Trust-based Routing in Pure Ad-hoc Wireless Networks

This thesis is

presented to the

School of Computer Science & Software Engineering

for the degree of

Doctor of Philosophy

of

The University of Western Australia

By

Asad Amir Pirzada

July 2007

"Trust is the outcome of observations leading to the belief that the actions of another may be relied upon, without explicit guarantee, to achieve a goal in a risky situation."

Greg Elofson [25]

Abstract

An ad-hoc network of wireless nodes is a temporarily formed network, created, operated and managed by the nodes themselves. Due to its peculiar establishment and operational properties it is also often termed an infrastructure-less, self-organised, or spontaneous network. In order to extend the communication range of the nodes, beyond a single hop, specially configured routing protocols are used. The unique feature of these protocols is their ability to form routes in spite of a dynamic topology.

For effective functioning of the network it is essential that the network nodes execute the routing protocols in a truthful manner regardless of their contemporary commitments and workload. In real life, this is more than often extremely difficult to realise, and so we often find malicious nodes also present in the same network. These nodes can either join externally or may originate internally by compromising an existing benevolent node in the network. These malicious nodes can carry out an array of attacks against the routing protocols leading to route severing, unavailability of service or deception.

A number of secure routing protocols, which make use of cryptographic algorithms to secure the routes, have recently been proposed. However in doing so, these protocols entail a number of prerequisites both during the network establishment and operation phase. These requirements include configuration of the nodes prior to joining the network and creation of a centralised or distributed trusted third party, creating a somewhat managed, rather than a pure, ad-hoc network.

Pure ad-hoc networks operate in a cooperative manner, which closely resembles the human behaviour, where a number of people that have never met each other, are able to communicate with each other based on mutual trust levels developed over a period of time. In order to sustain the improvised nature of ad-hoc networks, in this thesis, we have moved from the common mechanism of achieving trust via security to enforcing dependability through collaboration. We desist from the customary strategy of employing cryptography and instead use a trust model that is influenced by the human behavioural model. All nodes in the network independently execute this trust model and maintain their own assessment concerning other nodes in the network. Each node, based upon its individual experiences, rewards collaborating nodes for their benevolent behaviour and penalises malicious nodes for their malevolent conduct.

To highlight the efficacy of this unique approach, we apply the trust model to three contemporary reactive routing protocols in a pure ad-hoc network. These trust reinforced routing protocols locate dependable routes in the network by observing the sincerity in participation of other nodes using a set of trust categories. The routes worked out in this way are neither protected in terms of security nor minimal in terms of hops. However, these routes traverse nodes, which have been identified as more trustworthy than others and for this reason are more dependable in extemporised circumstances. Through the help of extensive simulations, we have demonstrated that the usage of these protocols significantly improves the overall performance of the network even in the presence of a high percentage of malicious nodes. These protocols, being independent of a trust infrastructure, also enable rapid deployment and improved operation with dynamic adaptation to the current scenario. The prime advantage being gained is the ability to seamlessly integrate ad-hoc wireless networks belonging to dissimilar organisations.

Preface

Parts of this thesis have already been published in a number of international refereed journals and conference proceedings. Published papers are cited appropriately throughout this thesis, and include:

- A. A. Pirzada and C. McDonald, "Secure Routing Protocols for Mobile Adhoc Wireless Networks," in Advanced Wired and Wireless Networks, T. A. Wysocki, A. Dadej, and B. J. Wysocki, Eds., Springer, Chapter 4, pages 57-80, 2004.

- A. A. Pirzada and C. McDonald, "Reliable Routing in Ad-hoc Networks using Direct Trust Mechanisms," Advances in Wireless Ad Hoc and Sensor Networks in book series Signals and Communication Technology, M. X. Cheng and D. Li, Eds., Springer, Chapter 6, 2007.

- A. A. Pirzada, C. McDonald, and A. Datta, "Performance Comparison of Trust-Based Reactive Routing Protocols," IEEE Transactions on Mobile Computing, 5(6):695-710, 2006.

- A. A. Pirzada and C. McDonald, "Performance Comparison of Reactive Routing Protocols under Attack Conditions," International Journal on Wireless & Optical Communications, World Scientific Publishing, 2(2):163-180, 2004.

- A. A. Pirzada, A. Datta, and C. McDonald, "Incorporating Trust and Reputation in the DSR Protocol for Dependable Routing," Elsevier Computer Communications Special Issue on Internet Communications Security, 29(15):2806-2821, 2006.

- A. A. Pirzada and C. McDonald, "Detecting and Evading Wormholes in Mobile Ad-hoc Wireless Networks," International Journal of Network Security, 3(2):188-199, 2006.

- A. A. Pirzada and C. McDonald, Securing Ad-hoc Networks Using Cryptographic Hashes, Journal of China Information Security, ICISA Press, ISSN 1009-8054, pages 436-445, 2004.

- A. A. Pirzada, A. Datta, and C. McDonald, "Trust Based Routing for Ad-hoc Wireless Networks," Proceedings of the IEEE International Conference on Networks (ICON'04), Vol. 1, pages 326-330, 2004.

- A. A. Pirzada, A. Datta, and C. McDonald, "Propagating Trust in Ad-hoc Networks for Reliable Routing," Proceedings of the International Workshop on Wireless Ad-hoc Networks (IWWAN'04), IEEE Computer Society, 2004.

- A. A. Pirzada and C. McDonald, "Establishing Trust In Pure Ad-hoc Networks," Proceedings of the 27th Australasian Computer Science Conference (ACSC'04), Vol. 26, pages 47-54, 2004.

- A. A. Pirzada, C. McDonald, and A. Datta, "Dependable Dynamic Source Routing without a Trusted Third Party," Proceedings of the 28th Australasian Computer Science Conference (ACSC'05), Vol. 38, pages 79-85, 2005.

- A. A. Pirzada and C. McDonald, "Inherent Robustness of Reactive Routing Protocols against Selfish Attacks," (best student paper), Proceedings of the International Workshop on Wireless Ad-hoc Networks (IWWAN'05), IEEE Computer Society, 2005.

- A. A. Pirzada, C. McDonald and A. Datta, "Reliable Link Reversal Routing for Mobile Ad-hoc Wireless Networks," Proceedings of the IEEE International Conference on Networks (ICON'05), Vol. 1, pages 234-239, 2005.

- A. A. Pirzada and C. McDonald, "Deploying Trust Gateways to Reinforce Dynamic Source Routing," Proceedings of the IEEE Conference on Industrial Informatics (INDIN'05), 2005.

The primary author of each of these publications, and the author responsible for the majority of the original research presented herein, is the author of this thesis.

Acknowledgements

First and foremost I would like to thank my Lord for giving me the opportunity, strength and determination to do a PhD at this prestigious institution. Without His incessant grace and mercy, my life time dream of doing a PhD could have never realised. I would also like to acknowledge here certain individuals who helped, supported and guided me during the tenure of my PhD.

I'm deeply indebted to the mentorship of my supervisor, Dr. Chris McDonald. In addition to being an excellent supervisor, he is a man of principles and I feel fortunate enough to be one of his students. His constant motivation and guidance, enabled me to traverse the different hurdles faced during my research. Also, the long hours he dedicated to working on papers and talking through ideas helped seed and solidify many of the new concepts presented in this thesis. I'm also thankful for the opportunity to work with Assoc. Prof Amitava Datta and Dr. Rachel Cardell-Oliver, who provided much inspiration, ideas, and insight into our research work.

I deeply appreciate the support of my family members during my PhD. During the numerous critical and intricate stages of my PhD, I have always found the prayers of my parents helping me out with them. However, I can never forget the sacrifices of my beautiful wife Shaghila and my two angels, Salaar and Menahil, who had to bear the direct impact of my PhD on their quality of life. Shaghila, without you I would have never been able to fulfill my dream. It was due to your shouldering of my responsibilities, which enabled me to devote myself to research. You are the

most loving, understanding and considerate companion that anyone could ever wish for and I owe you this achievement. Salaar and Menahil, you are the joy of my life and I appreciate your patience for forgoing the precious time that we could have spent together.

Special thanks go to the staff of the School of Computer Science & Software Engineering at The University of Western Australia for providing me with a pleasant work environment with excellent computing facilities. I would like to specifically thank the Head of School Dr. Nick Spadaccini, the School Manager Mrs. Louise Bolitho, the Computer Systems Administrator Mr. Ashley Chew and all other school staff for readily providing the requisite administrative or technical assistance, whenever it was requested.

General thanks to the National University of Sciences & Technology (NUST) Pakistan, Australian Government, The University of Western Australia and the School of Computer Science & Software Engineering for supporting me through the NUST Split MS/PhD Scholarship, International Postgraduate Research Scholarship (IPRS), the University Postgraduate Award for International Students (UPAIS) and an Adhoc Scholarship.

I would also like to thank Mahesh K. Marina from the Stony Brook University, New York, USA and Diana Senn from the ETH Information Security Group, Zurich, Switzerland for contributing NS-2 code in support of the simulations carried out during my research.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The demand for ubiquitous communication is constantly on the rise, with wireless technology being one of its potential solutions. Mobile Ad-hoc Networks (MANET) facilitate spontaneous setting up of communication with minimal reaction time, anywhere at anytime, which makes them an ideal candidate for makeshift situations like rescue, law enforcement and military operations. This independence from external infrastructure not only leverages the scalability of these networks but also liberates them from positional restraints.

The network nodes usually comprise computing devices with low power wireless transceivers, which severely limit the maximum range of these nodes. In order to overcome the limited transmission and reception barrier, mobile ad-hoc networks operate on a 'give and take' principle. Each node agrees to forward packets on behalf of other nodes in the network in return of a similar favour from others. Such a munificent atmosphere allows nodes to increase the effective range beyond their individual physical range. However, the mobility of the nodes aggravates the already fragile cooperative settings, and so forwarding nodes need to cater for the dynamic topology as well.

Specially configured routing protocols enable mobile nodes to institute and sustain operational routes in the network. These routing protocols can be categorised

into two main types: Proactive and Reactive [99]. Proactive routing protocols exchange route or link tables on a periodic basis and hence maintain up to date working routes at all instants of time. However in doing so, these protocols devour additional battery power as a result of regular exchange of connectivity information. In contrast, reactive routing protocols only discover routes in the network when no prior route information exists regarding a particular node. These route discoveries are initiated in an on-demand manner, and so help in conserving battery power, which is considered a vital resource in ad-hoc networks. Information retrieved through these route discoveries is retained and exploited to its utmost with an aim to minimise subsequent route discoveries. Performance comparison of proactive and reactive routing protocols indicate superior performance of the latter under diverse mobility conditions [8, 54]. Due to aforementioned reasons, in this thesis, we have primarily focussed our research on reactive routing protocols.

Both types of routing protocols require persistent cooperative behaviour from the intermediate nodes that primarily contribute to the route development. Similarly all nodes, which practically act like mobile routers [16], have the absolute control over the data that passes through them. In essence, the membership of any ad-hoc network indisputably calls for sustained benevolent behaviour from all participating nodes. However, due to a variety of problems, including the lack of security in the medium, energy and processing constraints, mobility and poor physical protection of the nodes, such an altruistic stance cannot always be guaranteed. As a result, it is possible to launch various attacks against these networks either from a malicious, compromised or selfish node [13].

In general, these attacks can by and large be divided into two types: Passive and Active [34]. In passive attacks the attacker does not disturb the routing protocol. It only eavesdrops on the routing traffic and endeavours to extract valuable information like node hierarchy and network topology from it. For example, if a route to a particular node is requested more frequently than to other nodes, the attacker might anticipate that the node is vital for the operation of the network,

and putting it out of action could bring down the entire network. Similarly, even when it might not be possible to isolate the precise position of a node, one may be able to determine information about the network topology by analysing the contents of routing packets. This attack is virtually impossible to detect in the wireless environment and hence also extremely difficult to prevent. In active attacks, the aggressor node has to expend some of its energy in order to carry out the attack. These attacks may be carried out against any layer of the TCP/IP protocol stack and range from communication jamming [109] to a vertex cut [75] in the network. Nodes that perform active attacks with the aim of disrupting other nodes by causing network outage are considered to be malicious, while nodes that perform attacks with the aim of saving power or computational resources for their own communications are considered to be selfish [62]. In active attacks, malicious nodes can disrupt the correct functioning of a routing protocol by modifying routing information, by fabricating false routing information or by impersonating nodes [19].

Attacks Using Modification: Routing protocols, for mobile ad-hoc wireless networks, operate on the assumption that intermediate nodes do not maliciously change the protocol fields of messages passed between nodes. This assumed trust permits malicious nodes to easily generate traffic subversion and Denial of Service (DoS) attacks [55]. Attacks using modification are generally targeted against the integrity of routing computations, and so by modifying routing information an attacker can cause network traffic to be dropped, redirected to a different destination, or take a longer route to the destination increasing communication delays. An example is for an attacker to send fake routing packets to generate a routing loop [48], causing packets to pass through nodes in a cycle without getting to their actual destinations, consuming energy and bandwidth. Similarly, by sending forged routing packets to other nodes, all traffic can be diverted to the attacker or to some other node. The idea is to create a black hole by routing all packets to the attacker and then discarding them. As an extension to the black hole, an attacker could

build a grey hole, in which it intentionally drops some packets but not others, for example, forwarding routing packets but not data packets. A more subtle type of modification attack is the creation of a tunnel or wormhole [75] in the network between two colluding malicious nodes linked through a private network connection. This exploit allows a node to short-circuit the normal flow of routing messages creating a virtual vertex cut in the network that is controlled by the two colluding attackers.

Attacks Using Fabrication: Fabrication attacks are carried out by generating false routing messages. These attacks are difficult to identify as the messages are received as legitimate routing packets. The rushing attack [35] is a typical example of a malicious attack, which is accomplished through fabrication. This attack is carried out against on-demand routing protocols that hold back duplicate packets at every node. An attacker rapidly spreads routing messages all through the network, suppressing legitimate routing messages when nodes discard them as duplicate copies. Similarly, an attacker can nullify an operational route to a destination by fabricating routing error messages asserting that a neighbour can no longer be contacted.

Attacks Using Impersonation: A malicious node can initiate many attacks in a network by masquerading as another node (spoofing) [30]. Spoofing occurs when a malicious node misrepresents its identity by altering its MAC or IP address in order to alter the vision of the network topology that a benign node can gather. As an example, a spoofing attack allows the creation of loops in the routing information collected by a node, with the result of partitioning the network.

In order to protect ad-hoc networks against selfish and malicious behaviour, a number of secure routing protocols have been developed. These protocols employ a variety of cryptographic tools to protect the core routing protocol, which in turn

secures the routes, thereby protecting the data that flows through them. More emphasis is laid upon the precise establishment of the routes, as there is no advantage in protecting the data if it never reaches its requisite destination. We carried out a comparison of contemporary secure routing protocols, which have been developed for mobile ad-hoc wireless networks [85]. The comparison indicates that most of the protocols presume the existence of a centralised or distributed trusted third party in the network. This assumption is also coupled with the pre-configuration of nodes with encryption keys prior to joining the network. As the name suggests, ad-hoc wireless networks are rarely established in a planned manner. They have the diverse nature of being impromptu, and so inherently oppose the dependence upon prerequisites. These networks are essentially impulsive and lack the instinctive attribute of planning and thus oppose extraneous requirements.

Trust and security are two tightly coupled concepts that cannot be desegregated [44, 50]. For example, cryptography is a means to implement security but it is highly dependent on trusted key exchange [2]. Similarly, trusted key exchange cannot take place without requisite security services in place. It is because of this inter-reliance that both of these terms are used interchangeably when defining a secure system. Trust in wired networks is usually achieved using indirect trust mechanisms, including trusted certification agencies and authentication servers. However, establishing this indirect trust still requires some out-of-band mechanism for initial authentication and is usually augmented with physical or location-based authentication schemes [110, 5].

Trust establishment in ad-hoc wireless networks is still an open and challenging field. Ad-hoc networks are based on naive 'trust-your-neighbour' relationships. These relationships originate, develop and expire on the fly and usually have a short lifespan. As the overall environment in such a network is cooperative by default, these trust relationships are extremely susceptible to attacks. For a number of reasons, including better service, selfishness, monetary benefits or malicious intent, some nodes can easily mould these relationships to extract desired goals. Also, the absence of

fixed trust infrastructure, limited resources, ephemeral connectivity and availability, shared wireless medium and physical vulnerability, make trust establishment virtually impossible. To overcome these problems, trust has been established in ad-hoc networks using a number of assumptions including pre-configuration of nodes with secret keys, or an omnipresent central trust authority [112]. In our opinion, these assumptions are against the very nature of ad-hoc networks, which are supposed to be improvised and spontaneous. We categorise the ones that are based on assumptions as 'managed ad-hoc networks' and those without these as 'pure ad-hoc networks'.

Managed ad-hoc networks are those, which have the provision of sustaining a trusted third party in the network. These networks thus require some a priori knowledge concerning the volume and setting of the network. This in turn restricts the size and mode of application of these networks. Managed ad-hoc networks are thus deemed suitable for law enforcement and military set-ups that generally have prior knowledge of forthcoming requirements. Establishing trust in managed networks is relatively simple to realise, as the absolute trust is placed in the cryptographic algorithms and their mode of implementation [26]. However, this is achieved at the cost of deviating from the unplanned nature of ad-hoc networks to a semi-organised one. In contrast, pure ad-hoc networks are not based upon any such assumptions, and hence permit rapid establishment of the network without any extraneous requirements. These networks are formed in a spontaneous and self organised manner without necessitating a physical or virtual infrastructure. Any wireless node can gain or lose the membership of the network at any time, without the need for prior registration [83].

According to Mayer et. al. [59] trust is defined as:

> "The willingness of a party to be vulnerable to the actions of another party based on the expectation that the other party will perform a particular action important to the trustor, irrespective of the ability to monitor or control the party".

Jøsang [43] defines:

> "Trust in a passionate entity (human) as the belief that it will behave without malicious intent and trust in a rational entity (system) as the belief that it will resist malicious manipulation".

Trust in entities is based on the fact that the trusted entity will not act maliciously in a particular situation. As no one can ever be absolutely sure of this fact, trust is solely dependent on the belief of the trustor. The derivation of trust may be due to direct trust, based on previous similar experiences with the same party, or indirect trust, based on recommendations from other trusted parties [96]. Trust is also time dependent, it grows and decays over a period of time. A pure ad-hoc network closely resembles this human behaviour model, where a number of people or nodes that have never met each other, are able to communicate with each other based on mutual trust levels developed over a period of time. According to Denning [20]:

> "Trust cannot be treated as a property of trusted systems but rather it is an assessment based on experience that is shared through networks of people".

As in real life, trust levels are determined by the particular actions that the trusted party can perform for the trustee. Similarly trust levels can be computed based on the effort that one node is willing to expend for another node. This effort can be in terms of battery consumption, packets forwarded or dropped or any other such parameter that helps to establish a mutual trust level. A trust model that is based on experience alone may not be secluded from attacks in an ad-hoc network but it can identify routes with a certain measure of confidence.

AODV [72], DSR [41] and TORA [70] are three well known reactive routing protocols which are undergoing wide ranging active research. These protocols have been developed for networks where all nodes can faithfully execute them in a munificent

manner. However, in real life such an altruistic stance is difficult to achieve, and so these protocols are more than often executed by nodes that divert from the basic requirements of participation [84]. In order to maintain the impromptu nature of ad-hoc networks, without making any extraneous assumptions, in this thesis, we apply a trust-based scheme to protect these routing protocols. In the scheme all nodes in the network independently execute a distributed trust model and maintain their own assessment concerning other nodes in the network [3]. Each node, based upon its individual experiences, rewards collaborating nodes for their benevolent behaviour and penalises malicious nodes for their malevolent conduct. Most of the events that are experienced by a node occur within the vicinity of its immediate neighbours, which helps to establish direct trust relationships between the neighbours. This trust is then associated with the routing process to form routes that can evade malicious nodes in the network with a high probability of success.

1.1 Thesis Aims and Scope

The thesis of this research is summarised by the statement:

> Trust can be established and maintained with the aim of improving the routing performance in a mobile ad-hoc wireless network, which is operating in an altruistic or antagonistic environment, without the need for a trusted third party either in the establishment or operational phase of the network.

The scope of the thesis can be summarised in the following four stages:

Review: This stage involves the evaluation of existing security schemes, which have been implemented to secure the routing process in an ad-hoc network. A review of different trust models, being used for distributed environments, will also be carried out in order to ascertain their aptness for use in ad-hoc networks.

Development of a Distributed Trust Model: This stage involves the development of the critical components of the distributed trust model.

Design of Trusted Routing Protocols: In order to determine the efficacy of the developed trust model, the model will be integrated with contemporary routing protocols.

Testing and Analysis: The trusted routing protocols will be simulated in an appropriate network simulator and the results obtained will be compiled and interpreted for analysis.

1.2 Thesis Contributions

The primary scientific contributions made by this thesis are summarised as follows:

Development of a Distributed Trust Model: A distributed trust model has been developed, which can derive, quantify and compute trust using the available knowledge in the network. The model also facilitates sharing of reputations among different network nodes using an independent or integrated reputation exchange mechanism.

Integration with Reactive Routing Protocols: The trust model has been integrated with three contemporary reactive routing protocols. Trust derivation of the model has been specially adapted to facilitate seamless integration with each routing protocol.

Performance Analysis of Trusted Routing Protocols: Performance analysis of the trusted routing protocols in an attacked ad-hoc network has been carried out. We have evaluated the performance of these protocols under active attacks and have monitored their inherent robustness against them. With the help of extensive simulations we have also presented results, which highlight the performance of each trust-based routing protocol in a network under

varying traffic loads, mobility conditions and attack patterns.

1.3 Thesis Outline

The thesis is structured as follows:

Related Work (Chapter 2): This chapter presents related previous work, which has been carried out to establish and employ trust in networks. These works include trust establishment through a variety of means including cryptographic and experience based mechanisms.

Reactive Routing Protocols (Chapter 3): This chapter explains the working of three commonly used reactive routing protocols. The different security vulnerabilities, in each of these protocols, are also highlighted.

