks*, In Proceedings of the Network and Distributed System Security Symposium (NDSS'02), February 2002.

[38] Sorav Bansal and Mary Baker, *Observation-based cooperation enforcement in ad hoc networks*, Technical Report (CS.NI/0307012), Standford University, 2003.

[39] John S. Baras and Tao Jiang, *Cooperative games, phase transitions on graphs and distributed trust in manet*, In Proceedings of the 43rd IEEE Conference on Decision and Control (CDC'04), December 2004, pp. 93–98.

[40] Stefano Basagni, Kris Herrin, Danilo Bruschi, and Emilia Rosti, *Secure pebblenets*, In Proceedings of the 2nd ACM International Symposium on Mobile Ad hoc Networking and Computing (MobiHoc'01), 2001, pp. 156–163.

[41] Marc Bechler, Hans-Joachim Hof, Daniel Kraft, Frank Pählke, and Lars C. Wolf, *A cluster-based security architecture for ad hoc networks*, In Proceedings of the

23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'04), vol. 4, March 2004, pp. 2393–2403.

[42] Elizabeth M. Belding-Royer and Charles E. Perkins, *An implementation study of the aodv routing protocol*, In Proceedings of the IEEE conference Wireless Communications and Networking Conference (WCNC'00), vol. 3, September 2000, pp. 1003–1008.

[43] ______, *Evolution and future directions of the ad hoc on-demand distance-vector routing protocol*, Ad Hoc Networks **1** (2003), no. 1, 125–150.

[44] Elizabeth M. Belding-Royer and Chai-Keong Toh, *A review of current routing protocols for ad-hoc mobile wireless networks*, IEEE Personal Communications **6** (1999), no. 2, 46–55.

[45] Thomas Beth, Malte Borcherding, and Birgit Klein, *Valuation of trust in open networks*, In Proceedings of the 3rd European Symposium on Research in Computer Security (ESORICS'94), 1994, pp. 3–18.

[46] Sonali Bhargava and Dharma P. Agrawal, *Scalable security schemes for ad hoc networks*, In Proceedings of the IEEE Military Communications Conference (MILCOM'02), vol. 2, 2002, pp. 1089–1094.

[47] Matt Blaze, Joan Feigenbaum, and Angelos D. Keromytis, *Keynote: Trust management for public-key infrastructures (position paper)*, In Proceedings of the 6th International Workshop on Security Protocols, 1998, pp. 59–63.

[48] Matt Blaze, Joan Feigenbaum, and Jack Lacy, *Decentralized trust management*, Proceedings of the IEEE Symposium on Security and Privacy, 1996, pp. 164–173.

[49] Ljubica Blazevic, *Scalable routing protocols with applications to mobility*, Ph.d. Thesis, EPFL, Lausanne, 2002.

[50] Ljubica Blazevic, Jean-Yves Le Boudec, and Silvia Giordano, *A location-based routing method for mobile ad hoc networks*, IEEE Transactions on Mobile Computing **4** (2005), no. 2, 97–110.

[51] Ljubica Blazevic, Levente Buttyán, Srdjan Čapkun, Silvia Giordano, Jean-Pierre Hubaux, and Jean-Yves Le Boudec, *Self-organization in mobile ad-hoc networks: the approach of terminodes*, IEEE Communications Magazine **39** (2001), no. 6, 166.

[52] Ljubica Blazevic, Silvia Giordano, and Jean-Yves Le Boudec, *Self organized routing in wide area mobile ad hoc network*, In Proceedings of the IEEE Symposium on Ad Hoc Wireless Networks (GLOBECOM'01), vol. 5, November 2001, pp. 2814–2818.

[53] Rakesh Babu Bobba, Laurent Eschenauer, Virgil Gligor, and William Arbaugh, *Bootstrapping security associations for routing in mobile ad-hoc networks*, In Proceedings of the Global Telecommunications Conference (GLOBECOM'03), vol. 3, December 2003, pp. 1511–1515.

[54] Muhammad J. Bohio and Ali Miri, *Authenticated secure communications in mobile ad hoc networks*, In Proceedings of the Canadian Conference on Electrical and Computer Engineering (CCECE'04), vol. 3, May 2004, pp. 1689–1692.

[55] ——, *Efficient identity-based security schemes for ad hoc network routing protocols*, Ad Hoc Networks **2** (2004), no. 3, 309–317.

[56] Dan Boneh and Matt Franklin, *Identity-based encryption from the weil pairing*, SIAM Journal on Computing **32** (2003), no. 3, 586–615.

[57] Souheila Bouam and Jalel Ben-Othman, *Securing data transmissions and retransmissions management in ad hoc networks*, In Proceedings of the International Conference on Wireless Networks, CSREA Press, 2004, pp. 144–150.

[58] Jean-Yves Le Boudec and Slavisa Sarafijanovic, *An artificial immune system approach to misbehavior detection in mobile ad hoc networks*, In Proceedings of the 1st International Workshop on Biologically Inspired Approaches to Advanced Information Technology (BioADIT'04), Lecture Notes in Computer Science, vol. 3141, 2004, pp. 396–411.

[59] Azzedine Boukerche, Khalil El-Khatib, Li Xu, and Larry Korba, *Anonymity enabling scheme for wireless ad hoc networks*, In Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM'04), December 2004, pp. 136–140.

[60] ———, *A novel solution for achieving anonymity in wireless ad hoc networks*, In Proceedings of the 1st ACM International Workshop on Performance Evaluation of Wireless Ad hoc, Sensor, and Ubiquitous Networks (PE-WASUN'04), 2004, pp. 30–38.

[61] ———, *Sdar: A secure distributed anonymous routing protocol for wireless and mobile ad hoc networks*, In Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks (LCN'04), 2004, pp. 618–624.

[62] Andre Boumso, Boucif Amar Bensaber, and Ismail Biskri, *Gakap, multicast key agreement protocol for ad hoc networks based on group activity probability*, In Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks (LCN'04), 2004, pp. 700–704.

[63] Raffaele Bruno, Marco Conti, and Enrico Gregori, *Mesh networks: Commodity multihop ad hoc networks*, IEEE Communications Magazine **43** (2005), no. 3, 123–131.

[64] Sonja Buchegger, *Coping with misbehavior in mobile ad-hoc networks*, Ph.D. Thesis, EPFL, Lausanne, April 2004.

[65] Sonja Buchegger and Jean-Yves Le Boudec, *Cooperative routing in mobile ad-hoc networks: Current efforts against malice and selfishness*, In Proceedings of the Mobile Internet Workshop (Informatik'02), October 2002.

[66] ———, *Nodes bearing grudges: Towards routing security, fairness, and robustness in mobile ad hoc networks*, In Proceedings of the 10th Conference on Euromicro Parallel, Distributed and Network-based Processing (PDP'02), January 2002, pp. 403–410.

[67] ———, *Performance analysis of the confidant protocol: Cooperation of nodes - fairness in dynamic ad-hoc networks)*, In Proceedings of the 3rd ACM International Symposium on Mobile Ad hoc Networking and Computing (MobiHoc'02), June 2002, pp. 226–236.

[68] ———, *Coping with false accusations in misbehavior reputation systems for mobile ad-hoc networks*, Technical Report (IC/2003/31), EPFL, Lausanne, May 2003.

[69] ———, *The effect of rumor spreading in reputation systems for mobile ad-hoc networks*, In Proceedings of the International Conference on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt'03), March 2003.

[70] ———, *A robust reputation system for mobile ad-hoc networks*, Technical Report (IC/2003/50), EPFL, Lausanne, July 2003.

[71] ———, *A robust reputation system for p2p and mobile ad-hoc networks*, In Proceedings of the 2nd Workshop on Economics of P2P Systems (P2PEcon'04), June 2004.

[72] ———, *Self-policing mobile ad-hoc networks by reputation*, IEEE Communication Magazine **43** (2005), no. 7, 101–107.

[73] Sonja Buchegger, Cedric Tissieres, and Jean-Yves Le Boudec, *A test-bed for misbehavior detection in mobile ad-hoc networks - how much can watchdogs really do?*, In Proceedings of the 6th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'04:), 2004, pp. 102–111.

[74] Caner Budakoglu and T. Aaron Gulliver, *Hierarchical key management for mobile ad-hoc networks*, In Proceedings of the 60th IEEE Vehicular Technology Conference (VTC'04), vol. 4, September 2004, pp. 2735–2738.

[75] Mike Burmester and Tri Van Le, *Secure communications in ad hoc networks*, In Proceedings of the 5th Annual IEEE Workshop on Information Assurance and Security, June 2004, pp. 234–241.

[76] Mike Burmester, Tri Van Le, and Alec Yasinsac, *Adaptive gossip protocols: Managing security and redundancy in dense ad hoc networks*, Journal of Ad Hoc Networks **4** (2006), no. 3, 504–515.

[77] Levente Buttyán and Jean-Pierre Hubaux, *Nuglets: a virtual currency to stimulate cooperation in self-organized mobile ad hoc networks*, Technical Report (DSC/2001/001), EPFL, Lausanne, 2001.

[78] ______, *Report on a working session on security in wireless ad hoc networks*, ACM SIGMOBILE Mobile Computing and Communications Review **7** (2003), no. 1, 74–94.

[79] ______, *Stimulating cooperation in self-organizing mobile ad hoc networks*, Mobile Networks and Applications **8** (2003), no. 5, 579–592.

[80] Jianfeng Cai and Udo Pooch, *Play alone or together - truthful and efficient routing in wireless ad hoc networks with selfish nodes*, In Proceedings of IEEE International Conference on Mobile Ad-hoc and Sensor Systems, October 2004, pp. 457–465.

[81] Stefano Campadello, *Peer-to-peer security in mobile devices: A user perspective*, In Proceedings of the 4th International Conference on Peer-to-Peer Computing (P2P'04), 2004, pp. 252–257.

[82] Catharina Candolin and Hannu H. Kari, *A security architecture for wireless ad hoc networks*, In Proceedings of the IEEE Military Communications Conference (MILCOM'02), October 2002.

[83] Licia Capra, *Reasoning about trust groups to coordinate mobile ad-hoc systems*, In Proceedings of the 1st IEEE Workshop on the Value of Security Through Collaboration, September 2005, pp. 142–152.

[84] Alvaro A. Cárdenas, Svetlana Radosavac, and John S. Baras, *Detection and prevention of mac layer misbehavior in ad hoc networks*, In Proceedings of the

2nd ACM workshop on Security of Ad hoc and Sensor Networks (SASN'04), 2004, pp. 17–22.

[85] R Castaneda and Samir R. Das, *Query localization techniques for on-demand protocols in ad hoc networks*, In Proceedings of the ACM International Conference on Mobile Computing and Networking (MOBICOM'99), 1999, pp. 186–194.

[86] Ana Cavalli and Jean-Marie Orset, *Secure hosts auto-configuration in mobile ad hoc networks*, Ad Hoc Networks **3** (2005), no. 5, 656–667.

[87] Ian D. Chakeres and Elizabeth M. Belding-Royer, *Aodv routing protocol implementation design*, In Proceedings of the 24th International Conference on Distributed Computing Systems Workshops (ICDCSW'04), 2004, pp. 698–703.

[88] Aldar C.-F. Chan, *Distributed symmetric key management for mobile ad hoc networks*, In Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'04), vol. 4, 2004, pp. 2414–2424.

[89] ______, *Probabilistic distributed key pre-distribution for mobile ad hoc networks*, In Proceedings of the IEEE International Conference on Communications, vol. 6, June 2004, pp. 3743–3747.

[90] Haowen Chan, Adrian Perrig, and Dawn Song, *Wireless sensor networks*, ch. Key Distribution Techniques for Sensor Networks, pp. 277–303, Kluwer Academic Publishers, Norwell, MA, USA, 2004.

[91] J. Chandra and L. L. Singh, *A cluster based security model for mobile ad hoc networks*, In Proceedings of the IEEE International Conference on Personal Wireless Communications (ICPWC'05), January 2005, pp. 413–416.

[92] David Chaum, *Untraceable electronic mail return address and digital pseudonyms*, ACM Communications **24** (1981), no. 2, 84–88.