Secure Routing Protocols (Chapter 4): This chapter discusses some routing protocols that use cryptographic mechanisms to secure the routing process in mobile ad-hoc wireless networks. A comparison of these routing protocols is also presented in the chapter.

The Trust Model (Chapter 5): The different components of the trust model are explained in this chapter. Reputation exchange mechanisms to share trust values among nodes have also been proposed.

Trusted Routing Protocols (Chapter 6): The trust model is applied to the three reactive routing protocols in this chapter. Trust categories are derived from each protocol, which are subsequently used to compute trust in different nodes. Trust application to the routing process is also elaborated.

Results and Analysis (Chapter 7): The performance of the trust-based reactive routing protocols is evaluated in this chapter. A variety of attacks are simulated and the performance of each protocol is monitored through a number of different tests.

Conclusions (Chapter 8): This chapter offers some concluding remarks and examines the possibility for future extensions.

1.4 Formatting Conventions

The formatting conventions followed in this thesis are explained as below.

Node n: Gives the identity of Node n.

`PACKET TYPE`: Represents a type of packet.

ROUTE TYPE: Signifies a specific type of route.

Structure: Represents a structure.

Fields and Flags: Symbolises a field or flag in a structure or packet.

Cases and Events: Represents a specific case or an event.

1.5 Terminology

The following terms are used extensively in this thesis.

Node: A computation device with a wireless transceiver.

Network: A set of interconnected nodes.

Routing Protocol: A protocol used by a node to determine the appropriate path over which packets are sent.

Route: The path taken by a packet to traverse the network as determined by the routing protocol.

Forwarding: The process of sending a received packet to another node in accordance with the routing protocol.

Direct Trust: The trust developed based upon direct interactions with another node.

Indirect Trust: The direct trust values exchanged between indirect nodes also known as reputation.

Aggregate Trust: A non-linear combination of the direct and indirect trust values.

Broadcast: Flooding of a packet in the network.

MAC: Abbreviation for Media Access Control, which is responsible for moving packets to and from a shared channel.

Chapter 2

Related Work

Related work focussing on trust computation, modelling and application is presented in the following sections.

2.1 Distributed Trust Model

The Distributed Trust Model [95] makes use of a protocol to exchange, revoke and refresh recommendations about other entities. By using a recommendation protocol each entity maintains its own trust database. This ensures that the trust computed is neither absolute nor transitive. The model uses a decentralised approach to trust management and uses trust categories and values for computing different levels of trust. The integral trust values vary from -1 to 4 signifying discrete levels of trust from complete distrust (-1) to complete trust (4). Each entity executes the recommendation protocol either as a recommender or a requestor and the trust levels are computed using the recommended trust value of the target and its recommenders. The model has provision for multiple recommendations for a single target and adopts an averaging mechanism to yield a single recommendation value. The model is most suitable for determining trust relationships in provisional and temporary environments and does not specifically target ad-hoc networks. Moreover,

as it requires that recommendations about other entities be passed, the handling of false or malicious recommendations has to be supported via some out-of-band mechanism.

2.2 Distributed Public-Key Model

The Distributed Public-Key Model [120] makes use of threshold cryptography [21] to distribute the private key of the Certification Authority over a number of servers. A (n,*t+1*) scheme allows any *t+1* servers out of total of n servers to combine their partial keys to create the complete secret key. Similarly, it requires that at least *t+1* servers must be compromised to acquire the secret key. The scheme is quite robust but has a number of factors that limit its application to pure ad-hoc networks. Primarily it requires an extensive pre-configuration of servers and a distributed central authority, secondly the *t+1* servers may not be accessible to any node desiring authentication and lastly asymmetric cryptographic operations are known to drain precious node batteries.

2.3 PGP Model

In the Pretty Good Privacy Model [29] all users act as independent certification authorities and have the capability to sign and verify keys of other users. PGP breaks the traditional central trust authority architecture and adopts a decentralised 'web of trust' approach. Each individual signs the keys of all other users in order to build a set of virtual interconnecting links of trust. PGP attaches various degrees of confidence levels from 'undefined' to 'complete trust' to the trustworthiness of public-key certificates and four levels of trustworthiness of introducers from 'don't know' to 'full trust'. Based on these trust levels, the user computes the trust level of the desired party. PGP is suitable for wired networks where a central key server can maintain a database of keys. However, in ad-hoc networks, creation of a central

key server creates a single point of failure and also requires uninterrupted access to the nodes. The other option, as in PGP, is where each node stores a subset of the public keys of other users using a subset of the trust graph [36] and merges these graphs with graphs of other users in order to discover trusted routes. This scheme involves extensive computation and memory requirements and is deemed limiting for ad-hoc networks [94].

2.4 Resurrecting Duckling Model

The Resurrecting Duckling Model [103, 102] is based upon a hierarchical graph of master-slave relationships. The slave (duckling) considers the first node that sends it a secret key through a secure channel as its master (mother duck). The slave always obeys the master and gets all instructions and access control lists from its master. The slave further becomes a master to other devices with whom it can share a secret key through secure means. This master-slave bond can only be broken either by a master, a timeout, or an event, after which the slave is no longer bonded and looks for another master. This model is most suitable for security in large-scale sensor nodes where pre-configuration has to be avoided. As this model demands a hierarchical security chain it is not appropriate for application to ad-hoc networks.

2.5 Watchdog and Pathrater

The Watchdog and Pathrater mechanism [58] has been specifically designed to optimise the forwarding mechanism in the Dynamic Source Routing protocol [41]. The mechanism basically consists of two components: Watchdog and Pathrater. The Watchdog is responsible for detecting selfish nodes that do not forward packets. To do so, each node in the network buffers every transmitted packet for sometime. During this interval, the node places its wireless interface into the promiscuous

mode in order to overhear whether the next node has forwarded the packet or not. The Pathrater assigns different rating to the nodes based upon the feedback that it receives from the Watchdog. These ratings are then used to select routes consisting of nodes with the highest forwarding rate. The range of the ratings vary from 0.0 to 0.8, where 0.5 signifies a node as neutral. These values are updated periodically by 0.01 each 200 ms. During route selection, these rating are averaged over all nodes present in a particular path and the route with the maximum rating is selected. The Watchdog and Pathrater mechanism is only applicable to the DSR protocol, with the route selection criteria being simply determined by the packet forwarding rate.

2.6 CONFIDANT

CONFIDANT (Cooperation Of Nodes, Fairness In Dynamic Ad-hoc NeTworks) [9] adds a trust manager and a reputation system to the Watchdog and Pathrater scheme. The trust manager evaluates the events reported by the Watchdog (monitor in this case) and issues alarms to warn other nodes regarding malicious nodes. The alarm recipients are maintained in a friends-list, which is configured through a user-to-user authentication mechanism [103]. To verify the source of alarms, a mechanism similar to PGP is employed. The reputation system maintains a blacklist of nodes at each node and shares them with nodes in the friends-list. The CONFIDANT protocol implements a punishment based scheme by not forwarding packets of nodes whose trust level drops below a certain threshold. The CONFIDANT protocol suffers from the same problems as that of the Watchdog and Pathrater mechanism. However, the protocol shows improved performance over the Watchdog and Pathrater mechanism through the use of reputations.

2.7 CORE

CORE (COllaborative REputation) [61] is similar to CONFIDANT, however it employs a complicated reputation exchange mechanism. CORE divides the reputation of a node into three distinct components: Subjective Reputation, which is observed through its own observations; Indirect Reputation, which is a positive report by another node and Functional Reputation, which is based upon behaviour monitored during a specific task. These reputations are weighted for a combined reputation value. This combined reputation value is used to make decisions regarding the inclusion or isolation of another node. CORE makes use of two types of entities, a requestor and one or more providers, to support a collaborative reputation mechanism. The requestor asks the providers for reputation values and validates the obtained results with the expected results that have been derived using the Watchdog. Positive trust ratings are exchanged while the negative ratings are locally derived using the Watchdog. CORE also inherits the same problems from the Watchdog and Pathrater mechanism but shows improved performance over the CONFIDANT protocol through the use of extensive reputations [66].

2.8 Terminodes

The Terminodes project [10, 11] makes use of a virtual currency called nuglets, which serves as a payment per forwarded packet. The nuglets are maintained by each node in a tamper resistant security module. The project uses a cryptographic infrastructure to ensure accuracy in transactions and avoid misuse of nuglets. The number of nuglets, held by a node, increase with every forwarded packet and decrease with each originated packet. The project endeavours to encourage forwarding by introducing two charging models: Packet trade model (Recipient to pay) and Packet purse model (Sender to pay). In the first model, each intermediate node has to purchase the packet from the sender of the packet. This increases the overall price

of a packet, which has to be paid by the destination. The advantage of this model is that the originator of a packet does not need to know in advance, the exact number of nuglets required to reach a particular destination, and can so send the packet for free. The obvious disadvantage here is that it doesn't stop malicious nodes from superfluous flooding. In the Packet Purse Model, the sending node has to load each packet with sufficient nuglets, so that it reaches the destination. During transit of the packet, each intermediate node supposedly takes one nuglet out of the packet as its forwarding fee. The advantage of this model is that it is resilient to flooding as the number of nuglets are limited in each packet. The disadvantage of this scheme is that the sender has to precisely know the number of nuglets that are required to be loaded into each transmitted packet and to ensure that the intermediate nodes don't overcharge during the forwarding mechanism.

2.9 Summary

The aforementioned schemes implement various mechanisms to improve upon the routing mechanism in an ad-hoc network. The use of a cryptographic infrastructure and special hardware devices, although help in establishing trust, are considered an extraneous requirement that contradicts the spontaneous nature of ad-hoc networks. Some schemes have certain pre- and post-establishment conditions, which restrict their application only to managed ad-hoc wireless networks. In addition, most of the previous work only takes into account the measure of the forwarding mechanism by network nodes. However, we accentuate that such a measure is not only inadequate for trust computation but is also vulnerable to deception. For example, any malicious node may fallaciously modify the packet contents during forwarding and still be rewarded with a positive rating by the trust model. In the next chapter we examine the working of some contemporary reactive routing protocols and highlight the possible vulnerabilities in them.

Chapter 3

Reactive Routing Protocols

In wireless ad-hoc networks, the nodes act like mobile IP routers and carry out basic functions like packet forwarding, routing and network management. Due to a number of factors including dynamic topology, energy constraints and unidirectionality of links, standard inter-router protocols cannot be immediately applied to mobile ad-hoc wireless networks. In addition, as the routers are moving most of the time, the network lacks traffic concentration points, a basic requirement of standard routing protocols. Many different types of routing protocols have been specifically developed for ad-hoc networks, out of which a number of reactive routing protocols are undergoing active research with the Internet Engineering Task Force (IETF) [38], primarily due to their improved performance and efficiency over proactive routing protocols.

All routing protocols are developed for altruistic environments and hence become susceptible to attacks when executed under antagonistic conditions. Each protocol, based upon its construction and modus operandi, is vulnerable to certain attacks. In Section 3.1, 3.2 and 3.3, we explain the AODV, DSR and TORA reactive routing protocols respectively and also highlight their inherent vulnerabilities.

3.1 Ad-hoc On-Demand Distance Vector Routing Protocol

The Ad-hoc On-Demand Distance Vector (AODV) protocol [72] is inherently a distance vector routing protocol that has been optimised for ad-hoc wireless networks. It is an on-demand protocol as it finds the routes only when required and is hence also reactive in nature. AODV borrows basic route establishment and maintenance mechanisms from the Dynamic Source Routing (DSR) protocol and hop-to-hop routing vectors from the Destination-Sequenced Distance-Vector (DSDV) routing protocol [73]. To avoid the problem of routing loops, AODV makes extensive use of sequence numbers in control packets. The AODV protocol utilises three distinct control packets for route discovery and maintenance i.e. `ROUTE REQUEST`, `ROUTE REPLY` and `ROUTE ERROR`. The structure of these packets, as specified by the IETF, is given in Appendix A.

When a source node intends communicating with a destination node, whose route is not known, it broadcasts a `ROUTE REQUEST` packet. Each `ROUTE REQUEST` packet contains an ID, source and the destination node IP addresses and sequence numbers together with a hop count and control flags. The ID field uniquely identifies the `ROUTE REQUEST` packet; the sequence numbers indicate the freshness of control packets and the hop-count maintains the number of nodes between the source and the destination. Each recipient of the `ROUTE REQUEST` packet that has not seen the Source IP and ID pair or doesn't maintain a fresher (with larger sequence number) route to the destination rebroadcasts the same packet after incrementing the hop-count. Such intermediate nodes also create and preserve a Reverse Route to the source node for a certain interval of time. When the `ROUTE REQUEST` packet reaches the destination node or any node that has a fresher route to the destination, a `ROUTE REPLY` packet is generated and unicast back to the source of the `ROUTE REQUEST` packet. Each `ROUTE REPLY` packet contains the destination sequence number, the source and the destination IP addresses, route lifetime together with a hop count

and control flags. Each intermediate node that receives the ROUTE REPLY packet, increments the hop-count, establishes a FORWARD ROUTE to the source of the packet and transmits the packet on the REVERSE ROUTE. For preserving connectivity information, each node executing AODV makes use of periodic HELLO messages to detect link breakages to nodes that it considers as its immediate neighbours. In the event a link break is detected, for a next hop of an active route, a ROUTE ERROR message is sent to its active neighbours that were using that particular route.

In order to facilitate multi-path support in AODV a number of extensions have been proposed [51, 53]. AOMDV [56] is one such extension that provides loop-free and disjoint alternate paths. During route discovery, the source nodes broadcast a ROUTE REQUEST packet that is flooded through the network. In contrast to AODV, each recipient node creates multiple REVERSE ROUTEs while processing the ROUTE REQUEST packets that are received from multiple neighbours. However, care is taken to verify loop freedom and path disjointness while creating these REVERSE ROUTEs. Similarly, one or more ROUTE REPLY packets are generated via a loop free path, by the destination or any node having a route to the destination, in response to each received ROUTE REQUEST packet. These ROUTE REPLY packets when received by the source or intermediate nodes, result in creation of multiple FORWARD ROUTEs leading to the same destination. To ensure loop freedom and path disjointness, AOMDV requires a minor modification to the AODV ROUTE REQUEST and ROUTE REPLY packet headers, so as to pass the last hop information that acts like a unique path identifier. AOMDV uses this information to ensure loop freedom while creating multiple paths to a single destination. The AOMDV protocol guarantees path or link disjointness if any two paths to a single destination have unique next hops as well as unique last hops. Similarly, AOMDV assures loop freedom by not preserving multiple paths to a single destination, which have the same destination sequence number. AOMDV offers two types of disjoint paths: Node Disjoint and Link Disjoint. In node disjoint paths, there are no common nodes, each multiple path is formed using distinct nodes. In link disjoint paths, there may be a common node

involved but there is no common link between any two nodes used in two different paths.

AOMDV provides the following three additional benefits over AODV:

- Low overhead of inter-node coordination
- Computes disjoint routes via a distributed protocol without using source routing
- Determines alternate paths with minimal packet overhead

The AODV and AOMDV protocols are both based upon the distance vector algorithm, and so involve maximum interaction with their immediate neighbours without reliance on any global knowledge. This limited perspective allows malicious nodes to mount a variety of modification attacks against ignorant nodes i.e., ordinary nodes that are not aware of the overall topology. The major vulnerabilities present in both AODV and AOMDV protocols are described as follows:

Deceptive incrementing of Sequence Numbers: Destination sequence numbers determine the freshness of a route. The destination sequence numbers maintained by different nodes are only updated when a newer control packet is received with a higher sequence number. Normally the destination sequence numbers received via control packets cannot be greater than the previous value held by the node plus one [107]. However, malicious nodes may increase this number, so as to advertise fresher routes towards a particular destination. If this difference is equal or larger than two from the last known sequence number for the same node, then there is a high probability that the network may be under a modification attack.

Deceptive decrementing of Hop Count: Both the AODV and AOMDV protocols prefer route freshness over shorter route lengths, in that a node prefers a control packet with a larger destination sequence and hop count number than

a control packet with a smaller destination sequence and hop count number. However, if the destination sequence numbers are the same then the route with the least hop count is given preference. Malicious nodes may exploit this mechanism in order to generate fallacious routes that portray minimal hop-counts.

To avoid confusion, we henceforth refer to AOMDV as the AODV protocol, during the scope of this thesis.

3.2 Dynamic Source Routing Protocol

As the name suggests, the Dynamic Source Routing (DSR) [41] protocol uses IP source routing. All data packets that are sent using the DSR protocol contain the complete list of nodes that the packet has to traverse. DSR uses three different types of control packets for route discovery and maintenance i.e. `ROUTE REQUEST`, `ROUTE REPLY` and `ROUTE ERROR`. The structure of these packets, as specified by the IETF, is given in Appendix B.

During route discovery, the source node broadcasts a `ROUTE REQUEST` packet with a unique identification number. The `ROUTE REQUEST` packet contains the address of the target node to which a route is desired. All nodes that have no information regarding the target node or have not seen the same `ROUTE REQUEST` packet append their IP addresses to the `ROUTE REQUEST` packet and re-broadcast it. In order to control the spread of the `ROUTE REQUEST` packets, the broadcast is done in a non-propagating manner with the IP Time-To-Live (**TTL**) field being incremented in each route discovery. The `ROUTE REQUEST` packets keep on spreading until they reach the target node or any other node that has a route to the target node.

The recipient node creates a `ROUTE REPLY` packet, which contains the complete list of nodes that the `ROUTE REQUEST` packet had traversed. Based upon implementation, the target node may respond to one or more incoming `ROUTE REQUEST` packets.

Similarly, the source node may accept one or more `ROUTE REPLY` packets for a single target node. The selection of the `ROUTE REPLY` can be made both on minimal hop count or latency. In this thesis, we have used a multi-path version [63] of the DSR protocol in which each `ROUTE REQUEST` packet received by the destination is responded to by an independent `ROUTE REPLY` packet [64].

For optimisation reasons, nodes maintain a PATH CACHE or a LINK CACHE scheme [32]. The former stores complete paths to a particular destination, while the latter only caches information related to individual links. The advantage of the LINK CACHE scheme is that it allows alternate paths to a destination even when some of the intermediate links have failed. All nodes either forwarding or overhearing data and control packets, add all useful information to their respective cache. This information is used to limit the spread of control packets for subsequent route discoveries. The DSR protocol also provides the facility for 'route shortening', known as a Gratuitous Route, to avoid redundant intermediary nodes in a route.

The DSR protocol is susceptible to the following attacks:

Routing Black Hole: In this attack an attacker sends forged routing packets, so as to make other nodes route traffic to a particular node. This node now has the ability to monitor all traffic flow and to selectively modify or drop packets at own discretion.

Gratuitous Detour: In the Gratuitous Detour attack [34], the attacker attempts to make a route through itself appear legitimate by adding virtual nodes to the route in spite of the fact that shorter usable routes may be existent in the network.

Deceptive alteration of IP addresses: During the propagation of the `ROUTE REQUEST` packet, intermediate nodes add their IP addressees to it for route creation. However, any malicious node may modify, delete or add IP addressees to create routes as per its own requirement. Doing so enables malicious nodes

to launch a variety of attacks in the network including creation of wormholes and black holes.

Deceptive alteration of Hop Count: The hop count field of the IP packet usually informs the recipient of the total number of hops that the packet has traversed so far. Consequently, malicious nodes may increase this number, so as to portray longer routes or decrease it to represent shorter routes. By doing so, a malicious node is able to degrade or upgrade routes, thereby creating a topology that is most favourable to it.

3.3 Temporally Ordered Routing Algorithm Protocol

The Temporally Ordered Routing Algorithm (TORA) protocol [70] is a distributed routing protocol for multi-hop networks. The unique feature of this protocol is that it endeavours to localise the spread of routing control packets. The protocol is essentially an optimised hybrid of the Gafni Bertsekas (GB) protocol [28] and the Lightweight Mobile Routing (LMR) protocol [15]. It guarantees loop freedom, multiple routes and minimal communication overhead even in highly dynamic environments. The protocol attempts to minimise routing discovery overhead, and in doing so prefers instant routes over optimal routes. The protocol supports source-initiated on-demand routing for networks with a high rate of mobility as well as destination oriented proactive routing for networks with lesser mobility.

TORA maintains state on a per-destination basis and runs a logically separate instance of the algorithm for each destination. TORA assigns directional heights to links, so as to direct the flow of traffic from a node with a greater height to another with a lower height. The significance of these heights, which are assigned based on the direction of a link towards the destination, is that a node may only forward

packets downstream but not upstream, i.e. to another node that has a higher, undefined or unknown height. The height is represented by a quintuple (τ, oid, r, δ, i) where the first three values represent a reference level and the last two represent the change with respect to the reference level. Each time a node loses its downstream link due to a link failure, a new reference level is computed using either a partial or full link reversal mechanism. The values in the height quintuple indicate the following:

τ	Logical time of a link failure
oid	Unique ID of the router that defined the reference level
r	Reflection indicator bit
δ	Propagation ordering parameter
i	Unique ID of the router

In the on-demand mode, TORA performs three routing functions: Route Creation, Route Maintenance and Route Erasure. To accomplish these functions it uses three distinct control packets: Query (`QRY`), Update (`UPD`) and Clear (`CLR`). The structure of these packets, as specified by the IETF, is given in Appendix C.

During route discovery, a source node requiring a route to a destination, broadcasts a `QRY` packet containing the destination address. The `QRY` packet is propagated through the network until it reaches the destination or any intermediate node possessing a route to the intended destination. The recipient of the `QRY` packet broadcasts an `UPD` packet that lists its height with respect to the destination. If the destination itself replies to a `QRY` packet it sets the height to zero in the `UPD` packet. Each node, receiving the `UPD` packet, sets its own height greater than the height specified in the `UPD` packet. This results in creation of a directed acyclic graph (DAG) with all links pointing in the direction of the destination as the root.

In the proactive mode, routes are created using the `OPT` packet that is sent out by the destination. The `OPT` packet, which is similar to the `UPD` packet, also consists of a sequence number for duplication avoidance. Each recipient nodes adjusts its

height data structure, and sends out a `OPT` packet to neighbouring nodes.

When a node discovers that it has no downstream links, either due to a link failure or to a link reversal, it modifies its height based upon five predefined cases. The first case (*GENERATE*) is applicable when there are no downstream links due to a link failure, in which the node defines a new reference level. All other cases (*PROPAGATE*, *REFLECT*, *DETECT* and *GENERATE*) are executed by nodes having no downstream links due to a link reversal. These cases define and propagate new reference levels, upon reception of `UPD` packets, based on different criteria as shown in Fig. 1 [71].

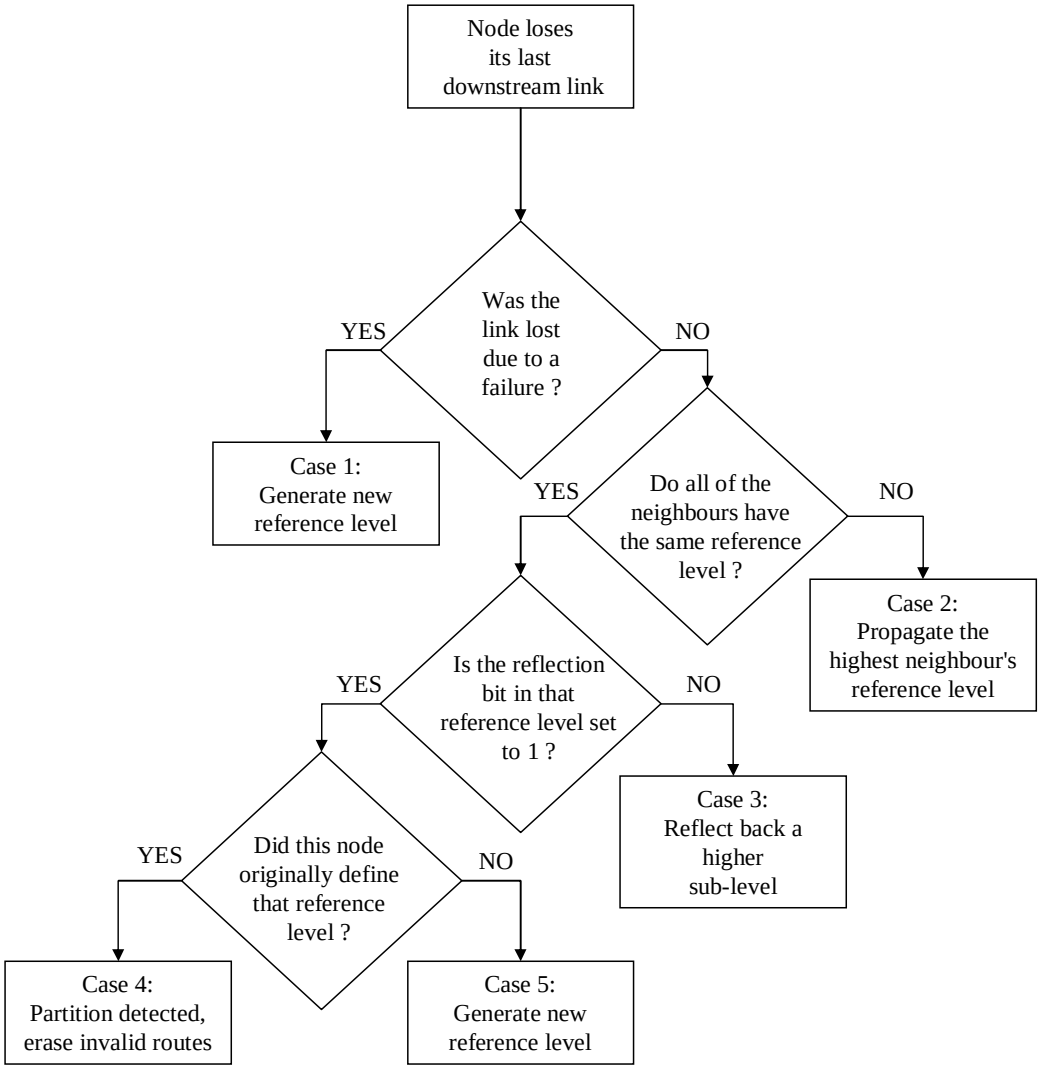


Figure 1: TORA Route Maintenance Tree

Whenever a partition is spotted through a *DETECT* case, the node sets its own and the height of all its neighbours to NULL and broadcasts a `CLR` packet. The neighbouring nodes that receive the `CLR` packet, based upon the reference level, also set the heights in a similar manner and rebroadcast the `CLR` packet. In this way the height of each node in the portion of the network that is partitioned is set to NULL and all invalid routes are erased.

As each node in TORA maintains multiple DAGs to the destination, therefore, in any network with an average n number of nodes each with $\frac{n}{2}$ downstream neighbours, a node could still effectively communicate with the destination node upon link failure of $\frac{n}{2} - 1$ nodes. However, to sustain this redundancy, each node maintains a height data structure, link status along with a number of state and auxiliary variables for each destination node.

TORA is not a standalone routing protocol but requires the services of the Internet MANET Encapsulation Protocol (IMEP) [17]. IMEP has been designed as a network layer protocol that provides link status, reliable in-order delivery of control packets, neighbour connectivity information, address resolution and other services to Upper Layer Protocols (ULP). IMEP endeavours to reduce overhead by aggregating a number of control packets into blocks. These blocks are transmitted with a sequence number and a list of nodes who have not yet acknowledged the receipt of the block. All such nodes upon receipt of the same block acknowledge the receipt. However, if after a certain number of retries the block is still not acknowledged, the link to that node is considered down and this information is conveyed to the ULP. Similarly, IMEP declares a link to be broken if no beacons are exchanged within the maximum beacon interval [8].

Two of the common attacks specific to the TORA protocol are:

Deceptive alteration of `QRY` and `UPD` Packets: During propagation of the `QRY` and `UPD` packets, any intermediate nodes may modify the destination IP address or the height quintuple, so as to create fallacious directed acyclic graphs.

Doing so enables malicious nodes to create deceptive paths and routing loops.

Malicious propagation of `CLR` packets: Malicious nodes may propagate illegitimate `CLR` packets, so as to erase genuine routes. By doing so, a malicious node is able to render inoperative certain critical routes and launch other attacks against the network.

3.4 Summary

Reactive routing protocols enable mobile nodes to institute operational routes in the network in an on-demand manner. These protocols conserve battery power by only discovering routes when specifically solicited. During route discovery, requests for routes to the destination are flooded in the network. These requests are responded to by route replies, which help to establish routes in the network. In order to ensure serviceability of the routes, error packets may be originated upon link severing to inform all concerned regarding a faulty route.

Each protocol has certain vulnerabilities, which may be exploited by a malicious node to cause routing predicaments. Most of the vulnerabilities experienced are due to the lack of integrity control during the routing process. In order to obtain effective routing, it is imperative that only benevolent nodes, making legitimate modifications to the packets, are involved into the routing process. In the next chapter, we discuss some secure routing protocols, which realise different security services in the network including integrity, confidentiality and authentication with the help of cryptographic mechanisms.

Chapter 4

Secure Routing Protocols

Ad-hoc network routing protocols usually operate at the second or third layer of the TCP/IP protocol suite. Generally speaking, a typical routing protocol works in the following manner. The source node, which requires a route to a destination, broadcasts a `ROUTE REQUEST` packet. Each intermediate recipient node retransmits this packet, if it is not a duplicate. When the packet reaches the destination, it originates a `ROUTE REPLY` packet that is unicast towards the sender. For route maintenance these protocols use `ROUTE ERROR` packets, which inform the active users regarding a route failure. In order to minimise route discovery broadcasts, the intermediate nodes, while forwarding or overhearing routing or data packets, may add observed routing information to locally maintained routing tables or cache.

The core security problems that affect ad-hoc networks, originate due to the route development by the intermediate nodes. In wired networks, usually all traffic originating of a particular subnet is routed towards a known trusted gateway. That trusted gateway usually forwards the traffic to its own trusted gateway, and so on until the traffic reaches its destined subnet. However, in case of ad-hoc networks, as the nodes are mobile, there is no single traffic concentration point and no known trust entity. Therefore, in order to achieve connectivity, the neighbouring and intermediate nodes have to be frequently trusted for their truthful execution.

In Section 4.1, we first highlight the issues that need to be considered before employing any security scheme to ad-hoc networks. We then explain in detail seven contemporary secure routing protocols in Section 4.2. This is followed by their comparison being presented in Section 4.3.

4.1 Security Issues

Some of the issues that need to be addressed before implementing any security scheme in ad-hoc networks are:

- Energy is one of the greatest constraints to a node's capabilities [114].
- Symmetric encryption/decryption algorithms and hashing functions consume minimal computational energy in comparison to public key algorithms [86].
- Transmission energy consumption is over three orders of magnitude greater than the energy consumption for encryption and hashing [12].
- It is problematic to establish and sustain a trusted third party in an ad-hoc network [82].
- Inter-node relationships are usually less formal, temporary and short-term [113].
- There are three types of adversaries that an ad-hoc network may have to deal with: malicious, compromised and selfish [88].

Nodes in a wireless ad-hoc network are extremely dependent upon efficient utilisation of their battery packs. Undue usage due to extra transmissions or computing can result in rapid battery draining. A less vivid but understated malicious behaviour is node selfishness in which nodes, in order to conserve their batteries, may be tempted to not relay packets. An easy solution against such attacks is the establishment of a trusted third party that can facilitate building of trust relationships

among communicating nodes. Public Key Infrastructure (PKI) is an effective way of establishing trust but is deemed unsuitable because it makes use of asymmetric cryptographic algorithms that have been known to be a target of Denial of Service attacks. Also, maintaining a centralised or distributed repository of certificates is itself a potential target of attack.

In a mobile ad-hoc network, as the nodes are constantly leaving and entering the network, the relationships between the nodes tend to be short lived and reciprocal in nature. Since the time span of such relationships is acute, a benevolent behaviour is expected of all nodes. However, in addition to benevolent nodes, there are also malicious and compromised nodes present in the network. Malicious nodes attempt to eavesdrop, replay, distort and impersonate messages while a compromised node is a benevolent node that has been taken over by an adversary. Compromised nodes produce valid signatures, identification and certificates and are hence very difficult to isolate. Selfish nodes consume network resources, such as the battery power and computation facilities of the correctly functioning nodes, but are unwilling to contribute their own resources to the ad-hoc network community.

4.2 Secure Routing Protocols

Many MANET routing protocols have been designed without having security in mind, based upon the naive assumption that all the nodes in the network are friendly. The actual scenario is quite different from this assumption as in a MANET a hostile node may join the network at any time as a friend, or any friendly node may become hostile. This malicious or hostile node can disrupt the network routing services and, in the worst-case scenario, cause a complete failure of the network communications.

The provisioning of security services to MANETs face a totally different set of problems. These problems include the lack of security in the medium, energy and processing constraints and poor physical protection of the nodes. Nonetheless, the

absence of a fixed infrastructure in MANET context ensures that no part of the network is committed to support independently any precise network functionality like routing, neighbour discovery, etc. Additionally the lack of centralised services like Key Distribution Centres (KDC), Certification Authorities (CA) and Name Servers complicate the introduction of security in the perspective of ad-hoc networks.

Routing in MANETs encompasses two major problems. One problem is to ensure that data is routed securely through trusted nodes and the second is the security of the routing protocol messages. In view of the fact that both data and control messages utilise the same wireless transmission medium, routing protocol messages can be modified to alter routing behaviour. This means that if a routing protocol message is altered to generate a false route, then no amount of security on data packets can correct this routing misbehaviour. This means that the security of the routing protocols is imperative for secure routes through the ad-hoc network.

A good secure routing protocol must conform to the following requirements to ensure that the discovered path from source to destination functions properly in the presence of malicious nodes [19]:

- Authorised nodes to perform route computation and discovery
- Minimal exposure of network topology
- Detection of spoofed routing messages
- Detection of fabricated routing messages
- Detection of altered routing messages
- Avoiding formation of routing loops
- Prevent redirection of routes from shortest paths

To achieve the said goals in ad-hoc networks, secure routing protocols use a combination of hash functions and asymmetric or symmetric cryptographic algorithms [81].

Hash functions, also known as one-way functions, take in a variable length input called the pre-image and convert it into a smaller fixed-length output called the hash. The security of this output hash lies in the one-way property of the hash function, which makes it computationally impossible to find the pre-image from the hash in the reverse direction. A hash function is said to be cryptographically strong if it is computationally impossible to find another pre-image for the same hash value, also called a collision. Hash functions are extremely efficient and are generally employed in ad-hoc networks for integrity verification and authentication of data and control packets.

Asymmetric algorithms, allow nodes to encrypt network traffic using public keys, so that the data is only recoverable by the possessor of the corresponding private key. The private key is kept secure by every node and the public keys are distributed using trusted third party servers or mutual sharing techniques [120]. Cryptosystems based on asymmetric algorithms use a variety of NP hard problems including composite number factorisation and discrete log problem [60], making them very secure and reliable. However, the actual implementation of such algorithms incurs colossal computation requirements and is generally deemed infeasible for ad-hoc network environments.

In contrast, symmetric algorithms make use of a single key for encryption and decryption, and are relatively a thousand times faster than asymmetric cryptographic algorithms. Hash functions, when used in conjunction with symmetric cryptographic algorithms prove extremely beneficial for securing the routing process and data in ad-hoc networks. Moreover, as the hashing process is extremely efficient and secure, it is used to counter a variety of attacks launched against the availability of nodes in an ad-hoc network.

In the following sub-sections, we discuss some contemporary secure routing protocols, which make use of cryptographic algorithms to protect the routing process in an ad-hoc network:

4.2.1 ARAN

The Authenticated Routing for Ad-hoc Networks (ARAN) [19] secure routing protocol is an on-demand routing protocol that identifies and shields against malevolent actions by malicious nodes in the ad-hoc network environment. ARAN relies on the use of digital certificates and can successfully operate in the managed-open scenario where no network infrastructure is pre-deployed, but a small amount of prior security coordination is expected. ARAN provides authentication, message integrity and non-repudiation in ad-hoc networks by using a preliminary certification process that is followed by a route instantiation process that guarantees end-to-end provisioning of security services.

ARAN requires the use of a trusted certificate server (T). All nodes are supposed to keep fresh certificates with the trusted server and should know T's public key. Prior to entering the ad-hoc network, each node has to apply for a certificate that is signed by T.

$$\mathrm{T} \rightarrow \mathrm{A} : \mathrm{cert_A} = [\mathrm{IP_A}, \mathrm{K_{A+}}, \mathrm{t}, \mathrm{e}]\mathrm{K_{T-}}$$

The certificate contains the IP address of the node ($\mathrm{IP_A}$), its public key ($\mathrm{K_{A+}}$), a timestamp of when the certificate was generated (t) and a time at which the certificate expires (e), along with the signature by T ($\mathrm{K_{T-}}$).

ARAN accomplishes the discovery of routes by a broadcast route discovery message from a source node, which is replied to in a unicast manner by the destination node. All the routing messages are authenticated at every hop from the source to the destination, as well as on the reverse path from destination to source. Each node along the path performs the following steps:

- Validates the previous node's signature
- Removes the previous node's certificate and signature
- Records the previous node's IP address

- Signs the original contents of the message
- Appends its own certificate
- Forward broadcasts the message

ARAN also provides similar measures for authenticated route set-up, route maintenance and key revocation through the use of certificates.

4.2.2 ARIADNE

ARIADNE [34] is an on-demand secure ad-hoc routing protocol based on the DSR protocol that protects against node compromise and relies only on extremely efficient symmetric cryptography. The security of ARIADNE is based upon the secrecy and authenticity of keys that are kept at the nodes. ARIADNE prevents a large number of Denial-of-Service attacks from malicious or compromised nodes.

ARIADNE provides assurance that the target node of a route discovery process can verify the initiator, that the initiator can verify each transitional node, which is on the path to the destination present in the `ROUTE REPLY` message, and that no intermediate node can reduce the node list in the `ROUTE REQUEST` or `ROUTE REPLY` messages. Route Discovery is performed in two stages: the Initiator floods the network with a `ROUTE REQUEST` that solicits a `ROUTE REPLY` from the Target. During route discovery the Target authenticates each node in the node list of the `ROUTE REQUEST` and the Initiator authenticates each individual node in the node list of the `ROUTE REPLY`. For node authentication, ARIADNE has three alternative techniques i.e. TESLA (Timed Efficient Stream Loss tolerant Authentication), Digital Signatures, or pair-wise shared secret keys.

TESLA [74] is a broadcast authentication protocol for authenticating messages. It is different from traditional asymmetric protocols as it achieves its asymmetry from clock synchronisation and delayed key disclosure. When used for authentication, each sender chooses a random key and generates a one-way key chain by repeatedly

computing a one-way hash function on the initial key [115]. The sender publishes each key of the key chain at a pre-determined key disclosure interval in the reverse order from its generation. The published key remains secret for the next key disclosure delay interval. The sender computes the TESLA key using a pseudo-random function on the hash key chain. This key is used to compute the Message Authentication Code (MAC) for any packet to be transmitted. When the receiver receives a packet, it buffers it until the sender discloses the required element of the key chain. To authenticate any received value on the one-way chain, a node verifies it by hashing it a number of times to determine if the computed value matches a previous known authentic key on the chain. If the key chain element is authentic then the recipient computes the corresponding TESLA key using the pseudo-random function on the hash key chain, calculates the MAC and compares the result with the MAC in the received packet. If the MAC match, it ensures the authenticity of the packet and the sender.

When TESLA is used for authentication, each sender picks a random initial key K_n and creates a one-way hash chain by repeatedly applying a hash function H on the preliminary value:

$$K_{n-1} = H[K_n], K_{n-2} = H[K_{n-1}], ...$$

In general,

$$K_i = H[K_{i+1}] = H^{n-i}[K_n]$$

The one-way hash chain provides the property that another node can only increase a metric in a routing update but cannot decrease it. To verify any obtained value of the one-way chain, a node computes any preceding key K_i from a key K_j, where $i < j$, using the following equation to determine if the computed value equals a previously known authentic key on the chain.

$$K_j = H^{i-j}[K_i]$$

Each key of the key chain is also published by the sender at known intervals in the

reverse order from its creation i.e. $K_0, K_1, ..., K_n$. With TESLA, each hop authenticates the information in the `ROUTE REQUEST` packet during route discovery. The `ROUTE REPLY` is buffered at the target until intermediate nodes release the corresponding TESLA keys. When the TESLA security condition is met at the target, a MAC is included in the `ROUTE REPLY`. When the initiator receives the `ROUTE REPLY` it verifies that the target MAC is valid including certain other verifications. If all the tests are successful the initiating node accepts the `ROUTE REPLY`. When route discovery is performed using digital signatures, the MAC list in the `ROUTE REQUEST` becomes a signature list, where the data that was used to compute the MAC is instead used to compute a signature. Route discovery using MACs is most efficient but it requires that all nodes possess a pair of shared keys. The MAC list in the `ROUTE REQUEST` is computed by means of a key that is shared between the target and the current node. This MAC is authenticated at the target and not returned in the `ROUTE REPLY`. Similarly, during route maintenance, to prevent unauthorised nodes from sending `ROUTE ERROR` packets, ARIADNE requires that the sender authenticate a `ROUTE ERROR`. If this authentication is delayed then each node that is able to authenticate the `ROUTE ERROR` packet, buffers it until the corresponding keys are received.

4.2.3 SAODV

The Secure Ad-hoc On-Demand Distance Vector (SAODV) [118] is an extension of the AODV routing protocol. It can be used to protect the route discovery mechanism of AODV by providing security features like integrity, authentication and non-repudiation. The protocol operates mainly by using new extension messages with the AODV protocol. In these extension messages there is a signature produced by digesting the AODV packet using the private key of the original sender of the routing message.

The SAODV scheme is based on the assumption that each node possesses certified public keys of all network nodes. Ownership of certified public keys enables

intermediate nodes to authenticate all in-transit routing packets. The originator of a routing control packet appends its RSA signature [97] and the last element of a hash chain to the routing packets. As the packets traverse the network, intermediate nodes cryptographically authenticate the signature and the hash value. The intermediate nodes generate the k^{th} element of the hash chain, with k being the number of traversed hops, and place it in the packet. The route replies are supplied either by the destination or intermediate nodes having an active route to the required destination.

The SAODV protocol gives two alternatives for `ROUTE REQUEST` and `ROUTE REPLY` messages. In the first case when a `ROUTE REQUEST` is sent, the sender creates a signature (i.e. encryption using the private key of the sender of all the fields in the AODV packet less the hop count) and appends it to the packet. Intermediate nodes authenticate the signature before creating or updating the Reverse Route to that host. The Reverse Route is stored only if the signature is verified. When this packet reaches the final destination, the node signs the `ROUTE REPLY` with its private key and sends it back. The intermediate and final nodes, again verify the signature before creating or updating a route to that host. The signature of the sender is also stored along with the route entry. The second case is also similar to the first one with the only disparity being that the `ROUTE REQUEST` message has another signature that is always stored along with the Reverse Route. This second signature is used in the regular and `GRATUITOUS ROUTE REPLY`s to future `ROUTE REQUEST`s that the node might reply to as an intermediate node.

4.2.4 SAR

Security-Aware Ad-hoc Routing (SAR) [116] presents a generalised framework for any on-demand secure ad-hoc routing protocol. SAR uses security information to dynamically control the choice of routes installed in the routing tables. SAR enables applications to selectively implement a subset of security services based on a cost-benefit analysis.