[93] ______, *The dining cryptographers problem: Unconditional sender and recipient untraceability*, Journal of Cryptography **1** (1988), no. 1, 65–75.

[94] Thomas M. Chen and Varadharajan Venkataramanan, *Dempster-shafer theory for intrusion detection in ad hoc networks*, IEEE Internet Computing **9** (2005), no. 6, 35–41.

[95] Xiaohu Chen, Michalis Faloutsos, and Srikanth V. Krishnamurthy, *Power adaptive broadcasting with local information in ad hoc networks*, In Proceedings of the 11th IEEE International Conference on Network Protocols (ICNP'03), November 2003, pp. 168–178.

[96] Hung-Yu Chien and Ru-Yu Lin, *Improved id-based security framework for ad hoc network*, Ad Hoc Networks **6** (2008), no. 1, 47–60.

[97] Krishna Chintalapudi and Ramesh Govindan, *Localized edge detection in sensor fields*, Ad Hoc Networks **1** (2003), no. 2-3, 273–291.

[98] Imrich Chlamtac, Marco Conti, and Jennifer J.-N. Liu, *Mobile ad hoc networking: Imperatives and challenges*, Ad Hoc Networks **1** (2003), no. 1, 13–64.

[99] Marco Conti and Enrico Gregori, *Ad hoc networking for pervasive systems*, Ad Hoc Networks **3** (2005), no. 2, 115–117.

[100] Claude Crépeau and Carlton R. Davis, *A certificate revocation scheme for wireless ad hoc networks*, In Proceedings of the 1st ACM Workshop on Security of Ad hoc and Sensor Networks (SASN'03), 2003, pp. 54–61.

[101] Giovanni Di Crescenzo, Renwei Ge, and Gonzalo R. Arce, *Threshold cryptography in mobile ad hoc networks under minimal topology and setup assumptions.*, Ad Hoc Networks **5** (2007), no. 1, 63–75.

[102] Jon Crowcroft, Julian Chesterfield, Richard J. Gibbens, Frank P. Kelly, and Sven Östring, *Providing incentives in providerless networks*, Ad Hoc Networks **2** (2004), no. 3, 283–289.

[103] Jon Crowcroft, Richard J. Gibbens, Frank P. Kelly, and Sven Östring, *Modelling incentives for collaboration in mobile ad hoc networks*, Performance Evaluation **57** (2004), no. 4, 427–439.

[104] Benjamin J. Culpepper and H. Chris Tseng, *Sinkhole intrusion indicators in dsr manets*, In Proceedings of the 1st International Conference on Broadband Networks (BROADNETS'04), 2004, pp. 681–688.

[105] B. Dahill, K. Sanzgiri, B. N. Levine, E. M. Belding-Royer, and C. Shields, *A secure routing protocol for ad hoc networks*, In Proceedings of the 10th IEEE International Conference on Network Protocols (ICNP'02), 2002, pp. 79–89.

[106] Partha Dasgupta, *Trust: Making and breaking cooperative relations*, ch. Trust as a Commodity, pp. 49–72, Department of Sociology, University of Oxford, 2000.

[107] Carlton R. Davis, *A localized trust management scheme for ad hoc networks*, In Proceedings of the 3rd International Conference on Networking (ICN'04), 2004, pp. 671–675.

[108] Richard Dawkins, *The selfish gene*, 1989 ed., Oxford University Press, 1976.

[109] Carlos de Morais Cordeiro, Hrishikesh Gossain, and Dharma P. Agrawal, *Multicast over wireless mobile ad hoc networks: Present and future directions*, IEEE Network **17** (2003), 52–59.

[110] Hongmei Deng and Dharma P. Agrawal, *Tids: Threshold and identity-based security scheme for wireless ad hoc networks*, Ad Hoc Networks **2** (2004), no. 3, 291–307.

[111] Hongmei Deng, Wei Li, and Dharma P. Agrawal, *Routing security in wireless ad hoc networks*, IEEE Communications Magazine **40** (2002), no. 10, 70–75.

[112] Prashant Dewan and Partha Dasgupta, *Trusting routers and relays in ad hoc networks*, In Proceedings of the International Conference in Parallel Processing Workshops (ICPPW'03), 2003, pp. 351–358.

[113] Prashant Dewan, Partha Dasgupta, and Amiya Bhattacharya, *On using reputations in ad hoc networks to counter malicious nodes*, In Proceedings of the

10th International Conference on Parallel and Distributed Systems (ICPADS'04), 2004, pp. 665–673.

[114] Bhavyesh Divecha, Ajith Abraham, Crina Grosan, and Sugata Sanyal, *Impact of node mobility on manet routing protocols models*, Journal of Digital Information Management (2007), 1–6.

[115] Djamel Djenouri, Lyes Khelladi, and Nadjib Badache, *A survey of security issues in mobile ad hoc and sensor networks*, IEEE Communications Surveys and Tutorials **7** (2005), no. 1-4, 2–28.

[116] Florian Dotzer, Lars Fischer, and Przemyslaw Magiera, *Vars: A vehicle ad-hoc network reputation system*, In Proceedings of the International Symposium on a World of Wireless, Mobile and Multimedia Networks, vol. 1, 2005, pp. 454–456.

[117] Olivier Dousse, François Baccelli, and Patrick Thiran, *Impact of interferences on connectivity in ad hoc networks*, IEEE/ACM Transactions on Networking **13** (2005), no. 2, 425–436.

[118] Xinjun Du, Ying Wang, Jianhua Ge, and Yumin Wang, *A group key establishment scheme for ad hoc networks*, In Proceedings of the17th International Conference on Advanced Information Networking and Applications (AINA'03), 2003, pp. 518–520.

[119] ———, *A method for security enhancements in aodv protocol*, In Proceedings of the 17th International Conference on Advanced Information Networking and Applications (AINA'03), 2003, pp. 237–240.

[120] Stephan Eichler, Florian Dotzer, Francisco Javier Fabra Caro Schwingenschlogl, and Jorg Eberspaher, *Secure routing in a vehicular ad hoc network*, In Proceedings of the 60th IEEE Vehicular Technology Conference (VTC'04-Fall), vol. 5, 2004, pp. 3339–3343.

[121] Thomas Engel, Daniel Fischer, Thomas Scherer, and Dagmara Spiewak, *A survey on security challenges in next generation mobile networks*, In Proceedings of the

3rd International Conference on Mobile Computing and Ubiquitous Networking (ICMU'06), October 2006, pp. 116–125.

[122] Ozkan M. Erdem, *Edkm: Efficient distributed key management for mobile ad hoc networks*, In Proceedings of the 9th International Symposium on Computers and Communications (ISCC'04), vol. 2, 2004, pp. 325–330.

[123] ______, *Efficient self-organized key management for mobile ad hoc networks*, In Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM'04), vol. 4, December 2004, pp. 2185–2190.

[124] L. Ertaul and N. Chavan, *Security of ad hoc networks and threshold cryptography*, In Proceedings of the International Conference on Wireless Networks, Communications, and Mobile Computing (Wirelesscom'05), vol. 1, June 2005, pp. 69–74.

[125] Laurent Eschenauer, Virgil D. Gligor, and John S. Baras, *On trust establishment in mobile ad-hoc networks.*, In Proceedings of the Security Protocols Workshop, Lecture Notes in Computer Science, vol. 2845, 2002, pp. 47–66.

[126] Laura Marie Feeney and Martin Nilsson, *Investigating the energy consumption of a wireless network interface in an ad hoc networking environment*, In Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'01), vol. 3, August 2001, pp. 1548–1557.

[127] Michal Feldman, John Chuang, Ion Stoica, and Scott Shenker, *Hidden-action in multi-hop routing*, In Proceedings of the 6th ACM conference on Electronic Commerce (EC'05), 2005, pp. 117–126.

[128] Raihana Ferdous, Vallipuram Mutthukkumarasamy, and Abdul Sattar, *Trust formalization in mobile ad-hoc networks*, In Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applicaitons Workshops (WAINA'10), April 2010, pp. 351–356.

[129] Alberto Fernandes, Evangelos Kotsovinos, Sven Östring, and Boris Dragovic, *Pinocchio: Incentives for honest participation in distributed trust management*,

In Proceedings of the 2nd International Conference on Trust Management (iTrust'04), Lecture Notes in Computer Science, vol. 2995, March 2004, pp. 63–77.

[130] Klas Fokine, *Key management in ad hoc networks*, Master's Thesis, Department of Electrical Engineering, Linköping University, 2002.

[131] Python Software Foundation, *Python programming language*, Url (http://www.python.org/), Last checked on 20 May, 2010.

[132] Zhenghua Fu, Benjamin Greenstein, Xiaoqiao Meng, and Songwu Lu, *Design and implementation of a tcp-friendly transport protocol for ad hoc wireless networks*, In Proceedings of the 10th IEEE International Conference on Network Protocols (ICNP'02), 2002, pp. 216–225.

[133] Diego Gambetta, *Trust: Making and breaking cooperative relations*, no. 13, ch. Can We Trust Trust?, pp. 213–237, Basil Blackwell, 1988.

[134] Saurabh Ganeriwal, Laura K. Balzano, and Mani B. Srivastava, *Reputation-based framework for high integrity sensor networks*, ACM Transactions on Sensor Networks **4** (2008), no. 3, 1–37.

[135] Xianjun Geng, Yun Huang, and Andrew B. Whinston, *Defending wireless infrastructure against the challenge of ddos attacks*, Mobile Networks and Applications **7** (2002), no. 3, 213–223.

[136] S. Ghazizadeh, O. Ilghami, E. Sirin, and F. Yaman, *Security-aware adaptive dynamic source routing protocol*, In Proceedings of the 27th Annual IEEE Conference on Local Computer Networks (LCN'02), 2002, p. 0751.

[137] Tirthankar Ghosh, Niki Pissinou, and Kami Makki, *Towards designing a trusted routing solution in mobile ad hoc networks*, Mobile Networks and Applications **10** (2005), no. 6, 985–995.

[138] Paolo Giorgini, Fabio Massacci, John Mylopoulos, and Nicola Zannone, *Requirements engineering meets trust management - model, methodology, and reasoning*, In Proceedings of the 2nd International Conference on Trust Management (iTrust'04), Lecture Notes in Computer Science, vol. 2995, March 2004, pp. 176–190.

[139] Javier Gomez, Andrew T. Campbell, Mahmoud Naghshineh, and Chatschik Bisdikian, *Conserving transmission power in wireless ad hoc networks*, In Proceedings of the 9th International Conference on Network Protocols (ICNP'01), 2001, pp. 24–34.

[140] Swee Keow Goo, James M. Irvine, Robert C. Atkinson, and John Dunlop, *Trust architecture for a personal distributed environment*, In Proceedings of the IEEE Vehicular Technology Conference (VTC'04), 2004, pp. 4908–4912.

[141] Tyrone Grandison and Morris Sloman, *Specifying and analysing trust for internet applications*, In Proceedings of the 2nd IFIP Conference on Towards The Knowledge Society: E-Commerce, E-Business, E-Government (I3E'02), vol. 233, 2002, pp. 145–157.

[142] ______, *Trust management tools for internet applications*, In Proceedings of the 1st International Conference on Trust Management, Lecture Notes in Computer Science, vol. 2692, May 2003, pp. 91–107.

[143] Qijun Gu, Peng Liu, and Chao-Hsien Chu, *Analysis of area-congestion-based ddos attacks in ad hoc networks*, Ad Hoc Networks **5** (2007), no. 5, 613–625.

[144] R. Guha, Ravi Kumar, Prabhakar Raghavan, and Andrew Tomkins, *Propagation of trust and distrust*, In Proceedings of the 13th International Conference on World Wide Web (WWW'04), 2004, pp. 403–412.

[145] Vikram Gupta, Srikanth Krishnamurthy, and Michalis Faloutsos, *Denial of service attacks at the mac layer in wireless ad hoc networks*, In Proceedings of the Military Communications Conference (MILCOM'02), 2002, pp. 1118–1123.

[146] Siddhartha Gupte and Mukesh Singhal, *Secure routing in mobile wireless ad hoc networks*, Ad Hoc Networks **1** (2003), no. 1, 151–174.