SAR requires that nodes having the same trust level must share a common secret key. SAR augments the routing process using hash digests and symmetric encryption mechanisms. The signed hash digests provide message integrity while the encryption of packets ensures their confidentiality. The routes discovered by SAR may not always be the shortest between any two communicating entities in terms of hop-count. However, these routes do have a quantifiable guarantee of security. If there is more than one route that satisfies the required security attributes, then SAR will find the shortest such route. SAR will find the optimal routes if all the nodes on the shortest path can satisfy the security requirements. However, if the ad-hoc network does not have a path with nodes that meet the security requirements, SAR will fail to find a route even if the network is connected. SAR, when implemented on the AODV protocol, adds two additional fields to the `ROUTE REQUEST` packet and one additional field to the `ROUTE REPLY` packet. The first field is the security requirement (**RQ SEC REQUIREMENT**) and is set by the sender. It indicates the preferred level of trust for the path to the destination. This field can be used to carry simple integer values or bit vectors to reflect the existing hierarchies or combinations of different types of security services. For example, a three-bit vector can be used to specify whether the nodes desire to do a simple hash, digital signature, or content encryption over the AODV routing packets. The second field added to the `ROUTE REQUEST` packet is the security guarantee (**RQ SEC GUARANTEE**) that signifies the maximum level of security provided by the discovered paths. **RQ SEC GUARANTEE** is updated at every hop during the route discovery phase. If **RQ SEC REQUIREMENT** has an integer representation then the **RQ SEC GUARANTEE** will be the minimum of all the security levels of the participating nodes in the path. However, if **RQ SEC REQUIREMENT** is represented in bit vectors, **RQ SEC GUARANTEE** will be computed by ANDing the **RQ SEC REQUIREMENT** values of the participating nodes in the path. The value thus computed is copied into the additional **RP SEC GUARANTEE** field of the `ROUTE REPLY` packet and sent back to the sender. The sender can use this value to establish the security level over the entire path. This value indicates the actual

level of trust that the path provides. Thus the sender can either use this security guarantee value or determine whether a more secure connection is required. This value is also copied into the routing tables, of the nodes in the reverse path, to preserve security information with reference to cached paths.

4.2.5 SEAD

The Secure Efficient Distance vector (SEAD) [33] is a proactive secure routing protocol based on the Destination-Sequenced Distance Vector protocol (DSDV) [73]. SEAD is specifically configured for the DSDV-SQ (Sequence number) version of the DSDV protocol. The DSDV-SQ version of the DSDV protocol prevents routing loops that are caused by out of order updates. Routing can be disrupted if a malicious node modifies the sequence number or the metric field of a routing table update message. SEAD deals with such attacks that modify routing information during the update phase of the DSDV-SQ protocol. It also provides measures to thwart replay attacks.

SEAD uses hash chain elements to authenticate the sequence number and metric of a routing table update message. In addition, the receiver of SEAD routing information also verifies the sender, making sure that the routing information originates from the right node. Each node uses a specific authentic element, which is signed from its hash chain, in each routing update that it sends about itself. This initial element of the one-way hash chain offers authentication for the lower bound on the metric in other routing updates for that node. The utilisation of a hash value, analogous to the sequence number and metric in a routing update entry, foils any node from promoting a route to some destination with a greater sequence number. Similarly, a node cannot present a route better than those for which it has already received an advertisement because the metric in an existing route cannot be decremented owing to the one-way nature of the hash chain. When a routing update is received by a node, it verifies the authenticity of the information for each entry in the update by means of the destination address, the sequence number and the

metric of the received entry, along with the most recent prior authentic hash value received from that destination's hash chain. The received elements are hashed the correct number of times to verify the authenticity of the received information. If the computed hash value and the real hash value match, the entry is considered to be authentic and the node processes it in the routing algorithm as a standard received routing update entry; if not, the node disregards the entry and does not adjust its routing table.

4.2.6 SLSP

Secure Link State Routing Protocol (SLSP) [69] provides secure proactive topology discovery and can be used either as a stand-alone protocol or as part of a hybrid routing framework when combined with a reactive protocol. It can operate in networks of recurrently changing topology and memberships. It is resilient against individual attackers and is capable of altering its range between local to network-wide topology discovery.

To function effectively without a central key management authority, SLSP enables each node to periodically broadcast its public key to nodes within its zone. In addition each node also broadcasts signed `HELLO` messages containing its Medium Access Control (MAC) address and IP address (MAC, IP) pair to its neighbours. The distribution of MAC addresses strengthens the scheme by forbidding nodes from spoofing at the data link layer. It also assists in protection against flooding DoS attacks. To achieve these goals, a Neighbour Lookup Protocol (NLP) is made an integral part of SLSP. NLP is responsible for the following tasks:

- Maintaining a mapping of MAC and IP layer addresses of the node's neighbours.
- Identifying potential discrepancies, such as the use of multiple IP addresses by a single data-link interface.

- Measuring the rates at which control packets are received from each neighbour by differentiating the traffic with the help of MAC addresses.

This helps in discarding nodes, which maliciously seek to exhaust network resources by injecting large volume of control packets in the network. SLSP requires that the MAC layer 48-bit address be passed to the network layer. This requires a straightforward alteration of the device driver, so that the data link address is 'passed up' to the routing protocol with every packet. NLP then extracts and maintains the 48-bit hardware source address for each received (overheard) frame, together with the encapsulated IP address. The mappings amid these two addresses are kept in the table until transmissions from the corresponding neighbouring nodes are overheard or the lost neighbour timeout period expires.

Link State updates (LSU) are recognised by the IP address of their originator and a 32-bit sequence number. To ensure that the LSUs only propagate in a zone of its origin, receiving nodes verify if they have the public key of the originating node, except if the key is provided with the LSU. The LSU is then verified, its hop chain updated, **TTL** decremented and rebroadcasted. A LSU is discarded based upon NLP notification or an error in the hash chain. A hash chaining mechanism is used to ensure that the hop counters can be authenticated at each hop.

SLSP employs a round robin servicing mechanism to provide the assurance that benign control traffic will be relayed even under clogging DoS attacks. To realise this mechanism, nodes maintain a priority standing of their neighbours according to the rate of control traffic experimented by NLP. The top priority is assigned to the nodes that are generating or relaying link state updates with the lowest pace and vice versa.

4.2.7 SRP

The Secure Routing Protocol (SRP) [68] is an extension for reactive routing protocols. The scheme is robust in the presence of multiple non-colluding nodes, and

provides precise routing information in a timely manner. SRP counters attacks that disrupt the route discovery mechanism and warrants the acquisition of accurate topological information. With SRP, a node initiating a route discovery is able to identify and discard replies with fake routing information. The fundamental assumption is the presence of a security association (SA) between the source node (S) and the destination node (T). The trust relationship could be established by possession of the public key of the other communicating end and then agreeing upon a shared secret key ($K_{S,T}$) between each other.

SRP, when applied to the DSR protocol, necessitates the addition of a 6-word header, which contains unique identifiers that tag the discovery process and a Message Authentication Code (MAC). While initiating a `ROUTE REQUEST`, the source node has to compute a MAC using a keyed hash algorithm that accepts as input the entire IP header, the `ROUTE REQUEST` packet and the shared key $K_{S,T}$. The transitional nodes, which pass on the `ROUTE REQUEST` towards the destination, calculate the frequencies of queries received from their neighbours in order to control the query propagation process. Each node maintains a priority ranking that is inversely proportional to the queries' rate in order to realise flow control. Any node overwhelming the network with unsolicited `ROUTE REQUEST`s will be served last due to the low priority ranking mechanism. When a `ROUTE REQUEST` is received, the destination node confirms the integrity and authenticity of the `ROUTE REQUEST` by computing the keyed hash of the required fields and then comparing it with the MAC contained in the SRP header. If the `ROUTE REQUEST` is legitimate, the destination responds with a `ROUTE REPLY` using the SRP header in a manner similar to the `ROUTE REQUEST`. The source node verifies the query identifiers and MAC integrity values of all `ROUTE REPLY` packets in order to avert any replay attacks.

Table 1: Comparison of Secure Routing Protocols for Ad-hoc Networks

Performance Parameters	ARAN	ARIADNE	SAODV	SAR	SEAD	SLSP	SRP
Type	Reactive	Reactive	Reactive	Reactive	Proactive	Proactive	Reactive
Encryption Algorithm	Asymmetric	Symmetric	Asymmetric	Symmetric/ Asymmetric	Symmetric	Asymmetric	Symmetric
MANET Protocol	AODV/DSR	DSR	AODV	AODV	DSDV	ZHLS	DSR/ZRP
Function	Uses cryptographic certificates to secure the route discovery and maintenance mechanism	Uses symmetric cryptography to secure the route discovery and maintenance mechanism	Uses asymmetric cryptography to secure the route discovery and maintenance mechanism	Uses explicit cooperative trust relationships to secure the route discovery mechanism	Uses one-way hash functions to secure topology discovery	Uses asymmetric cryptography to secure neighbour and topology discovery	Uses symmetric cryptography to secure the route discovery and maintenance mechanism
Synchronization	No	Yes	No	No	Yes	No	No
Central Trust Authority	CA Required	KDC Required	CA Required	CA/KDC Required	CA Required	CA/KDC Required	CA Required
Authentication	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Confidentiality	Yes	No	No	Yes	No	No	No
Integrity	Yes	Yes	Yes	Yes	No	No	Yes
Non-repudiation	Yes	No	Yes	Yes	No	Yes	No
Anti - Spoofing	Yes	Yes	Yes	Yes	No	Yes	Yes
DoS Attacks	No	Yes	No	No	Yes	Yes	Yes

4.3 Comparison

A comparison of the secure ad-hoc network routing protocols is shown in the Table 1. It reveals that most of the protocols have been configured to operate with reactive ad-hoc routing protocols. These protocols make use of symmetric and asymmetric cryptography to achieve the different security parameters. Some of the protocol authors have discussed the provisioning of specific security services by their protocols while others have not clearly defined the security coverage of their proposed protocols. As a result, a number of security services are claimed as being provided by most of these protocols. For example, some protocols only provide selective-field confidentiality while others provide connectionless confidentiality. The same may be reported for authentication, non-repudiation and integrity security services. There is as yet no universal agreement about many of the terms used in security literature [104], so for comparison purposes we have used security services as defined by ITU X.800 [39] and RFC 2828 [100].

Protocols using public keys provide a range of security services but may be computationally expensive to operate in real environments. This is due to the fact that the

use of public-key cryptography imposes a high processing overhead on the intermediate nodes and can be considered unrealistic for a wide range of network instances. An inefficient authentication system could become a target of a Denial-of-Service (DoS) attack by an attacker that floods nodes with malicious authentication messages, so as to saturate them with computationally expensive modulo operations. The prevention of DoS attacks in wireless networks is an intricate and challenging task. Some protocols have tried to avoid this attack by using symmetric algorithms and priority mechanisms to ensure availability of nodes.

Most of the secure protocols address attacks launched by a number of non-colluding malicious nodes. However, attacks by multiple colluding nodes, like wormhole or vertex cut, are difficult to detect and solutions to these attacks have only been suggested by ARIADNE. For node authentication ARIADNE and SEAD suggest using the TESLA broadcast authentication scheme with delayed key disclosure. However, as TESLA requires a clock synchronisation mechanism, which in turn is also prone to attacks, it is considered to be an unrealistic requirement for ad-hoc networks. For securing the mutable information in routing messages, ARIADNE, SAODV, SEAD and SLSP make use of hash chains, which is an efficient way to ensure integrity and authentication. In hostile environments, the network topology must not be exposed to adversaries by the routing messages. Exposure of the network topology may be an advantage for adversaries trying to destroy or capture nodes. The confidentiality of routing messages has been taken into account by the ARAN and SAR routing protocols.

4.4 Summary

In this chapter we have provided the description of several secure ad-hoc network routing protocols. We have presented a comparison of these protocols by highlighting their features, differences and characteristics. Most of these protocols have been developed as a practical response to specific problems that arise due to attacks on

ad-hoc network routing protocols. Consequently, the proposed solutions only cover a subset of all possible threats and are not flexible enough to be integrated with each other. While it is not clear that any particular algorithm or class of algorithm is best for all scenarios, each protocol has certain definite advantages and disadvantages, and is appropriate for application in a number of secure but managed ad-hoc network environments.

All secure routing protocols are dependent on protected key exchange and hence use a central trust authority for authentication. A variant of the central trust authority is the distributed public-key model that makes use of threshold cryptography to distribute the private key of the Certification Authority (CA) over a number of servers. Whatever the case may be, the creation and operation of a trust authority in a dynamic environment is considered both impractical and unsafe due to an number of reasons including: dependence on prerequisites, creation of a single point of failure and the requirement for sustained access.

Employing cryptographic mechanisms is another way of stimulating trust into the network, where the trust is being placed in the trusted third party and the encryption algorithm. However, as the nodes in an ad-hoc network are mobile in an open environment, they are more vulnerable to capture due to limited physical protection. Consequently, the compromise of a key repository or an authentication server can jeopardise the complete security infrastructure. The requirement of an omnipresent, and often omniscient, trust authority does resolve many of the core security issues in wired networks but it is neither practical nor feasible in an ad-hoc network. In the next chapter, we present a pragmatic trust model, which can effectively detect and evade malicious nodes, without the need of a trusted third party in the network.

Chapter 5

The Trust Model

Keeping in view the improvised nature of mobile ad-hoc wireless networks, we have developed an innovative trust model that can effectively function in a pure ad-hoc network environment. It is true that a number of ad-hoc networks are still established in a managed way but operate in a self-organised manner. However, we are looking towards networks that are self-organised both during their establishment as well as in their operation. This may seem like restricting the application of our trust model to applications other than the military or law enforcement agencies. However, we believe that our model is equally good for all types of environments. It provides the requisite level of trust in the absence of a trusted third party and reinforces existing trust levels in the presence of a trusted third party. Such an approach, when used in conjunction with cryptographic mechanisms, elevates the confidence in the underlying security schemes.

Our trust model [82] is an adaptation of the model presented by Marsh [57], which has been specifically configured for use in pure ad-hoc networks. Marsh's model works out the situational trust in agents using the general trust in the trustor. Each agent also assigns importance and utility to the situation in which it finds itself. General trust is defined as the trust that an entity places in another entity based upon all prior transactions in all situations. Importance and utility are considered

similar to knowledge, so that an agent can weigh up the costs and benefits that a particular situation holds at a particular time. In order to reduce the number of variables in our model, we merge the utility and importance of a situation into a single variable called weight, which is set depending upon the hostility of the network.

The trust model primarily represents three layers of trust. The finest layer of trust is extracted by monitoring and evaluating the events that the nodes are able to observe. These events are then combined to form the situational trust categories, which in essence represent trust in other nodes in different dimensions. These situational trust categories are then combined to form the aggregate trust, which represents a coarser level of trust in other nodes in the network. The aggregate trust layer permits a node to get a general perspective of another node, while the other two layers permit a much more refined evaluation and assignment of trust values.

The trust model can be functionally divided into the following three components: Trust agent, Reputation agent and the Combiner. A schematic representation of the trust model is shown in Fig. 2.

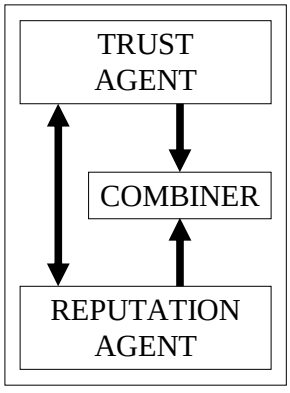


Figure 2: Structure of the Trust Model

The Trust agent, which derives trust levels from events that are directly experienced by a node, is explained in Section 5.1. In Section 5.2, the Reputation agent is discussed, which shares trust information about nodes with other nodes in the

network. The Combiner, which is described in Section 5.3, computes the aggregate trust in a node based upon the information it receives from the Trust and Reputation agents respectively.

5.1 Trust Agent

The trust model is executed by means of agents that reside on network nodes. Each agent operates autonomously and upholds its own point of view as regards to the trust hierarchy. Each agent accumulates data from events that are being experienced by a node in the current environment. These events are filtered and assigned weights, so as to compute the trust in other nodes. A trust agent essentially performs three functions: Trust Derivation, Quantification, and Computation. The estimated division of these functions with respect to the OSI reference model and the TCP/IP protocol suite is shown in Fig. 3.

<table>
<tr><th>OSI</th><th>PROPOSED</th><th>TCP/IP</th></tr>
<tr><td>Application</td><td rowspan="3">Computation
&
Quantification</td><td rowspan="3">Application</td></tr>
<tr><td>Presentation</td></tr>
<tr><td>Session</td></tr>
<tr><td>Transport</td><td rowspan="4">Derivation</td><td>Transport</td></tr>
<tr><td>Network</td><td>Internet</td></tr>
<tr><td>Data link</td><td rowspan="2">Host to Network</td></tr>
<tr><td>Physical</td></tr>
</table>

Figure 3: Segregation of Trust Agent Functions

5.1.1 Trust Derivation

Trust derivation is carried out in passive mode, i.e. without entailing the use of special interrogation packets. Essential information regarding other nodes can be gathered by examining the local network traffic. Taps are inserted at different points

in the protocol stack, so as to analyse events that are currently in progress. Likely events that can be recorded in passive mode are the measure and precision of:

- Frames received,
- Data packets forwarded,
- Control packets forwarded,
- Data packets received,
- Control packets received, and
- Streams established.

The information obtained from these events is split into one or more situational trust categories. These trust categories provide the situational trust values, which are essentially the trust in an entity in a particular context [108].

5.1.2 Trust Quantification

Trust generally portrays a continuous trend and hence discrete representation is insufficient to clearly represent trust. Secure routing protocols represent trust in a binary form by either the presence of security or its absence. As discussed earlier, the Distributed Trust Model [95] symbolises trust using six values ranging from distrust to complete trust. Similarly, Pretty Good Privacy (PGP) [29] uses four values ranging from unknown to fully trusted to signify trust levels. Discrete values, although straightforward to represent and categorise trust, are not suitable for application to ad-hoc networks. Trust in ad-hoc networks is never static since the trust relationships are consistently varying due to the dynamic topology. The period of interaction with other nodes being significantly succinct, necessitates that trust be represented as an incessant range to differentiate between nodes with comparable trust levels [42]. In our model we signify trust from -1 to +1, representing an unremitting range from complete distrust to absolute trust.

5.1.3 Trust Computation

During trust computation, weights are assigned to the previously monitored and quantified events. All nodes dynamically allocate these weights depending upon their own criteria and situation. These weights are represented in a continuous range from 0 to +1, where 0 signifies unimportant and +1 signifies most important. The situational trust categories are then combined to calculate the direct trust in a particular node. We define the direct trust in node y, by node x, as T_{xy} and is given by the following equation:

$$\mathrm{T}_{xy} = \sum_{i=1}^{n} [W_{xy}(i) \times \mathrm{T}_{xy}(i)]$$

where $W_{xy}(i)$ is the weight of the i^{th} trust category of node y to node x and $\mathrm{T}_{xy}(i)$ is the situational trust of node x in the i^{th} trust category of node y. The total number of trust categories n is dependent upon the protocol and scenario to which the trust model is being applied. Each node in the network maintains its own direct trust rating in relation to other nodes in the network as shown in Fig. 4.

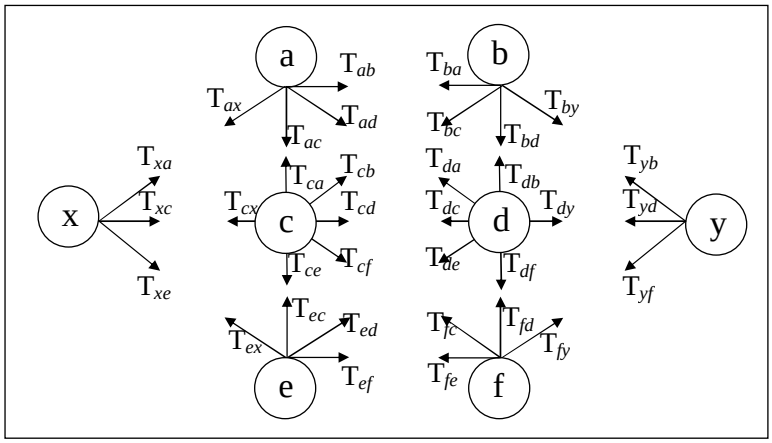


Figure 4: Direct Trust Relationships

The situational and direct trust values are maintained and dynamically updated to represent the current trust status of nodes. The situational and direct trust table from the perspective of node y (of Fig. 4) is shown in Table 2.

Table 2: Situational and Direct Trust Table

Node	Situational Trust in Category					Direct Trust
	1	*2*	...	...	*n*	
b	$T_{yb}(1)$	$T_{yb}(2)$	...	...	$T_{yb}(n)$	T_{yb}
d	$T_{yd}(1)$	$T_{yd}(2)$	...	...	$T_{yd}(n)$	T_{yd}
f	$T_{yf}(1)$	$T_{yf}(2)$	...	...	$T_{yf}(n)$	T_{yf}
...	...	...	...	...	...	...
...	...	...	...	...	...	...

5.2 Reputation Agent

The majority of the events experienced by a node occur through interactions between adjacent nodes. These events help in establishing direct trust relationships in the neighbourhood. However, in comparison, very few events are directly experienced between nodes that are more than one hop away. This necessitates that the reputation of nodes be also taken into consideration while computing trust in other nodes [113, 22]. The reputation agent requests recommendations from nodes regarding their belief in other nodes (REQUESTER). Similarly, it sends its own recommendations to other requesting nodes (RECOMMENDER). Two possible mechanisms, which can be used to disperse direct trust values, are: Integrated Reputation Exchange [77, 80, 91] and Independent Reputation Exchange [76]. In the former mechanism, existing control or data packets are utilised to spread reputations, while in the latter special packets are used to request and receive trust information.

5.2.1 Integrated Reputation Exchange

In an Integrated Reputation Exchange, the route discovery process or subsequent data connections are used to spread the trust information. In such a mechanism, the direct trust values are piggy-backed onto the control or data packets, which act as carrier, permitting trust dispersal beyond a single hop. For example, `ROUTE`

REPLY packets can be used to spread the direct trust information of nodes, as shown in Fig. 5.

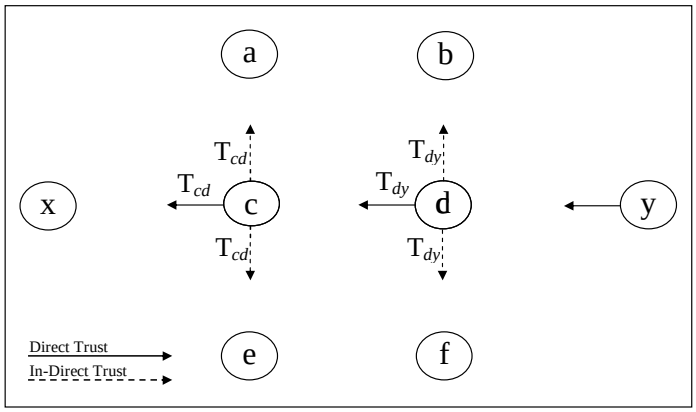


Figure 5: Trust Propagation through ROUTE REPLY Packets

Each node acting as a forwarder in an ongoing route discovery session, appends the direct trust value to the ROUTE REPLY packet, which was sent by the preceding node. In this manner, when the ROUTE REPLY packet is sent, it also distributes the trust information of other nodes in the network in a direct and indirect manner. The ROUTE REPLY packet from node y to node x, directly disseminates the trust level of node y to node c and that of node d to node x. In addition, the trust levels of nodes y and d can also be communicated in an indirect manner to nodes b and f and to nodes a and e respectively. However, for subsequent updating of trust information further ROUTE REPLY packets need to be initiated. Similarly, by using ROUTE REQUEST packets as a carrier, trust information can be spread all over the network owing to the flooding nature of the selected carrier. Using data packets as a carrier also permits real time reputation exchange, which is spread along the path of the data connection. Such mechanisms, although avoid additional packets for reputation exchange, are highly dependent upon the type of carrier selected and can accordingly influence the effective throughput and trust dispersal efficiency of the network.

5.2.2 Independent Reputation Exchange

The exchange of recommendations between the REQUESTER and the RECOMMENDER can also be implemented periodically or on an 'as per requirement' basis using a higher layer protocol. However, in order to control spread of fallacious requests and replies for recommendations, we propose using HashCash [4] tokens along with recommendations. HashCash is a CPU cost function that computes a proof-of-effort token using a hash function. This token can be derived in an interactive or non-interactive setting. In the interactive setting, the RECOMMENDER issues a challenge to the REQUESTER to solve. The REQUESTER, after solving the challenge, returns the result to the RECOMMENDER in the form of a token. The RECOMMENDER verifies the token to confirm the proof-of-effort by the REQUESTER. In the non-interactive setting, the REQUESTER computes a known challenge, based on some current information, and returns it to the RECOMMENDER in the form of a token. Solving the challenge is computationally expensive for the REQUESTER while its verification is simple to perform for the RECOMMENDER. This ensures that the token, which is presented by the REQUESTER to the RECOMMENDER, is backed by a quantifiable proof-of-effort by the REQUESTER. HashCash uses cryptographic hash functions to compute the tokens. Each token has four fields as shown in Fig. 6.

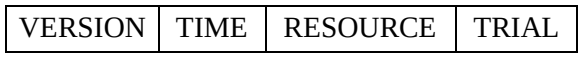


Figure 6: HashCash Token Format

The HashCash VERSION number identifies the type of hash function employed, TIME is the current date and time, RESOURCE is the challenge string that is negotiated between the REQUESTER and RECOMMENDER and TRIAL is the string that the REQUESTER endeavours to work out. The REQUESTER, when computing the token, tries different combinations of the trial string to find a token, which when hashed, provides a certain number of Most Significant Bits (MSB) as zero. The number of these MSBs define the amount of work that the REQUESTER

has to perform before finding a partial collision. The token is sent to the RECOMMENDER, which can easily verify it through a single hash operation.

We use HashCash in its non-interactive setting in order to reduce the overall number of exchange messages. The REQUESTER sends a Recommendation Request (`REC_REQ`) packet, which is responded to by a Recommendation Reply (`REC_REP`) packet by the RECOMMENDER indicating the reputation of the requested node. The `REC_REQ` packet contains the identity of the REQUESTER (ID_{RQ}), identity of the target node whose reputation is desired (ID_{TGT}), the situational or direct trust category whose value is being sought (ID_{TT}), a unique `REC_REQ` identification number (ID_{RRQ}) along with a HashCash token. The Recommendation Exchange Protocol used for requesting and replying to recommendations is shown in Fig. 7.

1. REQUESTER → RECOMMENDER (REC_REQ **)** ID_{RQ}, ID_{RRQ}, ID_{TT}, ID_{TGT}, Hash {Ver, TS, ID_{REC}, Trial}	
2. RECOMMENDER → REQUESTER (REC_REP **)** ID_{RRQ}, T_T	
ID_{RQ}	Identity of Requesting Node
ID_{REC}	Identity of Recommending Node
ID_{TGT}	Identity of Target Node
ID_{RRQ}	Unique REC_REQ number
ID_{TT}	Identity of Trust Type
T_T	Value of Trust Type
Ver	Identity of hash function
TS	Informs of date and time when this message was generated
Trial	Number to be determined in order to generate a valid token

Figure 7: Recommendation Exchange Protocol

The HashCash token contains the identity of the hash function, a timestamp (TS), the identity of the RECOMMENDER (ID_{REC}) and a trial number. This trial number is a random number that is found through trial and error in such a way that, when the token is hashed, it results in achieving a certain number of zeros in the Most Significant Bits. The higher the number of Most Significant Bits, the more time it takes to generate the token.

When the `REC_REQ` packet reaches the RECOMMENDER it verifies whether the TS falls within a permitted time threshold. This threshold is based upon loosely

synchronised clocks and can be adjusted based upon the antagonism of the environment. The HashCash token is then hashed to find the required number of zeroes in the MSB of the hash. If the resultant is true it implies that the sender must have spent an appreciable amount of effort before generating the `REC_REQ` packet. The RECOMMENDER, in response sends a `REC_REP` packet that contains the `REC_REQ` identification number ($\mathrm{ID_{RRQ}}$) and a trust value ($\mathrm{T_T}$) corresponding to the ordered 2-tuple ($\mathrm{ID_{TGT}}$, $\mathrm{ID_{TT}}$) from its situational and direct trust table. Reputations are continuously updated upon receipt of further reputations, so as to represent the current trust status of other nodes. The quality and accuracy of the reputations is also taken into account and the same can be maintained in the direct trust table as an additional situational trust category.

A direct trust value of a node d, which is sent by node b, is represented as $\mathrm{T}_{\underline{bd}}$ and the situational trust value in category n of node d that has been sent by node b is represented as $\mathrm{T}_{\underline{bd}}(n)$. Reputations received by node y (of Fig. 4) from its adjacent neighbours are logged in a format as shown in Table 3.

5.3 Combiner

The combiner receives trust values from the Trust and Reputation agents. To compute the total trust value of a target node, we have adopted and modified the probabilistic computing method by Beth et al. [6], however, any trust computing method [95, 45, 46] may be used for combining trust values. We explain the method by means of an example as shown in Fig. 8.

If T_{xz} represents the direct trust between nodes x and z and $\mathrm{T}_{\underline{zy}}$ represents the received recommended trust from node z between nodes z and y then the derived trust (T_{xzy}) between nodes x and y via node z would be as follows:

$$\mathrm{T}_{xzy} = \mathrm{T}_{xz} \odot \mathrm{T}_{\underline{zy}} = 1 - (1 - \mathrm{T}_{xz})^{\mathrm{T}_{\underline{zy}}}$$

Table 3: Reputation Table

Node	*b*	*d*	*f*	...	...
b	-	$T_{\underline{bd}}$ $T_{\underline{bd}}(1)$ $T_{\underline{bd}}(2)$ ... ... $T_{\underline{bd}}(n)$	$T_{\underline{bf}}$ $T_{\underline{bf}}(1)$ $T_{\underline{bf}}(2)$ ... ... $T_{\underline{bf}}(n)$	...	...
d	$T_{\underline{db}}$ $T_{\underline{db}}(1)$ $T_{\underline{db}}(2)$ ... ... $T_{\underline{db}}(n)$	-	$T_{\underline{df}}$ $T_{\underline{df}}(1)$ $T_{\underline{df}}(2)$ ... ... $T_{\underline{df}}(n)$	...	...
f	$T_{\underline{fb}}$ $T_{\underline{fb}}(1)$ $T_{\underline{fb}}(2)$ ... ... $T_{\underline{fb}}(n)$	$T_{\underline{fd}}$ $T_{\underline{fd}}(1)$ $T_{\underline{fd}}(2)$ ... ... $T_{\underline{fd}}(n)$	-	...	...
...	...	...	...	-	...
...	...	...	...	...	-

Similarly, the derived trust T_{xwy} between nodes x and y via node w, for which a recommendation has been received, is worked out as follows:

$$T_{xwy} = T_{xw} \odot T_{\underline{wy}} = 1 - (1 - T_{xw})^{T_{\underline{wy}}}$$

The derived trust (T_{zwy}) between nodes z and y via node w is void of any direct trust experiences and hence would be computed as follows:

$$T_{zwy} = T_{\underline{zw}}.T_{\underline{wy}}$$

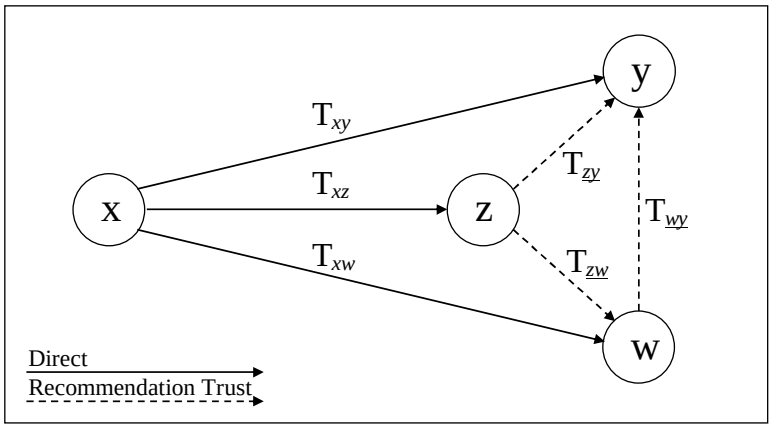


Figure 8: Derivation of Trust Relationships

The aggregate trust $\mathrm{T}(y)$ between node x and y is computed using all known direct and derived trust values in the following manner:

$$\mathrm{T}(y) = 1 - (1 - \mathrm{T}_{xy}).(1 - \mathrm{T}_{xzy}).(1 - \mathrm{T}_{xwy})$$

The derived and aggregate trust values are maintained by the combiner and dynamically updated whenever a change occurs either in the situational and direct trust table or the reputation table. The direct, recommendation and aggregate trust table maintained by node y (of Fig. 4) is shown in Table 4.

Table 4: Direct, Recommendation and Aggregate Trust Table

Node	Direct	Recommendation	Aggregate
a		$\mathrm{T}_{yba} = 1 - (1 - \mathrm{T}_{yb})^{\mathrm{T}_{ba}}$ $\mathrm{T}_{yda} = 1 - (1 - \mathrm{T}_{yd})^{\mathrm{T}_{da}}$	$\mathrm{T}(a) = 1-(1 - \mathrm{T}_{yba}).(1- \mathrm{T}_{yda})$
b	T_{yb}	$\mathrm{T}_{ydb} = 1 - (1 - \mathrm{T}_{yd})^{\mathrm{T}_{db}}$	$\mathrm{T}(b) = 1-(1 - \mathrm{T}_{yb}).(1 - \mathrm{T}_{ydb})$
c		$\mathrm{T}_{ybc} = 1 - (1 - \mathrm{T}_{yb})^{\mathrm{T}_{bc}}$ $\mathrm{T}_{ydc} = 1 - (1 - \mathrm{T}_{yd})^{\mathrm{T}_{dc}}$ $\mathrm{T}_{yfc} = 1 - (1 - \mathrm{T}_{yf})^{\mathrm{T}_{fc}}$	$\mathrm{T}(c) = 1-(1 - \mathrm{T}_{ybc}).\ (1- \mathrm{T}_{ydc}).(1- \mathrm{T}_{yfc})$
d	T_{yd}	$\mathrm{T}_{ybd} = 1 - (1 - \mathrm{T}_{yb})^{\mathrm{T}_{bd}}$ $\mathrm{T}_{yfd} = 1 - (1 - \mathrm{T}_{yf})^{\mathrm{T}_{fd}}$	$\mathrm{T}(d) = 1-(1 - \mathrm{T}_{yd}).(1- \mathrm{T}_{ybd}).(1- \mathrm{T}_{yfd})$
e		$\mathrm{T}_{yde} = 1 - (1 - \mathrm{T}_{yd})^{\mathrm{T}_{de}}$ $\mathrm{T}_{yfe} = 1 - (1 - \mathrm{T}_{yf})^{\mathrm{T}_{fe}}$	$\mathrm{T}(e) = 1-(1 - \mathrm{T}_{yde}).(1- \mathrm{T}_{yfe})$
f	T_{yf}	$\mathrm{T}_{ydf} = 1 - (1 - \mathrm{T}_{yd})^{\mathrm{T}_{df}}$	$\mathrm{T}(f) = 1-(1 - \mathrm{T}_{yf}).(1 - \mathrm{T}_{ydf})$

5.4 Summary

In this chapter, we have presented a trust model for establishing and maintaining trust, without the requirement of a trusted third party, in the network. The proposed model establishes trust in ad-hoc networks without any cryptographic means and is hence void of any superfluous assumptions and requirements. The trust model uses a trust agent to derive, quantify and compute direct trust levels, for all adjacent neighbours, depending upon their preceding and contemporary actions. The trust agent takes into account not only the packet forwarding events but also verifies a number of other parameters including the integrity of forwarded traffic and sincerity in execution of the routing protocol. To obtain trust information concerning nodes that are not in the near vicinity, nodes can make use of an integrated or independent reputation exchange mechanism to share reputations. In the integrated reputation exchange, the underlying routing protocol is used as a carrier to propagate direct trust values, while in the later a separate higher level protocol is used as the carrier. These direct trust values and reputations are then combined to form a single aggregate trust value, which can be used to differentiate between trustworthy and untrustworthy nodes. In the next chapter, we apply this trust model to three reactive routing protocols, in order to improve upon the routing performance in a network, which is under attack conditions.

Chapter 6

Trusted Routing Protocols

To demonstrate the utility of our proposed trust model, in this chapter, we present its applicability to the AODV, DSR and TORA routing protocols. As these protocols are currently under their development phase, and are being constantly improved, we have applied our model to the latest known working versions. We first elucidate how trust can be extracted from each of these protocols using different trust categories. Although, additional trust categories can also be extracted from each of these protocols, but here we have only described those that have maximum impact on trust development. We then explain how the extracted trust can be used to improve upon the routing performance in a mobile ad-hoc wireless network.

In Section 6.1, we first highlight the events that can be monitored in each routing protocol, which can subsequently be used to derive trust from the network. In Section 6.2, we explain how these events can be normalised to form situational trust categories. These situational trust categories are then used to compute the direct trust as described in Section 6.3. We then discuss possible trust propagation mechanisms in Section 6.4 and demonstrate the application of trust to the routing process in Section 6.5.

6.1 Trust Derivation

We exploit one or more inherent features of each routing protocol to develop the following trust categories:

6.1.1 Acknowledgments (P_A)

Applicability : DSR, AODV and TORA
A node can get information about the successful transmission of any packet that it had sent, through the following three methods:

Link-Layer Acknowledgements: Using Link-Layer acknowledgments the underlying MAC protocol provides feedback of the successful delivery of the transmitted data packets.

Passive Acknowledgements: In this method the sender node places itself in promiscuous mode after the transmission of any packet, so as to overhear the retransmission by the recipient nodes.

Network Layer Acknowledgements: In AODV, this method allows the sender of a `ROUTE REPLY` packet to explicitly request a `ROUTE REPLY` acknowledgement from the recipient of the `ROUTE REPLY` packet. In DSR, the sender is permitted to explicitly request a network layer acknowledgement from the next hop using the DSR options header. In TORA, network layer acknowledgements are sent in response to object block receptions in IMEP when either reliable delivery is required or a receiver is implicitly or explicitly included in the response list. Once a neighbouring node has acknowledged a given block of data, it is removed from the response list and is not required to acknowledge future retransmissions.

All of the above methods provide information about the successful transmission of a packet. However, the passive acknowledgment method also provides the following

information about the next hop, including:

- It is acting like a black hole if the packet is dropped and not retransmitted,
- It is performing a modification attack if the contents have been fallaciously modified,
- It is undertaking a fabrication attack if a self generated fallacious packet is transmitted,
- It is performing an impersonation attack if the MAC or IP addresses have been spoofed,
- It is showing selfish behaviour by not retransmitting a packet, and
- It is inducing latency by delaying the retransmission of the packet.

The method of passive acknowledgment can be further classified into acknowledgements for data packets and acknowledgements for control packets. The number of these acknowledgements occurring with respect to every node are maintained and tabulated as shown in Table 5. For every packet transmitted, the appropriate counter in the table for success or failure is incremented, depending if the neighbouring node has correctly forwarded it or not.

Table 5: Trust table for Category P_A

Node Acknowledgement	Route Request/QRY (R_q)		Route Reply/UPD (R_p)		Route Error/CLR (R_e)		Route Optimisation/OPT (R_o)		Data (D)	
	Success R_{qs}	Fail R_{qf}	Success R_{ps}	Fail R_{pf}	Success R_{es}	Fail R_{ef}	Success R_{os}	Fail R_{of}	Success D_s	Fail D_f

6.1.2 Packet Precision (P_P)

Applicability : DSR, AODV and TORA

The category Packet Precision verifies the integrity of the data and control packets

that are either received or forwarded by nodes in the network. For intermediate nodes that execute a distance vector routing protocol, the received `ROUTE REQUEST` packets are used to create the REVERSE ROUTEs to the source. These routes are later used to send data packets to the source nodes. In the same way, the `ROUTE REPLY` packets help in forming FORWARD ROUTEs that lead to the destination. The correctness of these control packets thus plays a vital role in the establishment of accurate routes through the network. Similarly, for source routing protocols the accuracy of control packets is equally imperative.

The precision of the control packets is determined when the routes are actually utilised while that of the data packets is verified during forwarding. For instance, if routing packets are received that are found to be correct and efficient, then the originator can be allotted a higher trust value along with the set of nodes provided in that packet. The above method can be further categorised into data and control packet types and allocated different trust values as shown in Table 6. Counters are maintained for every received packet and are incremented based upon the accuracy or inaccuracy of the packet.

Table 6: Trust table for Category P_P

Node Packet Precision	Route Request/QRY (R_q)		Route Reply/UPD (R_p)		Route Error/CLR (R_e)		Route Optimisation/OPT (R_o)		Data (D)	
	Success R_{qs}	Fail R_{qf}	Success R_{ps}	Fail R_{pf}	Success R_{es}	Fail R_{ef}	Success R_{os}	Fail R_{of}	Success D_s	Fail D_f

6.1.3 Gratuitous Route Replies (G_R)

Applicability : DSR and AODV

The DSR protocol provides the facility of 'route shortening' to avoid unnecessary intermediate nodes. For example, if a node overhears a data packet that is supposed to traverse a number of nodes before passing through it, then this node creates a shorter route, known as a gratuitous route, and sends it to the original sender. The AODV protocol permits intermediate nodes with fresher routes (larger destination

sequence numbers) to respond to `ROUTE REQUEST` queries, if the **DESTINATION ONLY** flag is not set in the `ROUTE REQUEST` packet. This reduces the overall latency and resource utilisation in contrast to the case where the `ROUTE REPLY` packets are generated only by the destination node. Although beneficial, these intermediate `ROUTE REPLY` packets don't inform the destination node regarding a route to the source node. Thus for connection-oriented sessions, the destination node has to initiate another route discovery process for the intended communication. In order to avoid such repetitive route discoveries the source node can set a **GRATUITOUS ROUTE REPLY** flag in the `ROUTE REQUEST` packet. If this flag is set then the intermediary responder to a `ROUTE REQUEST` is also required to create and unicast an appropriate `ROUTE REPLY` to the final destination. The `GRATUITOUS ROUTE REPLY` packets can be considered as a trust category as they provide the following information about the sender:

- It is displaying either malicious or benevolent behaviour, and
- It is not showing selfish behaviour.

If the gratuitous route is found to be accurate, then the originator can be allotted a higher trust value along with the set of nodes provided in that route. The above method can be used to allocate trust values to nodes, as shown in Table 7. All `GRATUITOUS ROUTE REPLY` packets that are found to be correct or incorrect are recorded using appropriate counters.

Table 7: Trust table for Category G_R

Node	Gratuitous Route Replies (G)	
	Success G_s	Fail G_f

6.1.4 Blacklists (B_L)

Applicability : DSR and AODV

DSR and AODV maintain blacklists for nodes displaying uni-directional behaviour,

i.e. if a neighbour node has received a packet and either due to a unidirectional link or selfish behaviour the sender cannot hear it retransmitting. If the MAC protocol is expected to provide feedback (like IEEE 802.11) then this implies that the links must be bi-directional and the neighbouring node is acting selfishly [98]. The blacklists can be used to provide trust values for nodes while computing route confidence levels. The format of the trust table based on blacklists is shown in Table 8.

Table 8: Trust table for Category B_L

Node	Present in Blacklist (B)

6.1.5 BEACON/HELLO Packets (H_M)

Applicability : AODV and TORA

AODV uses `HELLO` packets to maintain local neighbourhood connectivity information. Each active node that has not sent any broadcast in a certain period, broadcasts a `HELLO` packet (`ROUTE REPLY` packet with Hop-Count=0) with **TTL** set to 1. These packets ensure that all neighbours maintain active routes between each other at all times. All recipient nodes create FORWARD ROUTEs to the transmitting node. The absence of a `HELLO` packet from a neighbour for a certain duration makes the route to that node invalid. `HELLO` packets indicate that a node is actively participating in the routing mechanism and not acting as either a passive eavesdropper or a selfish node. TORA uses `BEACON` packets to ascertain the connection status between adjacent nodes. Each node periodically broadcasts a `BEACON` that contains a Router Identification number. In response to the `BEACON`, every recipient node broadcasts an `ECHO` packet that contains its IP address. These packets ensure that all nodes maintain local neighbourhood connectivity information at all times. This feature can be used to assess the reliability of the neighbours based upon the frequency and accuracy of these packets. The format of the trust table based on these messages is shown in Table 9.