[147] George C. Hadjichristofi, William Joseph Adams, and Nathaniel J. Davis IV, *A framework for key management in mobile ad hoc networks.*, In Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05), 2005, pp. 568–573.

[148] Rolf Haenni, Jacek Jonczy, and Reto Kohlas, *Two-layer models for managing authenticity and trust*, In Proceedings of the International Conference on Trust in E-Services: Technologies, Practices and Challenges, 2006, pp. 140–167.

[149] Charles B. Haley, Robin C. Laney, Jonathan D. Moffett, and Bashar Nuseibeh, *Picking battles: The impact of trust assumptions on the elaboration of security requirements*, In Proceedings of the 2nd International Conference on Trust Management (iTrust'04), Lecture Notes in Computer Science, vol. 2995, March 2004, pp. 347–354.

[150] Lu Han, Dongming Zhao, and Manli Zhou, *A network layer security mechanism based on collaborative intelligent agents in manet*, In Proceedings of the 3rd International Conference on Information Technology: Research and Education (ITRE'05), June 2005, pp. 56–59.

[151] Qi He, Dapeng Wu, and Pradeep Khosla, *A secure incentive architecture for ad-hoc networks: Research articles*, Wireless Communications and Mobile Computing **6** (2006), no. 3, 333–346.

[152] Anne Marie Hegland, Eli Winjum, Stig Fr. Mjølsnes, Chunming Rong, Øivind Kure, and Pål Spilling, *A survey of key management in ad hoc networks.*, IEEE Communications Surveys and Tutorials **8** (2006), no. 1-4, 48–66.

[153] Abdulrahman Hijazi and Nidal Nasser, *Using mobile agents for intrusion detection in wireless ad hoc networks*, In Proceedings of the 2nd IFIP International

Conference on Wireless and Optical Communications Networks (WOCN'05), March 2005, pp. 362–366.

[154] Fan Hong, Liang Hong, and Cai Fu, *Secure olsr*, In Proceedings of the 19th International Conference on Advanced Information Networking and Applications (AINA'05), 2005, pp. 713–718.

[155] Yih-Chun Hu, *Enabling secure high-performance wireless ad hoc networking*, Ph.d. Thesis (CMU-CS-03-144), School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA, May 2003.

[156] Yih-Chun Hu and David B. Johnson, *Implicit source routes for on-demand ad hoc network routing*, In Proceedings of the 2nd ACM International Symposium on Mobile Ad hoc Networking and Computing (MobiHoc'01), October 2001, pp. 1–10.

[157] ———, *Exploiting congestion information in network and higher layer protocols in multihop wireless ad hoc networks*, In Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04), 2004, pp. 301–310.

[158] ———, *Securing quality-of-service route discovery in on-demand routing for ad hoc networks*, In Proceedings of the 2nd ACM Workshop on Security of Ad hoc and Sensor Networks (SASN'04), 2004, pp. 106–117.

[159] Yih-Chun Hu, David B. Johnson, and Adrian Perrig, *Sead: Secure efficient distance vector routing for mobile wireless ad hoc networks*, In Proceedings of the 4th IEEE Workshop on Mobile Computing Systems and Applications, vol. 1, June 2002, pp. 3–13.

[160] Yih-Chun Hu and Adrian Perrig, *A survey of secure wireless ad hoc routing*, IEEE Security and Privacy **2** (2004), no. 3, 28–39.

[161] Yih-Chun Hu, Adrian Perrig, and David B. Johnson, *Efficient security mechanisms for routing protocols*, In Proceedings of the 10th Annual Network and Distributed System Security Symposium (NDSS'03), 2003, pp. 57–73.

[162] ______, *Packet leashes: A defense against wormhole attacks in wireless networks*, In Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications (INFOCOM'03), vol. 3, April 2003, pp. 1976–1986.

[163] ______, *Rushing attacks and defense in wireless ad hoc network routing protocols*, In Proceedings of the 2nd ACM workshop on Wireless Security (WiSe'03), 2003, pp. 30–40.

[164] ______, *Ariadne: A secure on-demand routing protocol for ad hoc networks*, Wireless Networks **11** (2005), no. 1-2, 21–38.

[165] Elgan Huang, Jon Crowcroft, and Ian Wassell, *Rethinking incentives for mobile ad hoc networks*, In Proceedings of the ACM SIGCOMM Workshop on Practice and Theory of Incentives in Networked Systems (PINS'04), 2004, pp. 191–196.

[166] He Huang and Shyhtsun Felix Wu, *An approach to certificate path discovery in mobile ad hoc networks*, In Proceedings of the 1st ACM workshop on Security of Ad hoc and Sensor Networks (SASN '03), 2003, pp. 41–52.

[167] Jean-Pierre Hubaux, Levente Buttyán, and Srdan Capkun, *The quest for security in mobile ad hoc networks*, In Proceedings of the 2nd ACM International Symposium on Mobile Ad hoc Networking and Computing (MobiHoc'01), 2001, pp. 146–155.

[168] Jean-Pierre Hubaux, Thomas Gross, Jean-Yves Le Boudec, and Martin Vetterli, *Towards self-organized mobile ad hoc networks: The terminodes project*, IEEE Communications Magazine **39** (2001), no. 1, 118–124.

[169] Todd Hughes, James Denny, and P. Andy Muckelbauer, *Dynamic trust applied to ad hoc network resources*, In Proceedings of the Autonomous Agents and Multi-Agent Systems Conference (AAMS'03), July 2003.

[170] IBM, *Rational clearquest*, Url (http://www-01.ibm.com/software/awdtools/clearquest/), Last checked on 20 May, 2010.

[171] ———, *Rational rose enterprise*, Url (http://www-01.ibm.com/software/awdtools/developer/rose/enterprise/), Last checked on 20 May, 2010.

[172] Omer Ileri, Siun-Chuon Mau, and Narayan B. Mandayam, *Pricing for enabling forwarding in self-configuring ad hoc networks.*, IEEE Journal on Selected Areas in Communications **23** (2005), no. 1, 151–162.

[173] Mohammad Ilyas (ed.), *The handbook of ad hoc wireless networks*, CRC Press, Inc., Boca Raton, FL, USA, 2003.

[174] Kai Inkinen, *New secure routing in ad hoc networks: Study and evaluation of proposed schemes*, Technical Report (HUT T-110.551), Helsinki University of Technology, Finland, 2004.

[175] Faiza Younas Janjua, Samia Sultan, Mariam Muzaffar, Zaheer Ahmed, and Shoab A Khan, *Authenticated routing of table driven protocol in an ad-hoc environment*, In Proceedings of the IEEE Networking and Communication Conference (INCC'04), June 2004, pp. 6–12.

[176] Shu Jiang, *A mix route algorithm for mix-net in wireless ad hoc networks*, In Proceedings of the IEEE Mobile Sensor and Ad-hoc and Sensor Systems (MASS'04), October 2004, pp. 406–415.

[177] Shu Jiang, Nitin H. Vaidya, and Wei Zhao, *A dynamic mix method for wireless ad hoc networks*, In Proceedings of the IEEE Military Communications Conference (MILCOM'01), 2001, pp. 873–877.

[178] Tao Jiang and John S. Baras, *Ant-based adaptive trust evidence distribution in manet*, In Proceedings of the 24th International Conference on Distributed Computing Systems Workshops (ICDCSW'04), March 2004, pp. 588–593.

[179] Tingyao Jiang and Qinghua Li, *A secure routing protocol for mobile ad hoc networks*, In Proceedings of the 3rd International Conference on Machine Learning and Cybernetics, August 2004, pp. 2825–2829.

[180] Tingyao Jiang, Qinghua Li, and Youlin Ruan, *Secure dynamic source routing protocol*, In Proceedings of the 4th International Conference on Computer and Information Technology (CIT'04), 2004, pp. 528–533.

[181] David B. Johnson and David Maltz, *Dynamic source routing in ad hoc wireless networks*, Mobile Computing (1996), 153–181.

[182] Catholijn M. Jonker, Joost J. P. Schalken, Jan Theeuwes, and J. Treur, *Human experiments in trust dynamics*, In Proceedings of the 2nd International Conference (iTrust'04), 2004, pp. 206–220.

[183] Audun Jøsang, *The right type of trust for distributed systems*, In Proceedings of the Workshop on New Security Paradigms (NSPW'96), 1996, pp. 119–131.

[184] ———, *Subjective evidential reasoning*, In Proceedings of the International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU'02), July 2002.

[185] ———, *Probabilistic logic under uncertainty*, In Proceedings of the 13th Australasian Symposium on Theory of Computing (CATS'07), 2007, pp. 101–110.

[186] Audun Jøsang, Roslan Ismail, and Colin Boyd, *A survey of trust and reputation systems for online service provision*, Decision Support Systems **43** (2007), no. 2, 618–644.

[187] Audun Jøsang, Claudia Keser, and Theo Dimitrakos, *Can we manage trust?*, In Proceedings of the 3rd International Conference on Trust Management (iTrust'05), Lecture Notes in Computer Science, vol. 3477, 2005, pp. 93–107.

[188] Audun Jøsang and Stephane Lo Presti, *Analysing the relationship between risk and trust*, In Proceedings of the 2nd International Conference on Trust Management (iTrust'04), Lecture Notes in Computer Science, vol. 2995, March 2004, pp. 135–145.

[189] Mike Just, Evangelos Kranakis, and Tao Wan, *Resisting malicious packet dropping in wireless ad hoc networks*, In Proceedings of the 2nd International Conference on Ad-Hoc, Mobile, and Wireless Networks (ADHOC-NOW'03), Lecture Notes in Computer Science, vol. 2865, 2003, pp. 151–163.

[190] Kevin Kane and James C. Browne, *Using uncertainty in reputation methods to enforce cooperation in ad-hoc networks*, In Proceedings of the 5th ACM Workshop on Wireless Security (WiSe'06), 2006, pp. 105–113.

[191] Frank Kargl, Andreas Klenk, Stefan Schlott, and Michael Weber, *Advanced detection of selfish or malicious nodes in ad hoc networks*, In Proceedings of the 1st European Workshop on Security in Ad-Hoc and Sensor Networks (ESAS'04), Lecture Notes in Computer Science, vol. 3313, 2004, pp. 152–165.

[192] Frank Kargl, Stefan Schlott, Andreas Klenk, Alfred Geiss, and Michael Weber, *Securing ad hoc routing protocols*, In Proceedings of the 30th EUROMICRO Conference, 1089-6503, September 2004, pp. 514–519.

[193] Chris Karlof and David Wagner, *Secure routing in wireless sensor networks: Attacks and countermeasures*, In Proceedings of the 1st IEEE International Workshop on Sensor Network Protocols and Applications, 2002, pp. 113–127.

[194] T. Kaya, G. Lin, G. Noubir, and A. Yilmaz, *Secure multicast groups on ad hoc networks*, In Proceedings of the 1st ACM Workshop on Security of Ad hoc and Sensor Networks (SASN'03), 2003, pp. 94–102.

[195] Issa Khalil, Saurabh Bagchi, and Cristina Nina-Rotaru, *Dicas: Detection, diagnosis and isolation of control attacks in sensor networks*, In Proceedings of the International Conference on Security and Privacy for Emerging Areas in Communications Networks, 2005, pp. 89–100.

[196] Aram Khalili, Jonathan Katz, and William A. Arbaugh, *Toward secure key distribution in truly ad-hoc networks*, In Proceedings of the 2003 Symposium on Applications and the Internet Workshops (SAINT-W'03), 2003, pp. 342–346.

[197] Dongkyun Kim, J. J. Garcia-Luna-Aceves, Katia Obraczka, Juan-Carlos Cano, and Pietro Manzoni, *Routing mechanisms for mobile ad hoc networks based on the energy drain rate*, IEEE Transactions on Mobile Computing **2** (2003), no. 2, 161–173.

[198] Young-Bae Ko and Nitin H. Vaidya, *Geotora: A protocol for geocasting in mobile ad hoc networks*, In Proceedings of the 8th International Conference on Network Protocols (ICNP'00), 2000, pp. 240–250.

[199] Jiejun Kong, *Anonymous and untraceable communications in mobile wireless networks*, Ph.d. Thesis, University of California, Los Angeles, 2004, Chair-Gerla, Mario.