Table 9: Trust table for Category H_M

Node	Beacon/Hello Packet (H)	
	Success H_s	Fail H_f

6.1.6 Destination Unreachable Messages (D_U)

Applicability : AODV

In AODV, data packets that are waiting for a route to be established are usually buffered at the node requesting the route. This buffering is accomplished in a 'First-in First-out' manner. However, if the route discovery exceeds the maximum number of `ROUTE REQUEST` retries then the data packets are dropped from the buffer and a Destination Unreachable Message is returned to the sending application. The category D_U ensures that the network is not saturated by extensive `ROUTE REQUEST`s propagating from the source node. AODV also supports local link repairing at the intermediate nodes. Any received data packet with a broken next hop is buffered for a certain interval in the Interface Queue, so as to facilitate the discovery of an alternate suitable link leading to the same destination. In the event that an alternate longer route is found, a `ROUTE ERROR` is propagated to the source node informing it of the link severing. The category D_U also maintains a count of such `ROUTE ERROR` packets to ascertain the sincerity of the sender, which can subsequently be used to compute route confidence levels. The format of the trust table based on the Destination Unreachable Message is shown in Table 10.

Table 10: Trust table for Category D_U

Node	Destination Unreachable Message (U)	
	Success U_s	Fail U_f

6.1.7 Salvaging (S_G)

Applicability : DSR

In DSR, if an intermediate node receives a packet for which the next hop is not

available, it may scan its route cache to find an alternate route to the final recipient. If it can locally find such a route, it salvages the packet to form a new source header and sends the packet across. It also informs the original sender of the packet about the failed link, through a `ROUTE ERROR` packet, to minimise subsequent salvage operations. If the salvaged route is found to be correct then it reveals that the sender of the `ROUTE ERROR` is displaying benevolent and altruistic behaviour. Hence, this information can be used to build up trust levels and considered as a trust category. All salvaged `ROUTE ERROR`s found to be correct or incorrect are recorded using counters, as shown in Table 11.

Table 11: Trust table for Category S_G

Node	Salvage Route Error (S)	
	Success S_s	Fail S_f

6.1.8 Authentication Objects (A_O)

Applicability : TORA

In TORA, IMEP enabled nodes are able to support multiple types of authentication, from simple to complex verification schemes. The IMEP messages between any two nodes are verified using IMEP Authentication Objects [40], which are passed with all IMEP messages. The Authentication Objects contain a digital signature of the contents of the IMEP message. Based on the type of Public Key Infrastructure (PKI), the nodes have the option to pass the certificate along with every IMEP message. The recipient node uses the Public Key from the certificate, and the digital signature to verify the contents of the IMEP message. If such a PKI exists (in case of a managed ad-hoc networks), then the Authentication Objects are used to define a situational trust category. However, in the event that the PKI is nonexistent, then the category A_O is simply disregarded. The trust table based upon Authentication Objects is shown in Table 12.

Table 12: Trust table for Category A_O

Node	Authentication Objects (A)	
	Success A_s	Fail A_f

6.2 Trust Quantification

The events recorded during the trust derivation process are quantised and assigned weights, so as to compute the situational trust values for different nodes. For example, in order to compute the situational trust in category Passive Acknowledgements (P_A), the events recorded in Table 5 are first quantised using the following equations:

$$
\begin{aligned}
R_p &= \frac{R_{ps} - R_{pf}}{R_{ps} + R_{pf}} \quad for \quad R_{ps} + R_{pf} \neq 0 \quad else \quad R_p = 0 \\
R_q &= \frac{R_{qs} - R_{qf}}{R_{qs} + R_{qf}} \quad for \quad R_{qs} + R_{qf} \neq 0 \quad else \quad R_q = 0 \\
R_e &= \frac{R_{es} - R_{ef}}{R_{es} + R_{ef}} \quad for \quad R_{es} + R_{ef} \neq 0 \quad else \quad R_e = 0 \\
R_o &= \frac{R_{os} - R_{of}}{R_{os} + R_{of}} \quad for \quad R_{os} + R_{of} \neq 0 \quad else \quad R_o = 0 \\
D &= \frac{D_s - D_f}{D_s + D_f} \quad for \quad D_s + D_f \neq 0 \quad else \quad D = 0
\end{aligned}
$$

By normalising the values of R_p, R_q, R_e, R_o and D we limit the trust values between -1 to +1. Negative values for trust can occur as a result of more failures than successes for an event. Hence, a trust value of -1 represents complete distrust, a value of 0 implies a non-contributing event and a value of +1 means absolute trust in a particular event. These trust levels are then assigned weights in a static or dynamic manner depending on their utility and importance. The situational trust $\mathrm{T}_{xy}(P_A)$ in node y for trust category P_A is computed by node x using the following equations:

For DSR/AODV

$$\mathrm{T}_{xy}(P_A) = W_{xy}(R_p) \times R_p + W_{xy}(R_q) \times R_q + W_{xy}(R_e) \times R_e + W_{xy}(D) \times D$$

For TORA

$$\begin{aligned}\mathrm{T}_{xy}(P_A) &= W_{xy}(R_p) \times R_p + W_{xy}(R_q) \times R_q + W_{xy}(R_e) \times R_e + W_{xy}(R_o) \times R_o \\ &\quad + W_{xy}(D) \times D\end{aligned}$$

where W_{xy} is the weight assigned by node x to the event that took place with node y. Similarly, all other events recorded in Tables 6 to 12 are quantised and weighed, in order to determine the situational trust in categories P_P, G_R, B_L, H_M, D_U, S_G and A_O respectively.

6.3 Trust Computation

The situational trust values from all trust categories (P_A, P_P, G_R, B_L, H_M, D_U, S_G and A_O) are then combined according to their assigned weights, to compute a direct trust level for a particular node. The direct trust in node y by node x is represented as T_{xy} and given by the following equations:

AODV

$$\begin{aligned}\mathrm{T}_{xy} &= W_{xy}(P_A) \times \mathrm{T}_{xy}(P_A) + W_{xy}(P_P) \times \mathrm{T}_{xy}(P_P) + W_{xy}(G_R) \times \mathrm{T}_{xy}(G_R) + \\ &\quad W_{xy}(B_L) \times \mathrm{T}_{xy}(B_L) + W_{xy}(H_M) \times \mathrm{T}_{xy}(H_M) + W_{xy}(D_U) \times \mathrm{T}_{xy}(D_U)\end{aligned}$$

DSR

$$\begin{aligned}\mathrm{T}_{xy} &= W_{xy}(P_A) \times \mathrm{T}_{xy}(P_A) + W_{xy}(P_P) \times \mathrm{T}_{xy}(P_P) + W_{xy}(G_R) \times \mathrm{T}_{xy}(G_R) + \\ &\quad W_{xy}(B_L) \times \mathrm{T}_{xy}(B_L) + W_{xy}(S_G) \times \mathrm{T}_{xy}(S_G)\end{aligned}$$

TORA

$$\begin{aligned}\mathrm{T}_{xy} &= W_{xy}(P_A) \times \mathrm{T}_{xy}(P_A) + W_{xy}(P_P) \times \mathrm{T}_{xy}(P_P) + W_{xy}(H_M) \times \mathrm{T}_{xy}(H_M) + \\ &\quad W_{xy}(A_O) \times \mathrm{T}_{xy}(A_O)\end{aligned}$$

where W_{xy} represents the weight assigned to a situational trust category of node y by node x. The direct trust tables for the DSR, AODV and TORA protocols are shown in Table 13. These tables are populated with entries, which are maintained and updated for each node based upon the frequency of events and severity of the situation.

Table 13: Direct Trust Tables

AODV

Node	Passive Acknowledgement	Packet Precision	Gratuitous Route Replies	Black Lists	Beacon/ Hello Packets	Destination Unreachable Messages	Direct Trust
y	$T_{xy}(P_A)$	$T_{xy}(P_P)$	$T_{xy}(G_R)$	$T_{xy}(B_L)$	$T_{xy}(H_M)$	$T_{xy}(D_U)$	T_{xy}

DSR

Node	Passive Acknowledgement	Packet Precision	Gratuitous Route Replies	Black Lists	Salvage Route Errors	Direct Trust
y	$T_{xy}(P_A)$	$T_{xy}(P_P)$	$T_{xy}(G_R)$	$T_{xy}(B_L)$	$T_{xy}(S_G)$	T_{xy}

TORA

Node	Passive Acknowledgement	Packet Precision	Authentication Objects	Beacon/ Hello Packets	Direct Trust
y	$T_{xy}(P_A)$	$T_{xy}(P_P)$	$T_{xy}(A_O)$	$T_{xy}(H_M)$	T_{xy}

6.4 Trust Propagation

Reputations or indirect trust can be shared in the network as a distinct higher level protocol or as an integrated component of the routing protocol. A higher layer reputation exchange protocol is easy to implement, where the direct trust values are sought from other network nodes. While an integrated reputation exchange protocol needs to be customised for each underlying routing protocol. We undertook an analytic study on applying an independent reputation exchange protocol to the three protocols [76, 78, 79]. The following points summarise the main observations of the study:

- In an ad-hoc network, the node energy is a precious resource. By employing HashCash tokens in such an environment, the number of malicious recommendation request packets is suppressed, facilitating overall energy savings.

This is achieved primarily due to the computation constraints imposed by the HashCash mechanism.

- The amount of work required for each token can be set interactively or non-interactively, based upon the importance of different types of recommendations. Interactive HashCash makes it possible to dynamically adjust the amount of work that the REQUESTER is required to perform based on the RECOMMENDER's CPU load.

- By using interactive HashCash mechanisms during times of peak load, Denial of Service attacks can be countered effectively. The verification of a token by a RECOMMENDER requires negligible resources, in contrast to the effort required for the generation of a token by the REQUESTER. Also, the collision size can be increased or decreased based on the severity of the potential attack. By exercising such a policy, malicious nodes will be restrained to overload the system with fallacious recommendation packets.

- The exchange of reputations augments the control packet overhead and latency, especially when the reputations are sought from nodes beyond a single hop.

- The REQUESTER should have prior trust in the RECOMMENDER, and the RECOMMENDER must know the target node, whose reputation is sought by the REQUESTER, in advance. This is most of the times difficult to determine pre-hand and thus consumes unnecessary computation and power resources of the REQUESTER for generating the HashCash token and the Recommendation Request (`REC_REQ`) packet. Similarly a RECOMMENDER, which can easily verify the proof of effort by the REQUESTER without having any knowledge of the target node, still has to spend its energy for transmitting a redundant Recommendation Reply (`REC_REP`) packet.

- The protocol is also vulnerable to tampering where the contents of the `REC_REQ` and `REC_REP` packets need to be safeguarded for accurate trust dissemination.

In order to limit the extraneous packets generated during independent reputation exchange, we have integrated the reputation exchange mechanism with the route discovery process of the three routing protocols. We propose piggy-backing the direct trust information of nodes onto control packets. This is achieved by appending the direct trust level of the preceding node to the propagated `ROUTE REQUEST`, `ROUTE REPLY`, `QRY` and `UPD` control packets.

In the normal DSR protocol all intermediate nodes append their IP addresses to the `ROUTE REPLY` and `ROUTE REQUEST` packets. We propose that, with the integrated reputation exchange, all intermediate nodes also affix the direct trust value (Trust of Node *n* in Node *n-1*) of the immediate sender (*n-1*) to the corresponding `ROUTE REPLY` and `ROUTE REQUEST` packet. All nodes receiving these `ROUTE REPLY` and `ROUTE REQUEST` packets, extract relevant trust information from them and add it to their respective reputation tables. The flooding of the `ROUTE REQUEST` packet permits propagating of trust information to all nodes in the network. However, the precise amount of trust information received by each node varies from a single hop to the maximum network diameter, depending upon the location of the querying node in the network. Similarly, the `ROUTE REPLY` packets convey the complete trust information of a path to a querying node, which helps a sending node to select a trustworthy route for a data connection from multiple available routes. The modified DSR `ROUTE REPLY` and `ROUTE REQUEST` packets are shown in Appendix D.

In contrast to a source routing protocol, both AODV and TORA have no facility for propagating any information beyond a single hop. This limitation is directly related to the protocols's mode of operation, in which routes are created on a hop-to-hop basis rather than from a source-to-destination basis. In order to retain the protocol's inherent working, we propagate the direct trust value of a node across one hop by piggy-backing it onto control packets. Each node that receives a `ROUTE REPLY`, `ROUTE REQUEST`, `UPD` or `QRY` packet appends to it the IP address (Last Node IP Address) and the direct trust value (Trust in Last Node) of the node, which had earlier sent that packet. This packet is then further sent into the network,

where each recipient node extracts the available trust information and appends the contemporary direct trust value and IP address of the sender to the packet. The modified AODV and TORA control packets are shown in Appendices E and F respectively.

6.5 Trust Application

Trust-based routing schemes generally segregate nodes into two possible states i.e. either benevolent or malevolent. With our proposed trust model, there is no such discrete segregation and all nodes in the network are considered potential routing candidates based upon their contemporary trust levels. In essence, our scheme facilitates best-effort delivery even in the presence of malicious and selfish nodes. Thus all available nodes, based upon their respective trust levels, are selected for the routing process. If nodes with a higher trust rating are not available in the neighbourhood then nodes with a lower trust rating are engaged and given a subsequent chance to improve upon their past performance.

All participating nodes in the network initially consider every other unknown[1] node in the network as trustworthy and set its direct trust value equal to the initial trust value (T_I). The T_I can be set in two different ways: neutral or trustworthy. In the neutral category the T_I of the node is set to 50%. This implies that all nodes are initially considered neutral until the time they have interactions with other nodes in the network. Based upon these interactions the trust values can either improve or diminish. However, this approach causes problems when dealing with nodes portraying grey holes. As these nodes vary their packet drop pattern, it is possible that the trust level of such nodes may increase more than that of a neutral node. Such an occurrence would prevent neutral nodes from being analysed by the trust model of node x, in the presence of a grey hole y in its near vicinity

[1] A node remains unknown to another node until the time it is involved in a predetermined number of mutual interactions.

with $T_{xy} > T_I$. The other mechanism that we have used, is to set the T_I of each node to 100% trustworthy or $T_I = 1$. This ensures that all nodes are judged in due time regarding their capability and past performance. The model then explicitly identifies malicious or trustworthy nodes based upon their current trust levels. Nodes that execute the protocol in a benevolent manner thus maintain a higher trust level than those which are detected as selfish or malicious by the trust model. This mechanism is resilient to the grey hole attack, since any malicious activity by a node drops its trust level in comparison to that of the neighbouring nodes.

Even though, the sending or forwarding nodes can make trust-based routing decisions, the packet will always be sent or forwarded to the next neighbour with the highest direct trust level. Accordingly, in the absence of a trusted next hop the packet will be sent or forwarded onto the path with the minimal hop count (fallback to the native protocol). In order to improve the routing performance in such a situation, we have added a trust thresholding scheme. In this scheme, all sending and forwarding nodes try to minimise the cost by selecting nodes which have a direct trust level equal to or greater than a specified trust threshold (T_T). In the event that there is no hop available with a trust level greater than the trust threshold, a new route discovery is initiated at the source nodes while a local link repair is initiated at the intermediate nodes. Any received data packet at the intermediary nodes, with an unavailable trustworthy next hop, is buffered for a certain interval in the Interface Queue, so as to facilitate the discovery of an alternate route leading to the same destination. If an alternate trustworthy route is found, the packet is sent onto that route.

The trust threshold can be specified in a variety of ways depending on the application. A higher threshold enforces a rigid forwarding criterion that is to be met by all network nodes. Such a threshold is helpful in cases where precise throughput and integrity verification is required. As a result, there is high probability that a traffic saturated node may be incorrectly diagnosed by the trust model as malicious. On

the other hand, a lower threshold provides this requisite leverage and is hence used to detect nodes that portray sustained malicious behaviour.

The size of the trust tables is limited to the number of nodes involved during the definition of any category. However, this number can vary according to the mobility, density and traffic load of the network. Therefore, any node, depending upon its memory and computational constraints, may adopt a purging policy, where the trust entries for nodes portraying sustained benevolent behaviour ($T_{xy} \geq T_T$) may be periodically deleted. In the following subsections, we explain the application of trust to the routing mechanism of each protocol.

6.5.1 Trust Application to AODV

In AODV, before initiating a new route discovery, the routing table is first scanned for a working route to the destination. Only in the event of unavailability of a route from the routing table, the `ROUTE REQUEST` packet is propagated. Thus, when the search is made for a route in the table, the least cost path in terms of number of hops is always returned. Similarly, each forwarding node scans its local routing table to find a least hop path leading to the packet's destination. We have modified this rule and have assigned the direct trust value as the cost of each next hop. Each time a packet is sent or forwarded, the sending or forwarding node first scans its routing table for all alternate paths leading to the same destination. It then compares the direct trust level of all adjacent nodes and selects the highest trust node, with the direct trust level at least equal to or greater than the specified trust threshold. If no such trustworthy node exists, a local repair is initiated. If an alternate trusted route is found, the packet is sent onto the new route. If no such route exists, the packet is dropped and a `ROUTE ERROR` packet is disseminated in the network.

We explain the trust-based routing mechanism of the trusted AODV protocol with the help of Fig. 9. For example, node *35* has received some data from node *14*, which has to be sent to node *50* in the presence of malicious nodes (*23*, *32*, *43*).

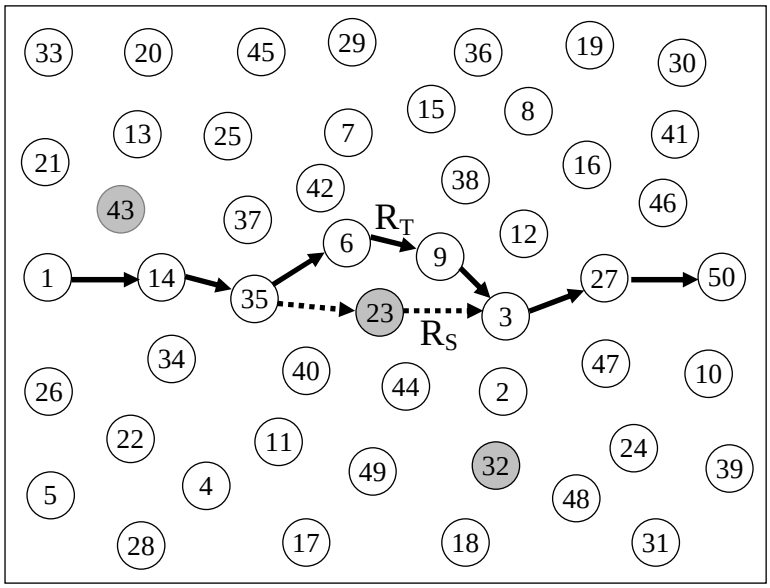


Figure 9: Trust-based Routing in AODV

Node *35* scans its routing table and discovers that it has a route to node *50* via nodes *6* and *23* respectively. In other words, node *35* has forward pointers to these two nodes for a route to node *50*. Node *35*, now based upon prior experiences with the two nodes, opts for node *6* as being more trustworthy than node *23*. If more than one node has exactly the same trust level, then the node having the shortest route to the destination is selected. The intermediate nodes, upon reception of the data packet, make similar decisions and thus the trusted route R_T is developed rather than the shortest route R_S. In the event of no alternate trustworthy next hop available with node *35*, the packet is simply dropped and route maintenance is initiated.

6.5.2 Trust Application to DSR

In DSR, before initiating a new route discovery, the cache is first scanned for a working route to the destination. In the event of route unavailability from the cache, the `ROUTE REQUEST` packet is propagated. In the LINK CACHE scheme the default cost of each link is one. Accordingly, when the search is made for a route

in the cache, the shortest path in term of number of hops is always returned. We modify this rule and associate the computed trust values to the respective links in the cache. Each time a new route is required, the modified Dijkstra [24] algorithm (Appendix G), is executed to find the route with the maximum trust level. This method ensures that malicious nodes with lower trust levels are avoided and bypassed in all subsequent route discoveries.

The DSR LINK CACHE can be modelled as a directed graph $G = (V, E)$ where V represents the set of all known nodes n in the network and E indicates the set of links between these nodes [23]. If $V = \{v_1, v_2, v_3, ...v_{n-1}, v_n\}$ and $E = \{v_1.v_2, v_2.v_3, ...v_{n-2}.v_{n-1}, v_{n-1}.v_n\}$, where $v_x.v_y \in E$ for $x \neq y$ signifies an edge between any two nodes $v_x, v_y \in V$, then the cost of the edge $v_x.v_y$ is represented by $c_{x.y}$.