[200] Jiejun Kong and Xiaoyan Hong, *Anodr: Anonymous on demand routing with untraceable routes for mobile ad-hoc networks*, In Proceedings of the 4th ACM International Symposium on Mobile Ad hoc Networking and Computing (MobiHoc'03), June 2003, pp. 291–302.

[201] Jiejun Kong, Xiaoyan Hong, and Mario Gerla, *A new set of passive routing attacks in mobile ad hoc networks*, In Proceedings of the IEEE Military Communications Conference (MILCOM'03, January 2003, pp. 796–801.

[202] Jiejun Kong, Haiyun Luo, Kaixin Xu, Daniel Lihui Gu, Mario Gerla, and Songwu Lu, *Adaptive security for multilevel ad hoc networks*, Wireless Communications and Mobile Computing **2** (2002), no. 5, 533–547.

[203] Jiejun Kong, Petros Zerfos, Haiyun Luo, Songwu Lu, and Lixia Zhang, *Providing robust and ubiquitous security support for mobile ad hoc networks*, In Proceedings of the 9th International Conference on Network Protocols (ICNP'01), 2001, pp. 251–260.

[204] Daniel Kraft and Günter Schäfer, *Distributed access control for consumer operated mobile ad-hoc networks*, In Proceedings of the 1st IEEE Consumer Communication and Networking Conference (CCNC'04), January 2004, pp. 35–40.

[205] Marwan Krunz, Alaa Muqattash, , and Sung-Ju Lee, *Transmission power control in wireless ad hoc networks: Challenges, solutions and open issues*, IEEE Network **18** (2004), no. 5, 8–14.

[206] Sunil Kumar, Vineet S. Raghavan, and Jing Deng, *Medium access control protocols for ad hoc wireless networks: A survey*, Ad Hoc Networks **4** (2006), no. 3, 326–358.

[207] Pradeep Kyasanur, *Selfish misbehavior at medium access control layer in wireless networks*, Master's Thesis, University of Illinois, Urbana-Champaign, December 2003.

[208] Pradeep Kyasanur and Nitin H. Vaidya, *Selfish mac layer misbehavior in wireless networks*, IEEE Transactions on Mobile Computing **4** (2005), no. 5, 502–516.

[209] Bernd Lamparter, Krishna Paul, and Dirk Westhoff, *Charging support for ad hoc stub networks*, Computer Communications **26** (2003), no. 13, 1504–1514.

[210] Bernd Lamparter, Marc Plaggemeier, and Dirk Westhoff, *Estimating the value of co-operation approaches for multi-hop ad hoc networks.*, Ad Hoc Networks **3** (2005), no. 1, 17–26.

[211] Pradip Lamsal, *An overview of modeling trust*, In Proceedings of the International Conference on Ubiquitous Computing and Ad Hoc Networks, April 2002.

[212] ______, *Requirements for modeling trust in ubiquitous computing and ad hoc networks*, Technical Report (HUT TML/Course T-110.557), Helsinki University of Technology, 2002.

[213] Hu Imm Lee, *Afora: Ad hoc routing in the face of misbehaving nodes*, Master's Thesis, Massachusetts Institute of Technology, September 2002.

[214] Seungjoon Lee, Bohyung Han, and Minho Shin, *Robust routing in wireless ad hoc networks*, In Proceedings of the International Conference on Parallel Processing Workshops (ICPPW'02), August 2002, pp. 73–78.

[215] Youn-Ho Lee, Heeyoul Kim, Byungchun Chung, Jaewon Lee, and Hyunsoo Yoon, *On-demand secure routing protocol for ad hoc network using id based cryptosystem*, In Proceedings of the 4th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'03), August 2003, pp. 211–215.

[216] Ruidong Li, Jie Li, Hisao Kameda, and Peng Liu, *Localized public-key management for mobile ad hoc networks*, In Proceedings of IEEE Global Telecommunications Conference (GLOBECOM'04), 2004, pp. 1284–1289.

[217] Ruidong Li, Jie Li, Peng Liu, and Jien Kato, *A novel hybrid trust management framework for manets*, In Proceedings of the 29th IEEE International Conference on Distributed Computing Systems (ICDCS'09), June 2009, pp. 251–256.

[218] Sheng-Ti Li and Xiong Wang, *Ad hoc network security with geographical aids*, In Proceedings of the IEEE International Conference on Networking, Sensing and Control, vol. 1, March 2004, pp. 474–479.

[219] Victor O. K. Li and Zhenxin Lu, *Ad hoc network routing*, In Proceedings of the IEEE International Conference on Networking, Sensing and Control, March 2004, pp. 100–105.

[220] Xiaoqi Li, *Trust model based self-organized routing protocol for secure ad hoc networks*, Technical report, Department of Computer Science and Engineering, The Chinese University of Hong Kong, April 2003.

[221] Xiaoqi Li, Michael R. Lyu, and Jiangchuan Liu, *Taodv: A trusted aodv routing protocol for mobile ad hoc networks*, Technical report, Department of Computer Science and Engineering, The Chinese University of Hong Kong, 2003.

[222] ———, *A trust model based routing protocol for secure ad hoc networks*, In Proceedings of the IEEE Aerospace Conference, vol. 2, March 2004, pp. 1286–1295.

[223] Yan Li, Zhifeng Zhao, Hai Wang, Shilei Shao, and Shaoren Zheng, *An estimation based adaptive fairness algorithm for ad hoc networks*, In Proceedings of the 17th

International Conference on Advanced Information Networking and Applications (AINA'03), March 2003, pp. 324–329.

[224] Zhenjiang Li and J. J. Garcia-Luna-Aceves, *Non-interactive key establishment in mobile ad hoc networks*, Ad Hoc Networks **5** (2007), no. 7, 1194–1203.

[225] Jörg Liebeherr and Guangyu Dong, *An overlay approach to data security in ad-hoc networks.*, Ad Hoc Networks **5** (2007), no. 7, 1055–1072.

[226] Ching Lin and Vijay Varadharajan, *Modelling and evaluating trust relationships in mobile agents based systems*, In Proceedings of the International Conference on Applied Cryptography and Network Security (ACNS'03), Lecture Notes in Computer Science, vol. 2846, 2003, pp. 176–190.

[227] Ching Lin, Vijay Varadharajan, Yan Wang, and Yi Mu, *On the design of a new trust model for mobile agent security*, In Proceedings of the 1st International Conference on Trust and Privacy in Digital Business (TrustBus'04), Lecture Notes in Computer Science, vol. 3184, November 2004, pp. 60–69.

[228] Ching Lin, Vijay Varadharajan, Yan Wang, and Vineet Pruthi, *Enhancing grid security with trust management*, In Proceedings of the IEEE International Conference on Services Computing (SCC'04), 2004, pp. 303–310.

[229] Justin Lipman, Mehran Abolhasan, Paul Boustead, and Joe Chicharo, *An optimised resource aware approach to information collection in ad hoc networks*, Ad Hoc Networks **3** (2005), no. 5, 643–655.

[230] Donggang Liu and Peng Ning, *Multilevel μtesla: Broadcast authentication for distributed sensor networks*, ACM Transactions in Embedded Computing Systems **3** (2004), no. 4, 800–836.

[231] Donggang Liu, Peng Ning, and Kun Sun, *Efficient self-healing group key distribution with revocation capability*, In Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS'03), 2003, pp. 231–240.

[232] Huey-Ing Liu and Yiyung Li, *A location based qos routing protocol for ad hoc networks*, In Proceedings of the 17th International Conference on Advanced Information Networking and Applications (AINA'03), 2003, pp. 830–833.

[233] Jinshan Liu and Valérie Issarny, *Enhanced reputation mechanism for mobile ad hoc networks*, In Proceedings of the 2nd International Conference on Trust Management (iTrust'04), Lecture Notes in Computer Science, vol. 2995, 2004, pp. 48–62.

[234] Yanbin Liu and Yang Richard Yang, *Reputation propagation and agreement in mobile ad-hoc networks*, In Proceedings of the IEEE Wireless Communications and Networking (WCNC'03), vol. 3, March 2003, pp. 1510–1515.

[235] Zhaoyu Liu, Anthony W. Joy, and Robert A. Thompson, *A dynamic trust model for mobile ad hoc networks*, In Proceedings of the 10th IEEE International Workshop on Future Trends of Distributed Computing Systems (FTDCS'04), 2004, pp. 80–85.

[236] Wenjing Lou, Wei Liu, and Yuguang Fang, *Spread: Enhancing data confidentiality in mobile ad hoc networks*, In Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications (INFOCOM'04), March 2004, pp. 2404–2413.

[237] Bin Lu and Udo W. Pooch, *Cooperative security-enforcement routing in mobile ad hoc networks*, In Proceedings of the 4th International Workshop on Mobile and Wireless Communications Network (MWCN'02), 2002, pp. 157–161.

[238] ———, *A lightweight authentication protocol for mobile ad hoc networks*, In Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05), vol. 2, 2005, pp. 546–551.

[239] Haiyun Luo, Jiejun Kong, Petros Zerfos, Songwu Lu, and Lixia Zhang, *Ursa: Ubiquitous and robust access control for mobile ad-hoc networks*, IEEE/ACM Transactions on Networking **12** (2004), no. 6, 1049–1063.

[240] Haiyun Luo, Petros Zerfos, Jiejun Kong, Songwu Lu, and Lixia Zhang, *Self-securing ad hoc wireless networks*, In Proceedings of the 7th International Symposium on Computers and Communications (ISCC'02), 2002, pp. 567–574.

[241] Abdalla Mahmoud, Ahmed Sameh, and Sherif El-Kassas, *Reputed authenticated routing for ad hoc networks protocol (reputed-aran)*, In Proceedings of the 2nd ACM International Workshop on Performance Evaluation of Wireless Ad hoc, Sensor, and Ubiquitous Networks (PE-WASUN'05), October 2005, pp. 258–259.

[242] S. Kami Makki, Peter Reiher, Kia Makki, Niki Pissinou, and Shamila Makki (eds.), *Mobile and wireless network security and privacy*, Springer, 2007.

[243] Source Configuration Management, *Concurrent versions system (cvs)*, Url (http://www.nongnu.org/cvs/), Last checked on 20 May, 2010.

[244] Mahesh K. Marina and Samir R. Das, *On-demand multipath distance vector routing in ad hoc networks*, In Proceedings of the 9th International Conference on Network Protocols, 2001, pp. 14–23.

[245] Stephen P. Marsh, *Formalising trust as a computational concept*, Ph.d. Thesis, University of Stirling, 1994.

[246] John Marshall, Vikram Thakur, and Alec Yasinsac, *Identifying flaws in the secure routing protocol*, In Proceedings of the IEEE International Performance Computing, and Communications Conference, 2003, pp. 167–174.

[247] Sergio Marti, T. J. Giuli, Kevin Lai, and Mary Baker, *Mitigating routing misbehavior in mobile ad hoc networks*, In Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (MobiCom'00), 2000, pp. 255–265.

[248] Fabio Martinelli, Marinella Petrocchi, and Anna Vaccarelli, *Local management of credits and debits in mobile ad hoc networks*, In Proceedings of the 8th IFIP Conference on Communications and Multimedia Security (CMS'04), vol. 175, 2005, pp. 31–45.

[249] MathWorks, *Matlab*, Url (http://www.mathworks.com/), Last checked on 20 May, 2010.

[250] Rosa Mavropodi, Panayiotis Kotzanikolaou, and Christos Douligeris, *Secmr - a secure multipath routing protocol for ad hoc networks.*, Ad Hoc Networks **5** (2007), no. 1, 87–99.

[251] Sirisha R. Medidi, Muralidhar Medidi, and Sireesh Gavini, *Detecting packet-dropping faults in mobile ad-hoc networks*, In Proceedings of the 37th Asilomar Conference on Signals, Systems and Computers, November 2003, pp. 1708–1712.

[252] Kamal Deep Meka, Mohit Virendra, and Shambhu Upadhyaya, *Trust based routing decisions in mobile ad-hoc networks*, In Proceedings of the Workshop on Secure Knowledge Management (SKM'06), 2006.

[253] Thomas S. Messerges, Johnas Cukier, Tom A. M. Kevenaar, Larry Puhl, René Struik, and Ed Callaway, *A security design for a general purpose, self-organizing, multihop ad hoc wireless network*, In Proceedings of the 1st ACM Workshop on Security of Ad hoc and Sensor Networks (SASN'03), 2003, pp. 1–11.

[254] Pietro Michiardi and Refik Molva, *Simulation-based analysis of security exposures in mobile ad hoc networks*, In Proceedings of the European Wireless Conference in Mobile Ad Hoc Networks, February 2002.

[255] ______, *A game theoretical approach to evaluate cooperation enforcement mechanisms in mobile ad hoc networks*, In Proceedings of the Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt'03), March 2003, pp. 3–6 (Anglais).

[256] ______, *Mobile ad hoc networking*, no. 12, ch. Ad Hoc Networks Security, pp. 329–354, Wiley InterScience, 2004.

[257] ______, *Analysis of coalition formation and cooperation strategies in mobile ad hoc networks*, Ad Hoc Networks **3** (2005), no. 2, 193–219.

[258] Nikola Milanovic, Miroslaw Malek, Anthony Davidson, and Veljko Milutinovic, *Routing and security in mobile ad hoc networks*, Computer **37** (2004), 69–73.

[259] Kevin L. Mills, *A brief survey of self-organization in wireless sensor networks*, Wireless Communications and Mobile Computing **7** (2007), no. 7, 823–834.

[260] R. N. Mir and A. M. Wani, *Secure distributed routing in ad hoc networks*, In Proceedings of the Conference on Convergent Technologies for Asia-Pacific Region (TENCON 2003), vol. 3, October 2003, pp. 1091–1095.

[261] Hugo Miranda and Luís Rodrigues, *Friends and foes: Preventing selfishness in open mobile ad hoc networks*, In Proceedings of the 23rd International Conference on Distributed Computing Systems (ICDCSW'03), 2003, pp. 440–445.

[262] Amitabh Mishra, Ketan Nadkarni, and Animesh Patcha, *Intrusion detection in wireless ad hoc networks*, IEEE Wireless Communications **11** (2004), 48–60.

[263] Prasant Mohapatra, Chao Gui, and Jian Li, *Group communications in mobile ad hoc networks*, Computer **37** (2004), no. 2, 52–59.

[264] Prasant Mohapatra, Jian Li, and Chao Gui, *Qos in mobile ad hoc networks*, IEEE Wireless Communications (2003), 44–52.

[265] Refik Molva and Pietro Michiardi, *Security in ad hoc networks*, In Proceedings of the Personal Wireless Communications, September 2003.

[266] L. Mui, M. Mohtashemi, and A. Halberstadt, *A computational model of trust and reputation for e-businesses*, In Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS'02), vol. 7, 2002, p. 188.

[267] Amitava Mukherjee, Somprakash Bandyopadhyay, and Debashis Saha, *Location management and routing in mobile wireless networks*, Artech House, Inc., Norwood, MA, USA, 2001.

[268] Anindo Mukherjee, Anurag Gupta, and Dharma P. Agrawal, *Totally distributed key management for dynamic groups in manets*, In Proceedings of the 24th

IEEE International Performance Computing and Communications Conference (IPCCC'05), April 2005, pp. 185–192.

[269] Anindo Mukherjee, Meetu Gupta, Hongmei Deng, and Dharma P. Agrawal, *Level-based key establishment for multicast communication*, In Proceedings of the IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC'04), 2004, pp. 2871–2875.

[270] Jochen Mundinger and Jean-Yves Le Boudec, *Analysis of a robust reputation system for self-organized networks*, European Transactions on Communication **16** (2005), no. 5, 375–384.

[271] ———, *Analysis of a reputation system for mobile ad-hoc networks with liars*, Performance Evaluation **65** (2008), no. 3-4, 212–226.

[272] Edith C. H. Ngai and Michael R. Lyu, *Trust- and clustering-based authentication services in mobile ad hoc networks*, In Proceedings of the 24th International Conference on Distributed Computing Systems Workshops (ICDCSW'04), 2004, pp. 582–587.

[273] Edith C. H. Ngai, Michael R. Lyu Rol, and Roland T. Chin, *An authentication service against dishonest users in mobile ad hoc networks*, In Proceedings of the IEEE Aerospace Conference, 2004, pp. 1275–1285.

[274] Hoang Lan Nguyen and Uyen Trang Nguyen, *A study of different types of attacks on multicast in mobile ad hoc networks.*, Ad Hoc Networks **6** (2008), no. 1, 32–46.

[275] Sze-Yao Ni, Yu-Chee Tseng, Yuh-Shyan Chen, and Jang-Ping Sheu, *The broadcast storm problem in mobile ad hoc networks*, In Proceedings of the ACM International Conference on Mobile Computing and Networking (MOBICOM'99), 1999, pp. 151–162.

[276] Dragos Niculescu and Badri Nath, *Localized positioning in ad hoc networks*, Ad Hoc Networks **1** (2003), no. 2-3, 247–259.

[277] Peng Ning and Kun Sun, *How to misuse aodv: A case study of insider attacks against mobile ad-hoc routing protocols*, Ad Hoc Networks **3** (2005), no. 6, 795–819.

[278] Philipp Obreiter, Birgitta König-Ries, and Georgios Papadopoulos, *Engineering incentive schemes for ad hoc networks - a case study for the lanes overlay*, In Proceedings of the 1st International Workshop on Pervasive Information Management (EDBT-Workshop), Lecture Notes in Computer Science, vol. 3268, 2004, pp. 395–404.

[279] The Institute of Electrical and Electronics Engineers, *Wireless lan medium access control (mac) and physical layer (phy) specifications, ieee std 802.11*, Standards, IEEE Computer Society LAN MAN Standards Committee, 1997.

[280] Opnet, *Network engineering, operations, and planning*, Url (http://www.opnet.com/solutions/network_planning_operations/), Last checked on 20 May, 2010.

[281] Santashil Palchaudhuri and David B. Johnson, *Power mode scheduling for ad hoc networks*, In Proceedings of the 10th IEEE International Conference on Network Protocols (ICNP'02), 2002, pp. 192–193.

[282] Panagiotis Papadimitratos and Zygmunt J. Haas, *Secure routing for mobile ad hoc networks*, In Proceedings of the SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS'02), January 2002.

[283] ______, *Secure data transmission in mobile ad hoc networks*, In Proceedings of the 2nd ACM Workshop on Wireless Security (WiSe'03), 2003, pp. 41–50.

[284] ______, *Secure link state routing for mobile ad hoc networks*, In Proceedings of the Symposium on Applications and the Internet Workshops (SAINT-W'03), 2003, pp. 379–383.

[285] ______, *Secure message transmission in mobile ad hoc networks*, Ad Hoc Networks **1** (2003), no. 1, 193–209.

[286] James Parker, Jeffrey Undercoffer, John Pinkston, and Anupam Joshi, *On intrustion detection and response for mobile ad hoc networks*, In Proceedings of the 23rd IEEE International Performance, Computing, and CommunicationsConference (IPCCC'04), April 2004, pp. 747–752.

[287] ParSec, *Glomosim, global mobile information systems simulation library*, Url (http://pcl.cs.ucla.edu/projects/glomosim/), Last checked on 20 May, 2010.

[288] Animesh Patcha and Amitab Mishra, *Collaborative security architecture for black hole attack prevention in mobile ad hoc networks*, In Proceedings of the IEEE Radio and Wireless Conference, July 2003, pp. 75–78.

[289] Animesh Patcha and Jung-Min Park, *A game theoretic approach to modeling intrusion detection in mobile ad hoc networks*, In Proceedings of the IEEE Workshop on Information Assurance and Security, June 2004, pp. 30–34.

[290] ______, *A game theoretic formulation for intrusion detection in mobile ad hoc networks*, International Journal of Network Security **2** (2006), no. 2, 131–137.

[291] Al-Sakib Khan Pathan, Md. Mahbub Alam, Md. Mostafa Monowar, and Md. Forhad Rabbi, *An efficient routing protocol for mobile ad hoc networks with neighbor awareness and multicasting*, In Proceedings of the E-Tech, July 2004, pp. 91–100.

[292] Krishna Paul and Dirk Westhoff, *Context aware detection of selfish nodes in dsr based ad-hoc networks*, In Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM'02), vol. 1, November 2002, pp. 178–182.

[293] Wuxu Peng, Yalin Wang, and Kia Makki, *Dynamic key management for secure routing in lcmrmg manet*, In Proceedings of the International Conference On Computer Communications and Networks (ICCCN'04), October 2004, pp. 227–232.

[294] Filip Perich, Jeffrey Undercoffer, Lalana Kagal, Anupam Joshi, Timothy Finin, and Yelena Yesha, *In reputation we believe: Query processing in mobile ad-hoc*

networks, In Proceedings of the Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous'04), 2004, pp. 326–334.

[295] Mark A. Perillo and Wendi Rabiner Heinzelman, *Sensor management policies to provide application qos*, Ad Hoc Networks **1** (2003), no. 2-3, 235–246.

[296] Charles E. Perkins, Elizabeth M. Belding-Royer, and Samir R. Das, *Ad hoc on-demand distance vector (aodv) routing*, Rfc3561, 2003.

[297] Adrian Perrig, Ran Canetti, Dawn Song, and J. D. Tygar, *Efficient and secure source authentication for multicast*, In Proceedings of the Network and Distributed System Security Symposium (NDSS'01), February 2001, pp. 35–46.

[298] Adrian Perrig, Ran Canetti, J. D. Tygar, and Dawn Song, *The tesla broadcast authentication protocol*, RSA CryptoBytes **5** (2002), 2–13.

[299] Adrian Perrig, J. D. Tygar, Dawn Song, and Ran Canetti, *Efficient authentication and signing of multicast streams over lossy channels*, In Proceedings of the IEEE Symposium on Security and Privacy (SP'00), May 2000, pp. 56–73.

[300] Yi Ping, Yao Yan, Hou Yafei, Zhong Yiping, and Zhang Shiyong, *Securing ad hoc networks through mobile agent*, In Proceedings of the 3rd International Conference on Information Security (InfoSecu'04), 2004, pp. 125–129.

[301] Asad Amir Pirzada, Amitava Datta, and Chris McDonald, *Propagating trust in ad-hoc networks for reliable routing*, In Proceedings of the International Workshop Wireless Ad Hoc Networks (IWWAN'04), 2004, pp. 58–62.

[302] ———, *Trust based routing for ad hoc wireless networks*, In Proceedings of the IEEE International Conference on Networks (ICON'04), 2004, pp. 326–330.

[303] ———, *Trustworthy routing with the aodv protocol*, In Proceedings of the International Networking and Communications Conference (INCC'04), 2004, pp. 19–24.

[304] ———, *Trustworthy routing with the tora protocol*, In Proceedings of the AusCERT Asia Pacific Information Technology Security Conference, May 2004.

[305] Asad Amir Pirzada and Chris McDonald, *Establishing trust in pure ad-hoc networks*, In Proceedings of the 27th Australasian Conference on Computer Science (ACSC'04), 2004, pp. 47–54.

[306] ———, *Kerberos assisted authentication in mobile ad-hoc networks*, In Proceedings of the 27th Australasian Conference on Computer Science (ACSC'04), 2004, pp. 41–46.

[307] ———, *Inherent robustness of reactive routing protocols against selfish attacks*, In Proceedings of the International Workshop on Wireless Ad-hoc Networks, 2005.

[308] ———, *Trust establishment in pure ad-hoc networks*, Wireless Personal Communications **37** (2006), no. 1-2, 139–168.

[309] Asad Amir Pirzada, Chris McDonald, and Amitava Datta, *Dependable dynamic source routing without a trusted third party*, In Proceedings of the 28th Australasian Conference on Computer Science (ACSC'05), 2005, pp. 79–85.

[310] ———, *Performance comparison of trust-based reactive routing protocols*, IEEE Transactions on Mobile Computing **5** (2006), no. 6, 695–710.

[311] Niki Pissinou, Tirthankar Ghosh, and Kia Makki, *Collaborative trust-based secure routing in multihop ad hoc networks*, In Proceedings of the Networking Conference, Lecture Notes in Computer Science, vol. 3042, May 2004, pp. 1446–1451.