In the standard DSR, one or more paths can be created between any two nodes depending upon the information available in the LINK CACHE. If the set of i possible paths between nodes $v_a, v_b \in V$ is represented by $P = \{p_1, p_2, p_3, ...p_{i-1}, p_i\}$ then any one path $p_i \in P$ is defined as:

$$p_i := (v_a.v_1, v_1.v_2, v_2.v_3....v_{n_i-2}.v_{n_i-1}, v_{n_i-1}.v_{n_i}, v_{n_i}.v_b)$$

where n_i is the number of disjoint nodes used to create the path p_i. The total cost c_i of path p_i is calculated as follows:

$$\begin{aligned} c_i &= c_{a.1} + c_{1.2} + c_{2.3} + ... + c_{n_i-2.n_i-1} + c_{n_i-1.n_i} + c_{n_i.b} \\ c_i &= \sum_{x=a,y=1}^{n_i,b} c_{x.y} \end{aligned}$$

The current DSR standard defines the cost of each edge to be equal to one. If the total length of path p_i is represented by $p_i^{n_i}$, we can represent the cost c_i as follows:

$$c_i = p_i^{n_i} = \sum_{1}^{n_i} 1$$

Any node sending a data packet tries to lower the cost c_i by minimising n_i, and in doing so, selects the path $p_c \in P$ with the least cost c_c such that:

$$c_c = p_c^{n_c} = min(p_i^{n_i}) \ \forall \ p_i \in P$$

The selection of nodes in p_c is only made at the source nodes and all intermediary nodes simply forward the packet in accordance with the path p_c contained in each data packet. However, depending on the mobility pattern of the network, there may be circumstances in which the source node may not have sufficient trust information regarding all the nodes in the computed path. To deal with such situations, we implement 'Trust Filters' at each node [87]. The distinctive feature of these filters is that they permit traffic to be only forwarded toward nodes, which have a trust level exceeding a specified threshold. This filtering process ensures that all packets destined towards a malicious destination or traversing a malicious node are not permitted to propagate in the network.

With the filtered DSR protocol, the edge costs are replaced with the difference between the maximum possible direct trust level (T_M) of any node and the known direct trust value of the edge destination vertex. The total cost c_i of any path p_i between nodes $v_a, v_b \in V$ is computed as follows:

$$\begin{aligned} c_i &= c_{a.1} \times c_{1.2} \times c_{2.3} \times .... \times c_{n_i-2.n_i-1} \times c_{n_i-1.n_i} \times c_{n_i.b} \\ c_i &= \prod_{x=a,y=1}^{n_i,b} c_{x.y} \end{aligned}$$

If the cost of the edge $v_x.v_y$ is given by $c_{x.y} = \mathrm{T}_M - \mathrm{T}_{x.y}$ then c_i can be represented as:

$$\begin{aligned} c_i &= (\mathrm{T}_M - \mathrm{T}_{a.1}) \times (\mathrm{T}_M - \mathrm{T}_{1.2}) \times (\mathrm{T}_M - \mathrm{T}_{2.3}) \times .... \\ &\quad \times (\mathrm{T}_M - \mathrm{T}_{n_i-2.n_i-1}) \times (\mathrm{T}_M - \mathrm{T}_{n_i-1.n_i}) \times (\mathrm{T}_M - \mathrm{T}_{n_i.b}) \\ c_i &= \mathrm{T}_M^{n_i} \prod_{x=a,y=1}^{n_i,b} \left(1 - \frac{\mathrm{T}_{x.y}}{T_M}\right) \end{aligned}$$

All sending nodes executing the filtered DSR protocol try to minimise the cost c_i by selecting nodes in p_i that have the highest direct trust level ($\mathrm{T}_{x.y}$). As a result, a sending node selects the path $p_c \in P$ with the least cost c_c such that:

$$c_c = min(c_i) \; \forall \; p_i \in P$$

In the event where $c_c = c_i$, then the path with the least length ($p_c^{n_c}$) is selected. Based upon the mobility and traffic pattern it is quite possible that the direct trust value ($\mathrm{T}_{a.y}$) of any node v_y present in p_i is unknown to the source node v_a and is hence set to the initial trust value (T_I). In the filtered DSR protocol the intermediate nodes also contribute to the routing mechanism. Any intermediary node, upon receipt of a data packet with a route that contains a node having a trust level lower than the trust threshold (T_T), rectifies the route and sends the packet on a trusted path along with a `ROUTE ERROR` packet to the source node. For example, an intermediate node v_w receives a data packet from node v_a destined for node v_b with a path p_c. Node v_w now scans its own LINK CACHE and compares the direct trust value of all nodes present in p_c. If p_c contains a malicious node $v_m \in V$ with $\mathrm{T}_{w.m} < \mathrm{T}_T$ then node v_w discards the portion of the path p_c that is beyond itself. v_w now searches its LINK CACHE for an alternate path to node v_b. If it finds another path $p_i \in P$, it creates the new path p_j for the received data packet by concatenating paths p_c and p_i in the following way:

$$p_j = p_c \cup p_i \ : \ \forall \ v_n \in p_i \ , \ \mathrm{T}_{w.n} \geq T_T$$

If there is no such $p_i \in P$, which contains nodes with $\mathrm{T}_{w.n} \geq \mathrm{T}_T$, then node v_w simply drops that data packet and sends a `ROUTE ERROR` packet to the source node v_a. If we consider the scenario shown in Fig. 10.

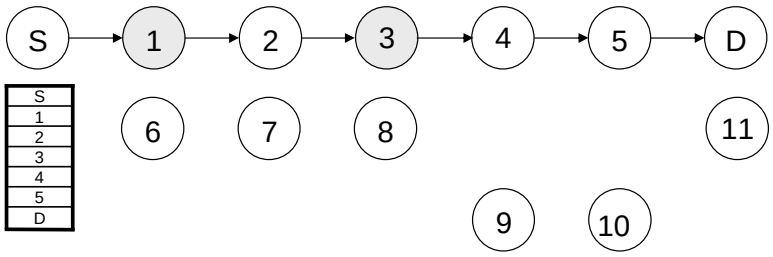


Figure 10: Standard DSR route between Node S and Node D

Assuming that based on prior route discoveries node S has the following link information in its cache:

$$LC_S = \ \{(S \rightarrow 1), (1 \rightarrow 2), (2 \rightarrow 3), (3 \rightarrow 4), (4 \rightarrow 5),$$

$$(5 \rightarrow D), (S \rightarrow 6), (6 \rightarrow 7), (7 \rightarrow 8), (8 \rightarrow 3)\}$$

where LC_S symbolises the LINK CACHE in respect of node S and $(X \rightarrow Y)$ symbolises a working link present in LC_S between node X and node Y. Node S has no link information regarding nodes *9*, *10* and *11*. The possible routes (R_{SD}) that can be built between node S and node D using the available information in LC_S would be:

$$\begin{aligned} R_{SD1} &= (S \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow D) \\ R_{SD2} &= (S \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow D) \end{aligned}$$

When the standard DSR protocol is used then the route R_{SD1} would be selected because its cost (number of hops) is lower than that of route R_{SD2}. Therefore, each time a data packet is sent from source node S to destination node D, the source header, which accompanies each packet, would contain the complete set of nodes present in route R_{SD1}. This implies that all traffic between nodes S and D would have to go via nodes *1*, *2*, *3*, *4* and *5* respectively to reach the destination node D. This indirectly facilitates malicious nodes *1* and *3* to launch a variety of attacks against the traffic that is passing through them. In addition, the route R_{SD1} would always comprise the same set of nodes until either node S, D or any of the intermediate nodes forming the route R_{SD1} move out of transmission range of the other nodes.

However, when the trust model is executed by the nodes, based upon the mobility of the network, each node undergoes a number of experiences with its immediate neighbours[2]. Considering node S was able to correctly identify the malicious behaviour of node *1* then node S would have a lower trust rating in respect of node *1* in its trust table. Consequently, node S prefers route R_{SD2} over the shorter route R_{SD1} due to the higher trust rating of the route R_{SD2}, as shown in Fig. 11.

In a similar way malicious node *1* is bypassed in all subsequent data connections

[2]The immediate neighbours of any node act as gateways to the ad-hoc network. All inbound traffic coming from the network has to flow through these gateways before it reaches any node. Similarly, all outbound traffic has to pass through these gateways in order to reach the network.

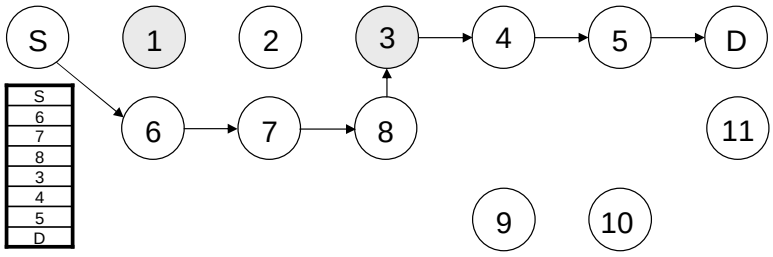


Figure 11: Source Computed Trusted DSR route between Node S and Node D

to other nodes in the network. However, there may be other nodes present in the network whose behaviour has not yet been evaluated by the trust model of node S for example node *3*. Such nodes are considered neutral and remain involved in the routing process until the time they are evaluated directly by a node. Based upon the mobility pattern, this evaluation may or may not take place during the network lifetime. Therefore, if such malicious nodes are not evaluated by the source node's trust model, they will be involved in the routing process, and can so launch attacks against the network traffic.

The traffic filters employ a mechanism similar to salvaging where, instead of checking only the connectivity of the next hop, they also verify the trust levels of all nodes present in the packet's source route header. When the standard DSR or the source computed trust-based routing scheme is used, all intermediate nodes blindly forward the packets to the nodes as listed in the source route. For example if node *8* (in Fig. 11) receives a data packet from node S, it should forward the packet to the next node *3* as given in the source route. However, with a trust filter implemented at node *8*, it first verifies the trust level of all the remaining nodes in the source route and then makes the appropriate routing decision.

As shown in Fig. 12, when node *8* receives the packet from node S with route R_{SD2}, it first verifies the trust levels of all the nodes beyond itself ($3 \rightarrow 4 \rightarrow 5 \rightarrow D$) listed in the accompanied source route. During this verification, it finds that node *3* has a lower trust value than the specified trust threshold. Node *8* now scans its LINK CACHE to find an alternate route to the destination using the modified Dijkstra

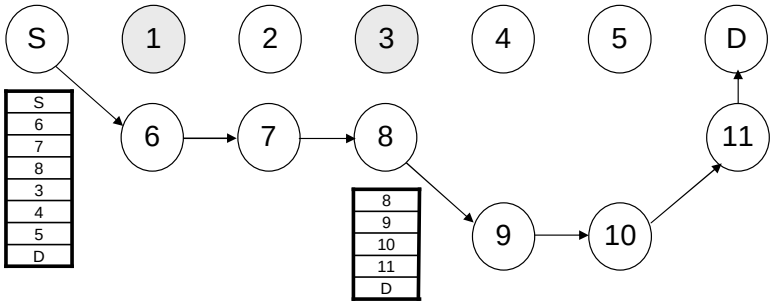


Figure 12: Filtered DSR route between Node S and Node D

algorithm. If such a route is found, the source route header of the data packet is modified accordingly and the packet is forwarded on the newly discovered route ($8 \rightarrow 9 \rightarrow 10 \rightarrow 11 \rightarrow D$). In the event that node *8* cannot find an alternate route to the destination, it may initiate a local route discovery or drop the packet along with a corresponding `ROUTE ERROR` to inform node S. However, the number of such salvage operations is limited to the **MAX_SALAVGE_COUNT**[3] [41]. The major benefit of these filters is that the incoming data packets are not dropped en-route, but only redirected to their actual destination through an alternate trustworthy path.

6.5.3 Trust Application to TORA

In TORA, as a result of each route discovery, the immediate neighbours inform the querying node of the destination node's height. Accordingly, each node also maintains a separate list of immediate neighbours for each known destination. This list consists of immediate neighbours that have downstream links and each may indicate a different height to the same destination. Thus, whenever a data packet needs to be forwarded to any destination, the forwarding node looks for the least height entry among this list of neighbours before initiating a new route discovery. The data is simply forwarded to an immediate neighbour, which had earlier indicated the least height of the destination. In the event of unavailability of a route

[3]Currently set to fifteen to avoid the packet traversing a never ending salvaging loop

from the routing table, the `QRY` packet is propagated.

We have modified the TORA height structure by adding T_{xy} as a sixth tuple to the existing height information [92]. Each height is represented as (τ, oid, r, T, δ, x), where the first four values represent the reference level and the last two represent the change with respect to the reference level. The linkage between the node heights and their corresponding direct trust values (T) enables a node to retrieve a route with a higher trust level rather than a route with the least height. In the event of similar trust levels between any two nodes, the other height components are taken into consideration. This mechanism ensures that if trustworthy routes are available then the malicious nodes with lower trust levels are avoided in succeeding data connections.

If (τ_y, oid_y, r_y, T_{xy}, δ_y, y) signifies the height H_y of node y determined by node x, N_{xi} represents all immediate neighbours i of node x and the link status between node x and node i is represented by $\mathrm{L}_{xi} \in \{0, 1\}$, then the selection of the most trustworthy next hop (tn_{min}) is undertaken as per the following rule:

$$
\begin{aligned}
&for\ all\ y\ \in\ \{\mathrm{N}_{xi}\ \mid\ \mathrm{H}_y \neq (-,-,-,\mathrm{T}_{xy},-,y)\ \ \wedge\ \ \mathrm{L}_{xy} = 1\}\\
&do\\
&\quad if\ \ (\mathrm{T}_{xy}\ >\ \mathrm{T}_{xi})\ \vee\ \ \backslash\\
&\quad ((\mathrm{T}_{xy}\ =\ \mathrm{T}_{xi})\ \wedge\ ((\tau_y, oid_y, r_y, \mathrm{T}_{xy}, \delta_y, y)\ <\ (\tau_i, oid_i, r_i, \mathrm{T}_{xi}, \delta_i, i)))\\
&\quad then\ \ tn_{min}\ =\ y\\
&done\\
&return\ tn_{min}
\end{aligned}
$$

Node x compares the heights between node y and node i in the following order:

$$
\begin{aligned}
&\quad if\ \ (\mathrm{T}_{xy}\ >\ \mathrm{T}_{xi})\ \ return\ \mathrm{TRUE}\\
else\quad &if\ \ (\tau_y\ >\ \tau_i)\quad return\ \mathrm{TRUE}\\
else\quad &if\ \ (oid_y\ >\ oid_i)\ return\ \mathrm{TRUE}\\
else\quad &if\ \ (r_y\ >\ r_i)\quad return\ \mathrm{TRUE}
\end{aligned}
$$

$$
\begin{array}{lll}
else & if\ (\delta_y > \delta_i) & return \text{ TRUE} \\
else & if\ (y > i) & return \text{ TRUE} \\
 & return \text{ FALSE} &
\end{array}
$$

We exploit the local neighbourhood trust information in TORA to influence the routing decisions made by the forwarding nodes in the network. Before forwarding any packet, the node first verifies whether the trust level of the next hop is the highest among the available immediate neighbours leading to a particular destination. If this is confirmed the packet is forwarded accordingly. However, if the selected node has a trust level lower than the minimum defined trust threshold, the forwarding node queues the packet and initiates a new route discovery. If an alternate route is found, the packet is transmitted on the new path. In the event that no alternate route is found through route discovery, the packet is dropped.

6.6 Summary

In this chapter we have explained the different situational trust categories, which can be derived using the three routing protocols. These situational trust categories are then assigned weights and combined to form the direct trust in other nodes. These trust values can be shared in the network using an independent or integrated reputation exchange protocol. The former, although directly implementable in all routing protocols, induces excessive packet overhead during the exchange of reputations, while the latter needs to be independently adapted and configured for each routing protocol.

The association of direct trust levels to the nodes, either in the routing tables or cache, ensures that if alternate trustworthy routes are available then the malicious nodes with lower trust levels are avoided and bypassed in all ongoing and subsequent data connections. In contrast to AODV and TORA, the trust-based route selection in DSR can only be carried out at the connection originating stage and not at the

forwarding stage. This is due to the fact that the intermediary nodes, participating in any active data connection, are able to derive trust in other nodes but cannot make trust-based routing decisions for that particular connection. However, these nodes can subsequently use the derived trust information when they act as source nodes and initiate new data connections. To overcome this inherent limitation, we have supported the DSR protocol with the help of trust filters, which enable source initiated as well as intermediary trust-based routing.

In all three protocols, the route selection is made dependent upon the direct trust of the selected path (in case of DSR) or the selected next hop (in case of AODV and TORA). This enables nodes to forward packets onto trustworthy routes, rather than the default least hop routes. This trust-based routing scheme thus permits benevolent nodes to evade malicious and selfish nodes in the network. In the next chapter, we determine the effectiveness of the trust-based routing protocols with the help of exhaustive simulations and carry out an analysis of the obtained results.

Chapter 7

Results and Analysis

In this chapter, we employ extensive simulations to assist in the evaluation of the three trusted routing protocols in a hostile network. We first explain the implementation details for the trusted routing protocols in Section 7.1. These details include the computation of weights for the trust model; the implementation methodology; the mobility and communication models; the attack profile and the assumptions made for implementation purposes. In Section 7.2 we explain the simulation environment and the monitored metrics. A series of tests are then carried out in Section 7.3, which evaluate the performance of the trust model under diverse mobility and attack conditions. The performance of the trusted protocols is also compared with some related previous work in this section. This chapter concludes with an analysis of the simulation results in Section 7.4.

7.1 Implementation Details

7.1.1 Computation of Weights

The assignment of weights to different trust categories was undertaken using a trial and error mechanism [90]. The weights for each category were initially adjusted

according to the average number of such packets received per node. These weights were then varied in both directions, so as to increase the overall throughput of the network with minimal packet loss. The optimal situational weights that are assigned to different situational trust categories are shown in Table 14.

Table 14: Optimal Situational Weights

Situational Trust Category	Weights	Protocol		
		AODV	DSR	TORA
Passive Acknowledgements	$W(P_A)$	0.2225	0.2225	0.2225
Packet Precision	$W(P_P)$	0.6675	0.6675	0.6675
Gratuitous Route Replies	$W(G_R)$	0.04	0.04	-
Salvaging	$W(S_G)$	-	0.07	-
Blacklists	$W(B_L)$	-	-	-
Beacon/HELLO Messages	$W(H_M)$	0.05	-	0.11
Destination Unreachable Messages	$W(D_U)$	0.02	-	-
Authentication Objects	$W(A_O)$	-	-	-

The situational trust category Authentication Objects (A_O) was not evaluated, as the simulation was conducted in a pure ad-hoc network environment. As seen from the table, the Passive Acknowledgements (P_A) and the Packet Precision (P_P) situational trust categories have maximum impact on the derivation of the direct trust in another node. As the MAC protocol is expected to provide feedback (like IEEE 802.11), it implies that the links are bi-directional, and so the category B_L can be safely ignored. The impact of the other categories G_R, S_G, H_M and D_U is also marginal upon the trust derivation process. In order to observe the effect of a common category in all three routing protocols and to reduce the computational complexity by limiting the number of recorded events, we have used the accuracy and quantity of accurate packets forwarded (P_A, P_P) by a neighbouring node as the measure of its direct trust. The categories P_P and P_A are employed, in combination, to protect the protocol against deceptive alteration of vital protocol fields and for identifying selfish node behaviour, respectively.

7.1.2 Methodology

After transmission, each node promiscuously listens for its neighbouring node to forward the packet. If the neighbour forwards the packet in the proper manner (correct hash including any required modification), its corresponding trust level, which is being maintained in a table, is incremented. However, if the neighbouring node modifies the packet in an inappropriate manner or does not forward the packet, its trust level is decreased. In our implementation, a node only maintains the number of failed or successful events of an event type. For example, if node q is a neighbour of node p, and p is monitoring the behaviour of q with respect to packet modification, p needs to keep track of only two numbers, (i) number of times q fallaciously modifies a packet, and (ii) number of times it broadcasts a packet with consistent or no modification. The ratio of these two numbers translates into the direct trust of p in q with respect to packet modification. Hence, p does not need to keep track of each event explicitly. There is one situation when an event needs to be kept in the buffer. Suppose p is monitoring q for black or grey hole attacks. p sends a packet to q and then holds the packet in a buffer and waits for q to forward it. p can listen to q's forwarding in promiscuous mode. Hence, the packet is held for the Trust Update Interval (TUI) in p's buffer to compare with q's subsequent transmissions. The TUI essentially represents the time a node should wait before clearing the send buffers and assigning the trust or distrust level to the subsequent node based upon the accuracy and timeliness of the observed forwarding event. If a packet is overheard before the TUI expires, it is verified for its integrity and the buffers are immediately cleared. As each node only needs to keep a small number of such packets in the buffer at any instant, the likelihood of a buffer overflow is extremely low.

7.1.3 Mobility Model

We employ the random way point movement model [7] for all simulations, in which a node first waits for the pause interval and then moves to a randomly chosen

position with a velocity chosen between 0 m/s to the maximum speed, waits there for the pause time, and then moves on to another random position. A pause time of 0 seconds implies continuous mobility where as a pause time equal to or greater than the simulation time means a static network. Similarly a maximum speed of 0 m/s also equates to zero mobility in the network.

7.1.4 Communications Model

The IEEE standard 802.11 Distributed Coordination Function (DCF) [37] is used as the MAC layer for the three routing protocols. All `ROUTE REQUEST` and `QRY` packets are broadcast using the un-slotted Carrier Sense Multiple Access protocol with Collision Avoidance (CSMA/CA). In CSMA/CA each broadcasting node waits for a vacant channel by sensing the medium. If the channel is vacant, it makes the transmission. In the event of a collision, the colliding stations wait using the Ethernet binary exponential back-off algorithm. To unicast packets, the node first reserves the channel by transmitting a short Ready-to-Send (RTS) frame. The intended recipient node, in response, sends a Clear-to-Send frame to the RTS sender. All nodes overhearing the RTS or CTS frames desist from transmitting for the Network Allocation Vector (NAV) interval. Upon receipt of the CTS, the packet is transmitted which is acknowledged by the recipient [105].

7.1.5 Attack Pattern

Malicious and selfish nodes simulate the following types of attacks in the network:

Modification Attack: These attacks are carried out by adding, altering or deleting IP addresses from the `ROUTE REQUEST`, `QRY`, `ROUTE REPLY`, `UPD`, `ROUTE ERROR`, `CLR` and Data packets, which pass through the malicious nodes.

Black Hole Attack: In this attack the malicious node drops all data packets, which it is supposed to forward. However, it participates in the route discovery

process, which is initiated by other nodes, so as to remain on the path of the data connections.

Grey Hole Attack: The grey hole attack is similar to the black hole attack, however, the malicious node also selectively forwards data packets at random intervals.