[312] Olivier Pourret, Patrick Naïm, and Bruce Marcot, *Bayesian networks: A practical guide to applications*, John Wiley and Sons Ltd., 2008.

[313] Nicolas Prigent, Christophe Bidan, Jean-Pierre Andreaux, and Olivier Heen, *Secure long term communities in ad hoc networks*, In Proceedings of the 1st

ACM Workshop on Security of Ad hoc and Sensor Networks (SASN'03), 2003, pp. 115–124.

[314] GNU Project, *Gcc, the gnu compiler collection*, Url (http://gcc.gnu.org/), Last checked on 20 May, 2010.

[315] ———, *Gdb: The gnu project debugger*, Url (http://www.gnu.org/software/gdb/), Last checked on 20 May, 2010.

[316] The VINT Project, *Network simulator (ns2)*, Url (http://www.isi.edu/nsnam/ns/), Last checked on 25 December, 2009.

[317] Jian-Jun Qi and Zeng zhi Li, *Managing trust for secure active networks*, In Proceedings of the 4th International Central and Eastern European Conference on Multi-Agent Systems (CEEMAS'05), Lecture Notes in Computer Science, vol. 3690, 2005, pp. 628–631.

[318] Ying Qiu and Peter Marbach, *Bandwidth allocation in wireless ad hoc networks: A price-based approach*, In Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communication Societies (INFOCOM'03), vol. 2, 2003, pp. 797–807.

[319] Svetlana Radosavac, John S. Baras, and Iordanis Koutsopoulos, *A framework for mac protocol misbehavior detection in wireless networks*, In Proceedings of the Workshop on Wireless Security, 2005, pp. 33–42.

[320] Barath Raghavan and Alex C. Snoeren, *Priority forwarding in ad hoc networks with self-interested parties*, In Proceedings of the Workshop on Economics of Peer-to-Peer Systems, 2003, pp. 13–19.

[321] Prabha Ramachandran and Alec Yasinsac, *Limitations of on demand secure routing protocols*, In Proceedings of the Workshop on Information Assurance, June 2004, pp. 52–59.

[322] Ram Ramanathan and Jason Redi, *A brief overview of ad hoc networks: Challenges and directions*, IEEE Communications Magazine **40** (2002), no. 5, 20–22.

[323] Sanjay Ramaswamy, Huirong Fu, Manohar Sreekantaradhya, John Dixon, and Kendall E. Nygard, *Prevention of cooperative black hole attack in wireless ad hoc networks*, In Proceedings of the International Conference on Wireless Networks, CSREA Press, 2003, pp. 570–575.

[324] Mahalingam Ramkumar and Nasir D. Memon, *An efficient key predistribution scheme for ad hoc network security.*, IEEE Journal on Selected Areas in Communications **23** (2005), no. 3, 611–621.

[325] Maxim Raya and Jean-Pierre Hubaux, *The security of vehicular ad hoc networks*, In Proceedings of the 3rd ACM workshop on Security of Ad hoc and Sensor Networks (SASN'05), 2005, pp. 11–21.

[326] Shukor Abd Razak, Normalia Samian, Mohd. Aizaini Ma'arof, S. M. Furnell, N. L. Clarke, and P. J. Brooke, *A friend mechanism for mobile ad hoc networks*, Journal of Information Assurance and Security **4** (2009), 440–448.

[327] Yacine Rebahi, Vicente E. Mujica-V, and Dorgham Sisalem, *A reputation-based trust mechanism for ad hoc networks*, In Proceedings of the 10th IEEE Symposium on Computers and Communications (ISCC'05), 2005, pp. 37–42.

[328] M. Tamer Refaei, Vivek Srivastava, Luiz DaSilva, and Mohamed Eltoweissy, *A reputation-based mechanism for isolating selfish nodes in ad hoc networks*, In Proceedings of the 2nd Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MOBIQUITOUS'05), 2005, pp. 3–11.

[329] Jalel Rejeb, Meenakshi Vohra, and Thuy T. Le, *Ike-based secure wireless and mobile networks*, In Proceedings of the 6th IEEE Circuits and Systems Symposium on Emerging Technologies: Frontiers of Mobile and Wireless Communication, vol. 2, May 2004, pp. 567–570.

[330] Jian Ren and Jie Wu, *Survey on anonymous communications in computer networks*, Computer Communications **33** (2010), no. 4, 420–431.

[331] Wei Ren, Dit yan Yeung, Hai Jin, and Mei Yang, *Pulsing roq ddos attack and defense scheme in mobile ad hoc networks*, International Journal of Network Security **4** (2007), no. 2, 227–234.

[332] Sini Ruohomaa and Lea Kutvonen, *Trust management survey*, In Proceedings of the 1st International Conference on Trust Management (iTrust'05), vol. 3477, 2005, pp. 77–92.

[333] Yosuke Sagawa, Tomonori Asano, and Hiroaki Higaki, *Loop-based source routing protocol for mobile ad-hoc networks*, In Proceedings of the 17th International Conference on Advanced Information Networking and Applications (AINA'03), March 2003, pp. 834–837.

[334] Naouel Ben Salem, Levente Buttyán, Jean-Pierre Hubaux, and Markus Jakobsson, *A charging and rewarding scheme for packet forwarding in multi-hop cellular networks*, In Proceedings of the 4th ACM international Symposium on Mobile Ad hoc Networking and Computing (MobiHoc'03), 2003, pp. 13–24.

[335] Vishal Sankhla, *Smart: A small world based reputation system for manets*, Master's Thesis, Department of Electrical Engineering, University of Southern California, October 2004.

[336] Paolo Santi, *Topology control in wireless ad hoc and sensor networks*, ACM Computer Survey **37** (2005), no. 2, 164–194.

[337] Kimaya Sanzgiri, Daniel Laflamme, Bridget Dahill, Brian Neil Levine, Clay Shields, and Elizabeth M. Belding-royer, *Authenticated routing for ad hoc networks*, IEEE Journal On Selected Areas In Communications **23** (2005), 598–610.

[338] Slavisa Sarafijanovic and Jean-Yves Le Boudec, *An artificial immune system for misbehavior detection in mobile ad-hoc networks with virtual thymus, clustering,*

danger signal, and memory detectors., In Proceedings of the 3rd International Conference on Artificial Immune Systems (ICARIS'04), Lecture Notes in Computer Science, vol. 3239, 2004, pp. 342–356.

[339] ______, *An artificial immune system approach with secondary response for misbehavior detection in mobile ad-hoc networks*, IEEE Transactions on Neural Networks: Special Issue on Adaptive Learning Systems in Communication Networks **16** (2005), no. 5, 1076–1087.

[340] ______, *An artificial immune system for misbehavior detection in mobile ad-hoc networks with virtual thymus, clustering, danger signal and memory detectors*, International Journal of Unconventional Computing **1** (2005), 221–254.

[341] Fumiaki Sato, Hirohisa Takahira, and Tadanori Mizuno, *Message authentication scheme for mobile ad hoc networks*, In Proceedings of the 11th International Conference on Parallel and Distributed Systems (ICPADS'05), 2005, pp. 50–56.

[342] Reijo Savola and Jarkko Holappa, *Self-measurement of the information security level in a monitoring system based on mobile ad hoc networks*, In Proceedings of IEEE International Workshop on Measurement Systems for Homeland Security, Contraband Detection and Personal Safety (Orlando, FL, USA, 29–30), March 2005, pp. 42–49.

[343] Nitesh Saxena, Gene Tsudik, and Jeong Hyun Yi, *Access control in ad hoc groups*, In Proceedings of the International Workshop on Hot Topics in Peer-to-Peer Systems (HOT-P2P'04), 2004, pp. 2–7.

[344] ______, *Efficient node admission for short-lived mobile ad hoc networks*, In Proceedings of the 13th IEEE International Conference on Network Protocols (ICNP'05), 2005, pp. 269–278.

[345] Rüdiger Schollmeier, Ingo Gruber, and Michael Finkenzeller, *Routing in mobile*

ad-hoc and peer-to-peer networks: A comparison, In Proceedings of the Workshops on Web Engineering and Peer-to-Peer Computing, Lecture Notes in Computer Science, May 2002, pp. 172–186.

[346] Igor Sedov, Sebastian Speicher, and Clemens Cap, *Security management for ad-hoc networked resource-limited mobile devices*, In Proceedings of the 60th IEEE Wireless Technologies for Global Security (VTC'04-Fall), September 2004, pp. 3262–3266.

[347] Diana Senn, *Reputation and trust management in ad hoc networks with misbehaving nodes*, Master's Thesis, Institut für. Technische Informatik und. Kommunikationsnetze, July 2003.

[348] Stefaan Seys and Bart Preneel, *Arm: Anonymous routing protocol for mobile ad hoc networks*, In Proceedings of the 20th IEEE International Conference on Advanced Information Networking and Applications (AINA'06), 2006, pp. 133–137.

[349] Borzoo Shadpour, Shahrokh Valaee, and Baochun Li, *A self-organized approach for stimulating cooperation in mobile ad hoc networks*, In Proceedings of the 22nd Biennial Symposium in Communications, June 2004.

[350] Glenn Shafer, *A mathematical theory of evidence*, Princeton University Press, 1976.

[351] Brian Shand, Nathan Dimmock, and Jean Bacon, *Trust for ubiquitous, transparent collaboration*, Wireless Networks **10** (2004), no. 6, 711–721.

[352] Reza Shokri, Nasser Yazdani, and Ahmad Khonsari, *Chain-based anonymous routing for wireless ad hoc networks*, In Proceedings of the 4th IEEE Consumer Communications and Networking Conference (CCNC'07), 2007, pp. 297–302.

[353] Open Source, *Gnuplot*, Url (http://www.gnuplot.info/), Last checked on 20 May, 2010.

[354] ———, *Ubuntu*, Url (http://www.ubuntu.com/), Last checked on 20 May, 2010.

[355] ———, *Valgrind*, Url (http://valgrind.org/), Last checked on 20 May, 2010.

[356] Avinash Srinivasan, Joshua Teitelbaum, Huigang Liang, Jie Wu, and Mihaela Cardei, *On trust establishment in mobile ad-hoc networks*, ch. Reputation and Trust-based Systems for Ad Hoc and Sensor Networks, Wiley & Sons, 2007.

[357] Vikram Srinivasan, Pavan Nuggehalli, Carla-Fabiana Chiasserini, and Ramesh R. Rao, *Energy efficiency of ad hoc wireless networks with selfish users*, In Proceedings of the European Wireless Conference (EW'02), February 2002.

[358] ———, *An analytical approach to the study of cooperation in wireless ad hoc networks.*, IEEE Transactions on Wireless Communications **4** (2005), no. 2, 722–733.

[359] Vivek Srivastava, James Neel, Allen B. MacKenzie, Rekha Menon, Luiz A. DaSilva, James E. Hicks, Jeffrey H. Reed, and Robert P. Gilles, *Using game theory to analyze wireless ad hoc networks.*, IEEE Communications Surveys and Tutorials **7** (2005), no. 1-4, 46–56.

[360] Frank Stajano and Ross J. Anderson, *The resurrecting duckling: Security issues for ad-hoc wireless networks*, In Proceedings of the 7th International Workshop on Security Protocols, 2000, pp. 172–194.

[361] Ioanna Stamouli, Patroklos G. Argyroudis, and Hitesh Tewari, *Real-time intrusion detection for ad hoc networks*, In Proceedings of the 6th IEEE International Symposium on World of Wireless Mobile and Multimedia Networks (WOWMOM'05), 2005, pp. 374–380.

[362] D. Sterne, P. Balasubramanyam, D. Carman, B. Wilson, R. Talpade, C. Ko, R. Balupari, C-Y. Tseng, T. Bowen, K. Levitt, and J. Rowe, *A general cooperative intrusion detection architecture for manets*, In Proceedings of the 3rd IEEE International Workshop on Information Assurance (IWIA'05), 2005, pp. 57–70.

[363] John A. Stine and Gustavo de Veciana, *A paradigm for quality-of-service in wireless ad hoc networks using synchronous signaling and node states*, IEEE Journal on Selected Areas in Communications **22** (2004), no. 7, 1301–1321.

[364] Ivan Stojmenovic, Amiya Nayak, and Johnson Kuruvila, *Design guidelines for routing protocols in ad hoc and sensor networks with a realistic physical layer*, IEEE Communications Magazine (2005), 101–106.

[365] Maria Striki and John S. Baras, *Towards integrating key distribution with entity authentication for efficient, scalable and secure group communication in manets*, In Proceedings of the IEEE International Conference on Communications, vol. 7, 2004, pp. 4377–4381.

[366] Dhanant Subhadrab, Saswati Sarkar, and Farooq Anjum, *Efficacy of misuse detection in adhoc networks*, In Proceedings of the 1st Annual IEEE International Conference on Sensor and Ad Hoc Communications and Networks (SECON'04), October 2004, pp. 4–13.

[367] V. Sumathy, P. Narayanasmy, K. Baskaran, and T. Purusothaman, *Gls with secure routing in ad-hoc networks*, In Proceedings of the Conference on Convergent Technologies for Asia-Pacific Region (TENCON'03), vol. 3, October 2003, pp. 1072–1076.

[368] Bo Sun, Yong Guan, Jian Chen, and Udo W. Pooch, *Detecting black-hole attack in mobile ad hoc networks*, IEE Conference Publications **2003** (2003), no. CP492, 490–495.

[369] Bo Sun, Kui Wu, and Udo W. Pooch, *Alert aggregation in mobile ad hoc networks*, In Proceedings of the 2nd ACM Workshop on Wireless Security (WiSe'03), 2003, pp. 69–78.

[370] ______, *Routing anomaly detection in mobile ad hoc networks*, In Proceedings of the 12th International Conference on Computer Communications and Networks (ICCCN'03), October 2003, pp. 25–31.

[371] ———, *Towards adaptive intrusion detection in mobile ad hoc networks*, In Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM'04), 2004, pp. 3551–3555.

[372] Yan Lindsay Sun, Zhu Han, Wei Yu, and K. J. Ray Liu, *A trust evaluation framework in distributed networks: Vulnerability analysis and defense against attacks*, In Proceedings of the 25th IEEE International Conference on Computer Communications (INFOCOM'06), April 2006, pp. 1–13.

[373] Yan Lindsay Sun, Wei Yu, Zhu Han, and K. J. Ray Liu, *Information theoretic framework of trust modeling and evaluation for ad hoc networks*, IEEE Journal on Selected Areas in Communications **24** (2006), no. 2, 305–317.

[374] Swaminathan Sundaramurthy and Elizabeth M. Belding-Royer, *The ad-mix protocol for encouraging participation in mobile ad hoc networks*, In Proceedings of the 11th IEEE International Conference on Network Protocols (ICNP'03), November 2003, pp. 156–167.

[375] Karthikeyan Sundaresan, Hung-Yun Hsieh, and Raghupathy Sivakumar, *Ieee 802.11 over multi-hop wireless networks: Problems and new perspectives*, Ad Hoc Networks **2** (2004), no. 2, 109–132.

[376] Girish Suryanarayana, Justin R. Erenkrantz, and Richard N. Taylor, *An architectural approach for decentralized trust management*, IEEE Internet Computing **9** (2005), no. 6, 16–23.

[377] Jian Tang, Guoliang Xue, and Weiyi Zhang, *Reliable routing in mobile ad hoc networks based on mobility prediction*, In Proceedings of the IEEE International Conference on Mobile Ad-hoc and Sensor Systems, October 2004, pp. 466–474.

[378] Scalable Network Technologies, *Qualnet*, Url (http://www.scalable-networks.com/), Last checked on 20 May, 2010.

[379] George Theodorakopoulos and John S. Baras, *Trust evaluation in ad-hoc networks*, In Proceedings of the 3rd ACM Workshop on Wireless Security (WiSe'04), October 2004, pp. 1–10.

[380] Georgios Theodorakopoulos, *Distributed trust evaluation in ad-hoc networks*, Master's Thesis, University of Maryland, 2004.

[381] Tigris, *Subversion, open source software engineering tools*, Url (http://subversion.tigris.org/), Last checked on 20 May, 2010.

[382] Santtu Toivonen, Gabriele Lenzini, and Ilkka Uusitalo, *Context-aware trust evaluation functions for dynamic reconfigurable systems*, In Proceedings of the Models of Trust for the Web Workshop (MTW'06), vol. 190, May 2006.

[383] ______, *Context-aware trustworthiness evaluation with indirect knowledge*, In Proceedings of the 2nd International Conference on Semantic Web Policy Workshop (SWPW'06), October 2006.

[384] Mark Torgerson and Brian Van Leeuwen, *Routing data authentication in wireless ad hoc networks*, Technical Report (SAND2001-3119), Sandia National Laboraties, Albuquerque, New Mexico, October 2001.

[385] Mahmoud Tounsi, Mohamed Hamdi, and Noureddine Boudriga, *A public key-based authentication framework for multi-hop ad hoc networks*, In Proceedings of the 12th IEEE Mediterranean Electrotechnical Conference (MELECON'04), vol. 2, May 2004, pp. 775–778.

[386] Huu Tran, Michael Hitchens, Vijay Varadharajan, and Paul Watters, *A trust based access control framework for p2p file-sharing systems*, In Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05), vol. 9, January 2005, pp. 302c–302c.

[387] Yuh-Ren Tsai and Shiuh-Jeng Wang, *Routing security and authentication mechanism for mobile ad hoc networks*, In Proceedings of the IEEE 60th Vehicular Technology Conference (VTC'04), vol. 7, September 2004, pp. 4716–4720.

[388] Chin-Yang Tseng, Poornima Balasubramanyam, Calvin Ko, Rattapon Limprasittiporn, Jeff Rowe, and Karl Levitt, *A specification-based intrusion detection system for aodv*, In Proceedings of the 1st ACM Workshop on Security of Ad hoc and Sensor Networks (SASN'03), 2003, pp. 125–134.

[389] Chin-Yang Tseng, Tao Song, Poornima Balasubramanyam, Calvin Ko, and Karl N. Levitt, *A specification-based intrusion detection model for olsr*, In Proceedings of the International Symposium on Recent Advances in Intrusion Detection (RAID'05), Lecture Notes in Computer Science, vol. 3858, 2005, pp. 330–350.

[390] Malik Tubaishat and Sanjay Madria, *Sensor networks: An overview*, IEEE Potentials **22** (2003), no. 2, 20–23.

[391] A. Urpi, M. Bonuccelli, and S. Giodano, *Modelling cooperation in mobile ad hoc networks: A formal description of selfishness*, In Proceedings of the Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks, 2003, pp. 3–5.

[392] W. Usaha and Javier A. Barria, *A reinforcement learning ticket-based probing path discovery scheme for manets*, Ad Hoc Networks **2** (2004), no. 3, 319–334.

[393] Vijay Varadharajan, *Trustworthy computing (extended abstract)*, In Proceedings of the ACM Workshop on Wireless Security (WiSe'04), Lecture Notes in Computer Science, vol. 3306, November 2004, pp. 13–16.

[394] ______, *Authorization and trust enhanced security for distributed applications*, In Proceedings of the 7th International Conference on Information and Communications Security (ICICS'05), Lecture Notes in Computer Science, vol. 3803, 2005, pp. 1–20.

[395] Vijay Varadharajan, Rajan Shankaran, and Michael Hitchens, *Security for cluster based ad hoc networks*, Computer Communications **27** (2004), no. 5, 488–501.

[396] Srdjan Čapkun, Levente Buttyán, and Jean-Pierre Hubaux, *Self-organized public-key management for mobile ad hoc networks*, IEEE Transactions on Mobile Computing **2** (2003), no. 1, 52–64.

[397] Srdjan Čapkun and Jean-Pierre Hubaux, *Biss: Building secure routing out of an incomplete set of security associations*, In Proceedings of the 2nd ACM workshop on Wireless Security (WiSe'03), September 2003, pp. 21–29.

[398] Srdjan Čapkun, Jean-Pierre Hubaux, and Levente Buttyán, *Mobility helps security in ad hoc networks*, In Proceedings of the 4th ACM International Symposium on Mobile Ad hoc Networking and Computing (MobiHoc'03), 2003, pp. 46–56.

[399] V. Vetriselvi and R. Parthasarathi, *Secure communication for multipath ad hoc network*, In Proceedings of the Conference on Convergent Technologies for Asia-Pacific Region (TENCON 2003), vol. 3, October 2003, pp. 1086–1090.

[400] Giovanni Vigna, Sumit Gwalani, Kavitha Srinivasan, Elizabeth M. Belding-royer, and Richard A. Kemmerer, *An intrusion detection tool for aodv-based ad hoc wireless networks*, In Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC'04), 2004, pp. 16–27.

[401] Mohit Virendra, Murtuza Jadliwala, Madhusudhanan Chandrasekaran, and Shambhu Upadhyaya, *Quantifying trust in mobile ad-hoc networks*, In Proceedings of the IEEE International Conference on Integration of Knowledge Intensive Multiagent Systems (KIMAS'05), April 2005, pp. 65–70.

[402] Peng-Jun Wan, Gruia Călinescu, and Chih-Wei Yi, *Minimum-power multicast routing in static ad hoc wireless networks*, IEEE/ACM Transactions on Networking **12** (2004), no. 3, 507–514.

[403] Bo Wang, Sohraab Soltani, Jonathan K. Shapiro, and Pang-Ning Tan, *Local detection of selfish routing behavior in ad hoc networks*, In Proceedings of the 8th International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN'05), 2005, pp. 392–399.

[404] Cuirong Wang, Xiaozong Yang, and Yuan Gao, *A routing protocol based on trust for manets*, In Proceeding of the 6th Annual International Conference on Grid

and Cooperative Computing (GCC'05), Lecture Notes in Computer Science, vol. 3795, 2005, pp. 959–964.

[405] Pan Wang, Douglas S. Reeves, and Peng Ning, *Secure address auto-configuration for mobile ad hoc networks*, In Proceedings of the 2nd Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MOBIQUITOUS'05), 2005, pp. 519–522.

[406] Weichao Wang, Yi Lu, and Bharat Bhargava, *On security study of two distance vector routing protocols or mobile ad hoc networks*, In Proceedings of the 1st IEEE International Conference on Pervasive Computing and Communications (PERCOM'03), 2003, pp. 179–186.

[407] Weihong Wang, Ying Zhu, and Baochun Li, *Self-managed heterogeneous certification in mobile ad hoc networks*, In Proceedings of the Vehicular Technology Conference, 2003, pp. 2137–2141.

[408] Weizhao Wang and Xiang-Yang Li, *Low-cost truthful multicast in selfish and rational wireless ad hoc networks*, In Proceedings of the 1st IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS'04), October 2004, pp. 534–536.

[409] Yong Wang, Garhan Attebury, and Byrav Ramamurthy, *A survey of security issues in wireless sensor networks*, IEEE Communications Surveys and Tutorials **8** (2006), no. 1-4, 2–23.

[410] Yu Wang and J. J. Garcia-Luna-Aceves, *Performance of collision avoidance protocols in single-channel ad hoc networks*, In Proceedings of the 10th IEEE International Conference on Network Protocols (ICNP'02), November 2002, pp. 68–77.

[411] Damian Watkins and Craig Scott, *Methodology for evaluating the effectiveness of intrusion detection in tactical mobile ad-hoc networks*, In Proceedings of the IEEE Conference on Wireless Communications and Networking Conference (WCNC'04), no. 0-7803-8344-3/04, 2004, pp. 622–627.

[412] Guo Wei, Xiong Zhongwei, and Li Zhitang, *Dynamic trust evaluation based routing model for ad hoc networks*, In Proceedings of the International Conference on Wireless Communications, Networking and Mobile Computing, vol. 2, September 2005, pp. 727–730.

[413] Andre Weimerskirch and Dirk Westhoff, *Identity certified authentication for ad-hoc networks*, In Proceedings of the 1st ACM Workshop on Security of Ad hoc and Sensor Networks (SASN'03), 2003, pp. 33–40.

[414] Kilian Weniger, *Passive duplicate address detection in mobile ad hoc networks*, In Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC'03), vol. 3, March 2003, pp. 1504–1509.

[415] ______, *Pacman: Passive autoconfiguration for mobile ad hoc networks*, IEEE Journal on Selected Areas in Communications (JSAC): Special Issue on 'Wireless Ad hoc Networks' (2005), 507–519.

[416] Attila Weyland and Torsten Braun, *Cashnet - cooperation and accounting strategy for hybrid networks*, In Proceedings of the 2nd Workshop on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt'04), 2004, pp. 423–424.

[417] Anthony D. Wood and John A. Stankovic, *Denial of service in sensor networks*, IEEE Computer **35** (2002), no. 10, 54–62.

[418] Konrad Wrona, *Distributed security: Ad hoc networks and beyond*, In Proceedings of the PAMPAS Workshop, September 2002.

[419] Bing Wu, Jianmin Chen, Jie Wu, and Mihaela Cardei, *Wireless network security*, Signals and Communication Technology, ch. A Survey of Attacks and Countermeasures in Mobile Ad Hoc Networks, pp. 103–135, Springer US, December 2007.

[420] Bing Wu, Jie Wu, Eduardo B. Fernandez, Mohammad Ilyas, and Spyros Magliveras, *Secure and efficient key management in mobile ad hoc networks*, Journal of Network Computer Application **30** (2007), no. 3, 937–954.

[421] Yan Xia, Ren-Fa Li, and Ken-Li Li, *Intrusion detection using mobile agent in ad-hoc networks*, In Proceedings of the 3rd International Conferenceon Machine Leaming and Cybernetics, vol. 6, August 2004, pp. 3383–3388.

[422] Chaoyue Xiong, Tadao Murata, and Jason Leigh, *An approach for verifying routing protocols in mobile ad hoc networks using petri nets*, In Proceedings of the 6th IEEE CAS Symposium on Emerging Technologies: Mobile and Wireless Communications, vol. 2, May 2004, pp. 537–540.

[423] Gang Xu and Liviu Iftode, *Locality driven key management architecture for mobile ad hoc networks*, In Proceedings of the 1st IEEE International Conference on Mobile Ad-hoc and Sensor Systems, October 2004, pp. 436–446.

[424] Qi Xu, *Secure forwarding in personal ad hoc networks*, Master's Thesis, Faculty of Electrical Engineering, Mathematics and Computer Science, University of Twente, May 2005.

[425] Zheng Yan and Peng Zhang, *Trust evaluation based security solution in ad hoc networks*, In Proceedings of the 7th Nordic Workshop on Secure IT Systems, October 2003.

[426] Chou-Chen Yang, Chia-Meng Chen, and Ting-Yi Chang, *A routing authentication mechanism for wireless ad hoc networks*, In Proceedings of the IEEE lnternational Conference on Networking, Sensing and Control, vol. 1, March 2004, pp. 456–461.

[427] Hao Yang, Haiyun Luo, Fan Ye, and Songwu Lyand Lixia Zhang, *Security in mobile ad hoc networks: Challenges and solutions*, IEEE Wireless Communications (2004), 38–47.

[428] Hao Yang, Xiaoqiao Meng, and Songwu Lu, *Self-organized network-layer security in mobile ad hoc networks*, In Proceedings of the 1st ACM Workshop on Wireless Security (WiSe'02), 2002, pp. 11–20.

[429] Rui Jun Yang, Qun Hua Pan, Wei Nong Wang, and Ming Lu Li, *Secure enhancement scheme for routing protocol in mobile ad hoc networks*, In Proceedings of the 3rd International Workshop on Mobile Distributed Computing (ICDCSW'05), 2005, pp. 664–667.

[430] Shahan Yang and John S. Baras, *Modeling vulnerabilities of ad hoc routing protocols*, In Proceedings of the 1st ACM Workshop on Security of Ad hoc and Sensor Networks (SASN'03), 2003, pp. 12–20.

[431] Jun Yao and Guihua Zeng, *Key agreement and identity authentication protocols for ad hoc networks*, In Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04), vol. 2, 2004, pp. 720–724.

[432] Walt Teh-Ming Yao, *Trust management for widely distributed systems*, Ph.d. Thesis, Jesus College, University of Cambridge, February 2003.

[433] Po-Wah Yau and Chris J. Mitchell, *Reputation methods for routing security for mobile ad hoc networks*, In Proceedings of the 1st Joint Workshop on Mobile Future and Symposium on Trends in Communications (SympoTIC '03), October 2003, pp. 130–137.

[434] ______, *Security vulnerabilities in ad hoc networks*, In Proceedings of the 7th International Symposium on Communication Theory and Applications, July 2003, pp. 99–104.

[435] ______, *2harp: A secure routing protocol to detect failed and selfish nodes in mobile ad hoc networks*, In Proceedings of the 5th World Wireless Congress (WWC'04), May 2004, pp. 1–6.

[436] Zhenqiang Ye, Srikanth V. Krishnamurthy, and Satish K. Tripathi, *A routing framework for providing robustness to node failures in mobile ad hoc networks*, Ad Hoc Networks **2** (2004), no. 1, 87–107.

[437] Seung Yi and Robin Kravets, *Key management for heterogeneous ad hoc wireless*

networks, In Proceedings of the 10th IEEE International Conference on Network Protocols (ICNP'02), 2002, pp. 202–205.

[438] ———, *Composite key management for ad hoc networks*, In Proceedings of the 1st Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous'04), 2004, pp. 52–61.

[439] Seung Yi, Prasad Naldurg, and Robin Kravets, *Security-aware ad hoc routing for wireless networks*, In Proceedings of the 2nd ACM International Symposium on Mobile Ad Hoc Networking and Computing, 2001, pp. 299–302.

[440] ———, *A security-aware routing protocol for wireless ad hoc networks*, In Proceedings of the 3rd ACM International of Mobile Ad hoc Networking and Computing, 2002, pp. 226–236.

[441] Younghwan Yoo and Sanghyun Ahn, *A simple load-balancing approach in cheat-proof ad hoc networks*, In Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM'04), December 2004, pp. 3573–3577.

[442] G. S Yovanof and K. Erikci, *Performance evaluation of security-aware routing protocols for clustered mobile ad hoc networks*, In Proceedings of the International Workshop on Wireless Ad-Hoc Networks, June 2004, pp. 286–290.

[443] Wei Yu and K. J. R. Liu, *Attack-resistant cooperation stimulation in autonomous ad hoc networks*, IEEE Journal on Selected Areas in Communications **23** (2005), no. 12, 2260–2271.

[444] Wei Yu, Yan Sun, and K. J. Ray Liu, *Hadof: Defense against routing disruptions in mobile ad hoc networks*, In Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'05), March 2005, pp. 1252–1261.

[445] Manel Guerrero Zapata, *Secure ad hoc on-demand distance vector routing*, ACM SIGMOBILE Mobile Computing and Communications Review **6** (2002), no. 3, 106–107.

[446] Manel Guerrero Zapata and N. Asokan, *Securing ad hoc routing protocols*, In Proceedings of the 1st ACM Workshop on Wireless Security (WiSe'02), 2002, pp. 1–10.

[447] Chi Zhang, Xiaoyan Zhu, Yang Song, and Yuguang Fang, *A formal study of trust-based routing in wireless ad hoc networks*, In Proceedings of the 29nd Annual Joint Conference of the IEEE Computer and Communications (INFOCOM'10), March 2010, pp. 1–9.

[448] Junqi Zhang, Vijay Varadharajan, and Yi Mu, *A scalable multi-service group key management scheme*, In Proceedings of the Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services (AICT/ICIW'06), 2006, pp. 172–177.

[449] Qi Zhang and Dharma P. Agrawal, *Impact of selfish nodes on route discovery in mobile ad hoc networks*, In Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM'04), October 2004, pp. 2914–2918.

[450] Yanchao Zhang, Wei Liu, and Wenjing Lou, *Anonymous communications in mobile ad hoc networks*, In Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'05), 2005, pp. 1940–1951.

[451] Yanchao Zhang, Wei Liu, Wenjing Lou, Yuguang Fang, and Younggoo Kwon, *Ac-pki: Anonymous and certificateless public-key infrastructure for mobile ad hoc networks*, In Proceedings of the IEEE International Conference on Communications, May 2005, pp. 3515–3519.

[452] Yanchao Zhang, Wenjing Lou, and Yuguang Fang, *Sip: A secure incentive protocol against selfishness in mobile ad hoc networks*, In Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC'04), vol. 3, 2004, pp. 1679–1684.

[453] Yanchao Zhang, Wenjing Lou, Wei Liu, and Yuguang Fang, *A secure incentive protocol for mobile ad hoc networks*, Wireless Networks **13** (2007), no. 5, 569–582.

[454] Qing Zhao and Lang Tong, *A multiqueue service room mac protocol for wireless networks with multipacket reception*, IEEE/ACM Transactions on Networking **11** (2003), no. 1, 125–137.

[455] Jane Zhen and Sampelli Srinivas, *Preventing replay attacks for secure routing in ad hoc networks*, In Proceedings of the ADHOC-NOW, Lecture Notes in Computer Science, vol. 2865, 2003, pp. 140–150.

[456] Sheng Zhong, Jiang Chen, and Yang Richard Yang:, *Sprite: A simple, cheat-proof, credit-based system for mobile ad-hoc networks*, In Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'03), vol. 3, 2003, pp. 1987–1997.

[457] Sheng Zhong, Li Erran Li, Yanbin Grace Liu, and Yang Richard Yang, *On designing incentive-compatible routing and forwarding protocols in wireless ad-hoc networks: An integrated approach using game theoretical and cryptographic techniques*, Wireless Networks **13** (2007), no. 6, 799–816.

[458] Lidong Zhou and Zygmunt J. Haas, *Securing ad hoc networks*, IEEE Network **13** (1999), no. 6, 24–30.

[459] Yihong Zhou, Dapeng Wu, and Scott M. Nettles, *Analyzing and preventing mac-layer denial of service attacks for stock 802.11 systems*, In Proceedings of the 1st IEEE/ACM Workshop on Broadband Wireless Services and Applications (BroadWISE'04), October 2004, pp. 22–28.

[460] Bo Zhu, Zhiguo Wan, Mohan S. Kankanhalli, Feng Bao, and Robert H. Deng, *Anonymous secure routing in mobile ad-hoc networks*, In Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks (LCN'04), November 2004, pp. 102–108.

[461] Danyu Zhu and Matt Mutka, *Promoting cooperation among strangers to access internet services from an ad hoc network*, In Proceedings of the 2nd IEEE International Conference on Pervasive Computing and Communications (PerCom'04), 2004, pp. 229–238.

[462] Huafei Zhu, Feng Bao, and Robert H. Deng, *Computing of trust in distributed networks*, Technical Report (Cryptology ePrint Archive: Report 2003/056), InfoComm Security Department, Institute for InfoComm Research, Singapore, 2003.

[463] ———, *Computing of trust in wireless networks*, In Proceedings of the 60th IEEE Vehicular Technology Conference, September 2004, pp. 2621–2624.

[464] Sencun Zhu, Sanjeev Setia, Shouhuai Xu, and Sushil Jajodia, *Gkmpan: An efficient group rekeying scheme for secure multicast in ad-hoc networks*, Journal of Computer Security **14** (2006), no. 4, 301–325.

[465] Sencun Zhu, Shouhuai Xu, Sanjeev Setia, and Sushil Jajodia, *Establishing pairwise keys for secure communication in ad hoc networks: A probabilistic approach*, In Proceedings of the 11th IEEE International Conference on Network Protocols (ICNP'03), 2003, pp. 326–335.

[466] ———, *Lhap: A lightweight hop-by-hop authentication protocol for ad-hoc networks*, In Proceedings of the 23rd International Conference on Distributed Computing Systems (ICDCSW'03), 2003, pp. 749–755.

[467] P. R. Zimmermann, *The official pgp user's guide*, MIT Press, 1995.

[468] Charikleia Zouridaki, Brian L. Mark, Marek Hejmo, and Roshan K. Thomas, *A quantitative trust establishment framework for reliable data packet delivery in manets*, In Proceedings of the 3rd ACM Workshop on Security of Ad hoc and Sensor Networks (SASN'05), November 2005, pp. 1–10.