7.1.6 Legitimate Packet Loss

In addition to malicious packet drop, data packets can also be lost by any node due to the following reasons:

MAC Layer Collisions: All three protocols do not guarantee packet delivery, and so data packets are not buffered for retransmission. In the event of a collision involving a data packet, the packet is simply considered lost. The responsibility of retransmission of the packet is left to the higher layers in the protocol stack.

Saturation of Interface Queues: AODV, DSR and TORA implement Network Interface Queues (IFQ) to buffer packets, which are ready to be transmitted and are received by the network protocol stack. These IFQ generally limit the maximum number of packets that can be held and may also implement a maximum timeout policy for packets in the IFQ. As a result, any packet awaiting a route in the IFQ for an extended period, may simply be discarded without any notification.

The legitimate packet drops are influenced by the mobility pattern of the network and accordingly influence the different performance metrics of the network. This is confirmed by the fact that the throughput of the three protocols always remains lower than 100% even when no malicious nodes are present in the network. However, the ratio of legitimate to deliberate packet drops is negligible, as will be confirmed by the results.

7.1.7 Assumptions

It is possible that a node may spoof its IP or MAC address in order to steal a data connection or to deceive the trust model. Such an activity, which may not be directly perceivable to the neighbouring nodes, is still detectable if a MAC to IP address binding is maintained at each node. Each time an IP address corresponding to a MAC address is changed, it is considered as a modification attack, and so the spoofing node can be graded untrustworthy by its adjacent nodes. However, in our simulations, we have assumed that the malicious nodes do not carry out spoofing attacks and that the MAC to IP bindings remain consistent.

Collusion attacks, like wormholes, can be detected to some extent in a source routing protocol like DSR, where the modification of packets to create a tunnel can be observed by the category packet precision [89]. However, such detection will not work in a hop-to-hop routing protocol like AODV or TORA. In order to evaluate the precise impact of similar types of attacks upon the three reactive routing protocols, we have assumed that all malicious nodes work in a non-colluding manner. However, each malicious node sporadically alters its attack profile by randomly switching between the three types of attacks. All nodes, except for the malicious nodes, execute the trust model for the network lifetime.

For accurate derivation of trust, the participating nodes also need to support the following features:

- Promiscuous mode operation
- Omnidirectional transceivers
- Comparable transmission and reception range of transceivers

7.2 Simulation Environment

7.2.1 Setup

The Network Simulator (NS-2) [65], along with the CMU ad-hoc networking extensions [14], was used to evaluate the performance of the three trust-based routing protocols under attack conditions [93]. The relevant pseudo-code for the trusted AODV, DSR and TORA routing protocols is given in Appendices H, I and J respectively. To avoid confusion, we henceforth refer to the trust-based AODV, DSR and TORA protocols as just the AODV, DSR and TORA protocols respectively. The simulation parameters are listed in Table 15.

Table 15: Simulation Parameters

Examined protocols	AODV, DSR and TORA
Simulation time	900 seconds
Simulation area	1000 x 1000 m
Number of nodes	50
Propagation model	Two-ray Ground Reflection
Transmission range	250 m
Transmission power consumption	281.8 mW
Reception power consumption	281.8 mW
Movement model	Random waypoint
Maximum speed	20 m/s
Maximum pause time	1000 seconds
Traffic type	CBR (UDP)
Maximum connections	10, 20 & 30
Payload size	512 bytes
Packet rate	4 pkt/sec
Maximum malicious nodes	20
Trust Update Interval	5 seconds
Types of attacks	Modification, Black and Grey hole
Black/Grey hole attacks W(P_A)	0.25
Modification attacks W(P_P)	0.75

7.2.2 Metrics

To evaluate the performance of the protocols, we use the following metrics:

Packet Loss: The total number of data packets lost legitimately or through malicious action without any notification.

Packets Forwarded: The number of data packets that were forwarded successfully by the intermediary nodes.

Throughput: The ratio between the number of data packets received by the application layer of destination nodes to the number of packets sent by the application layer of source nodes.

Routing Packet Overhead: The ratio between the total number of control packets generated (excluding `HELLO` and `BEACON` packets) to the total number of data packets received during the simulation time.

Average Latency: The mean time in seconds taken by the data packets to reach their respective destinations.

Path Optimality: The ratio between the number of hops in the optimal path to the number of hops in the path taken by the data packets.

Energy Consumption: The amount of energy (in Joules) consumed per node during the simulation time.

Probability of Detection: The ratio between the number of nodes whose behaviour (malicious or benevolent) is identified correctly to the actual number of such nodes present in the network.

7.3 Results and Discussions

The following tests were conducted to evaluate the performance of the three protocols under varying attack and mobility conditions:

Test 1: Comparison between Trusted and Standard Routing Protocols

Test 2: Varying the number of malicious nodes

Test 3: Varying the node pause times

Test 4: Varying the node maximum speeds

Test 5: Varying the trust update interval

Test 6: Comparison between Trusted AODV and ARAN

Test 7: Comparison between Trusted and Pathrated DSR

Test 8: Reputation exchange with varying number of malicious nodes

Test 9: Reputation exchange with varying node maximum speeds

In Tests 1 to 7, we evaluate the trusted routing protocols, which exclusively make use of the direct trust to influence the routing process. We then integrate reputation exchange mechanisms in these protocols and assess their performance in Tests 8 and 9. Each test was carried out under multiple traffic loads with the number of connections set to 10, 20 and 30. Accordingly, AODV with ten sources is represented by AODV-10, DSR with twenty sources by DSR-20, and TORA with thirty sources by TORA-30 respectively. To get an accurate picture, each protocol is tested against exactly the same scenario and connection pattern. The performance metrics are obtained through ensemble averaging [117] over 100 simulations, each with a different mobility and connection pattern.

7.3.1 Test 1 : Comparison between Trusted and Standard Routing Protocols

In Test 1, we have used the parameters as listed in Table 16.

Table 16: Test 1 Specific Simulation Parameters

Number of Connections	10
Maximum speed	20 m/s
Pause time	10 seconds
Minimum malicious nodes	0
Maximum malicious nodes	20

The results of Test 1 (shown in Fig. 13) highlight the effectiveness of the trusted AODV, DSR and TORA routing protocols in comparison with their standard counterparts. The results indicate that the packet loss in the standard protocols is up to 20% higher than that in the trusted routing protocols. Nodes, which execute the standard routing protocols, cannot differentiate between malevolent and benevolent nodes and hence select shortest possible routes. These routes, if containing malevolent nodes, induce higher packet loss in the network. However, nodes executing the trust model, which have been successful in detecting malicious node behaviour, bypass the malevolent nodes in the routing process and hence lower the packet loss. The total number of packets forwarded using the standard routing protocols also remains lower than that of the trusted protocols.

Nodes, executing the trusted TORA, exhibit the highest forwarding rate, followed by trusted AODV and DSR protocols respectively. The higher forwarding rate of the trusted protocols has a positive effect on the throughput of the network. Trusted AODV augments the throughput of the standard AODV by up to 20%, trusted TORA up to 18.5% and trusted DSR up to 11.8%, in the presence of 40% malicious nodes. The higher packet delivery rate of the trusted protocols also keeps the routing overhead lower, which is computed per received data packet. However, as the trusted protocols endeavour to find the most trusted paths in the network, the selected paths may sometimes deviate considerably from the optimal paths. This increases the length of the paths, thereby increasing the latency of the network.

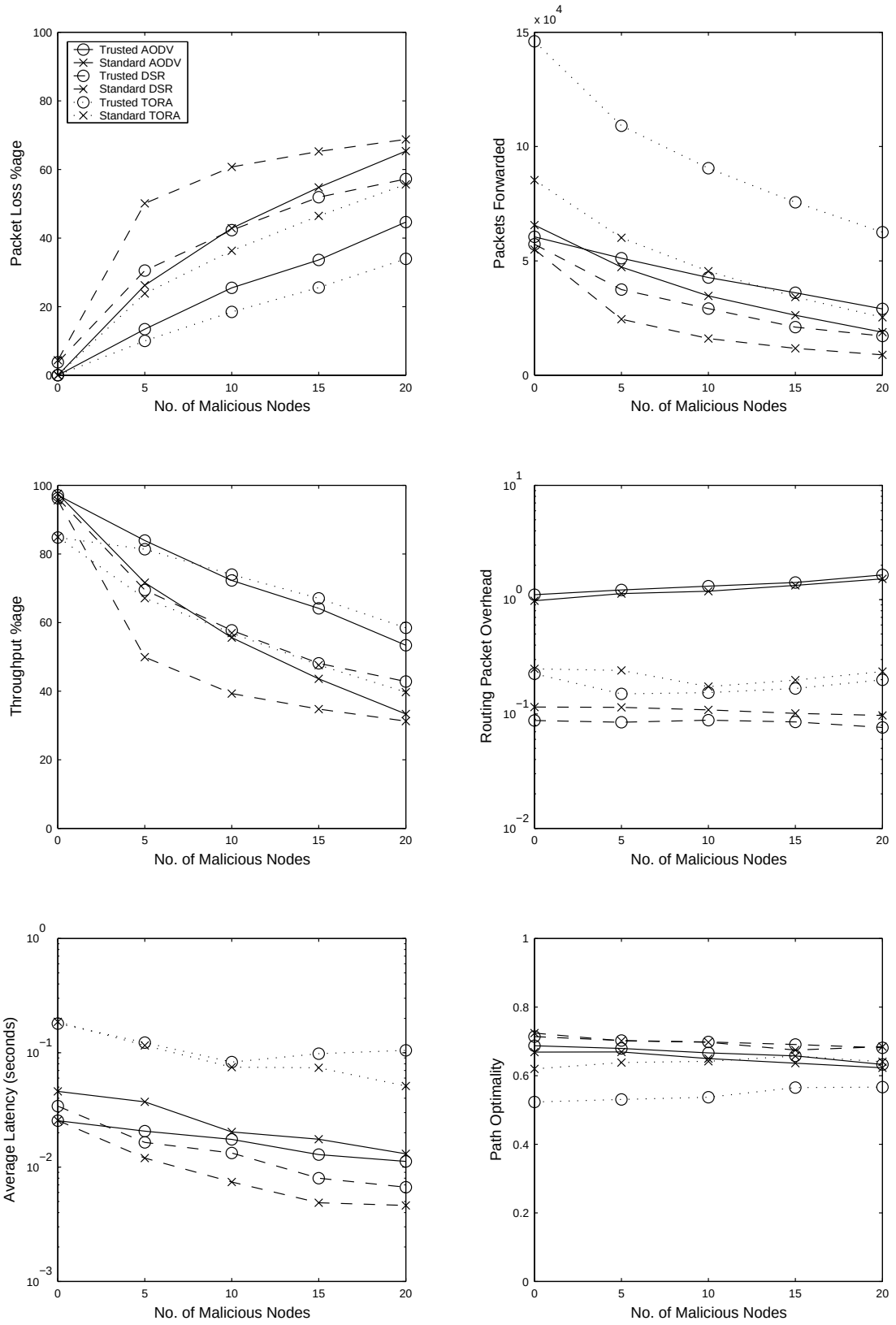


Figure 13: Test 1 : Performance Results

7.3.2 Test 2 : Varying Number of Malicious Nodes

The parameters listed in Table 17 were used in Test 2.

Table 17: Test 2 Specific Simulation Parameters

Maximum speed	20 m/s
Pause time	10 seconds
Minimum malicious nodes	0
Maximum malicious nodes	20

The results of Test 2 (shown in Fig. 14 and 15) indicate that in the absence of malicious nodes, the legitimate packet loss is about 1% for AODV and TORA protocols and about 3% for DSR. The higher packet loss for DSR is primarily due to its specific working in which multi-path selection can only be made at the source node. Consequently, if a `ROUTE ERROR` is propagated, all intermediary nodes drop the packets present in their IFQ. However, intermediary nodes using AODV and TORA are able to re-route data packets, and are so able to minimise the packet loss. TORA makes effective use of its inherent multi-path feature and is hence able to forward a large number of packets at all traffic loads with minimal loss.

The throughput of all three protocols rapidly drops with the increase in the number of malicious nodes. The rate of this drop is the highest for DSR-10, which degrades from 96% with no malicious nodes to about 42% with 40% malicious nodes. The throughput of AODV drops from 97% to 53% for a similar increase in the number of malicious nodes. The TORA-10 protocol degrades from an 85% throughput to about 58%. However, the increase in the number of sources causes TORA-20 and TORA-30 to undergo a congestive collapse [8]. This is essentially due to the positive feedback loop created in TORA/IMEP due to the increased number of MAC layer collisions. These collisions, incorrectly make IMEP believe that the links to adjacent nodes are severed. In response TORA generates more `UPD` packets, which closes a serviceable link that is temporarily congested. This leads to generation of further `QRY` packets to find alternate routes despite the availability of working routes. This increased control packet overhead, essentially closes the feedback loop

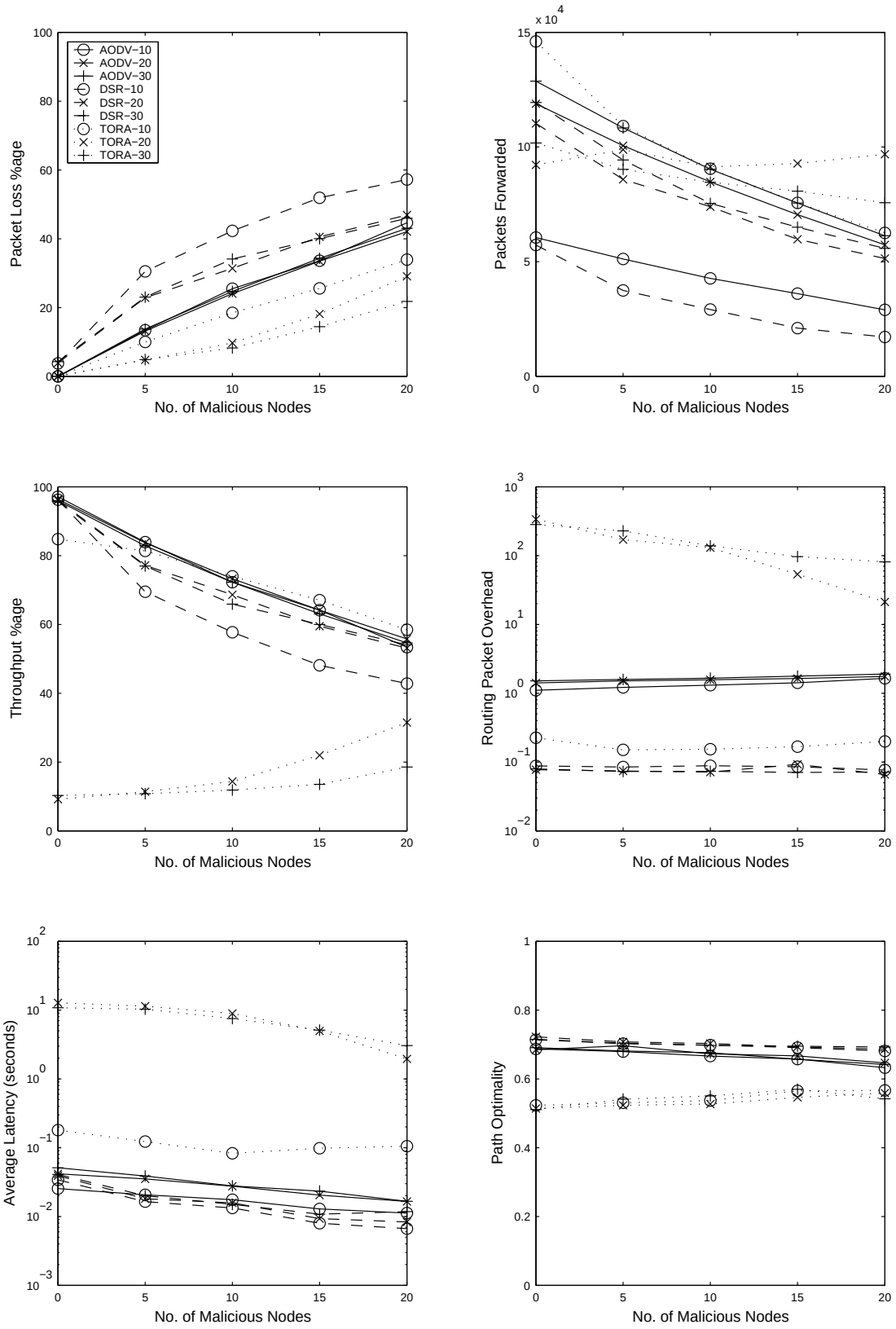


Figure 14: Test 2 : Performance Results

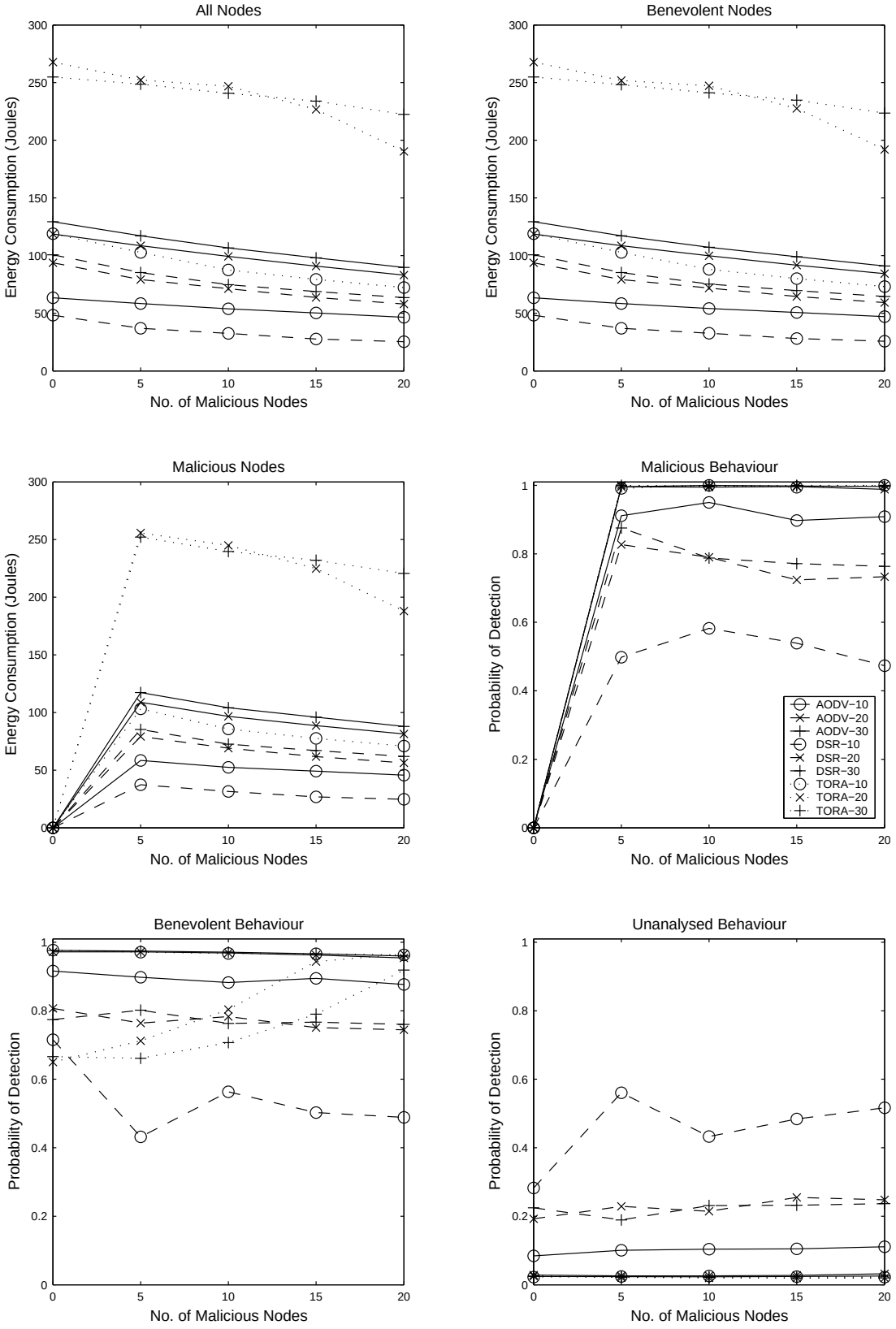


Figure 15: Test 2 : Energy and Detection Results

causing further congestion. On the other hand, the malicious nodes in the network, which drop the network traffic, aid TORA in recovering from this phenomenon by reducing the traffic load and thereby the number of MAC layer collisions. This indirectly improves the performance of the TORA-20 from 10% with no malicious nodes to about 30% with 40% malicious nodes. The performance of the TORA-30 protocol shows improvement by up to 10% with the increase in malicious nodes.

The routing overhead of all three protocols remains significantly constant with the increase in the number of malicious nodes. AODV, on the average, generates one control packet for each received data packet, DSR generates one for every twelve received data packets while TORA-10 generates one control packet for every five received data packets. The increased control packet overhead in AODV and TORA is primarily due to their route discovery mechanism that requires the `ROUTE REQUEST` or `UPD` packet to be broadcast over the network, whereas the DSR protocol makes effective use of its inherent caching strategy to limit this routing overhead. The control packet overhead of TORA-20 and TORA-30 remains exponentially high when no malicious nodes are present and drops when the malicious nodes in the network inadvertently support the protocol by lowering the amount of traffic in the network.

The latency of the network remains minimal for the DSR protocol, where trust-based routing decisions are only made once upon the initiation of the data connection. The trust-based AODV and TORA protocols, on the other hand, have to make such decisions at the source as well as all intermediary nodes. This indirectly increases the delay in the packet traversal time. The latency of TORA-20 and TORA-30 remains higher due to the data packets going into extended routing loops.

The path optimality of the network with all three protocols remains uniform with the increase in the number of malicious nodes. The data packets, which finally do reach their intended destinations in the presence of malicious nodes, have traversed the path that contains no black hole but may or may not contain a grey hole.

The path optimality of the DSR protocol remains higher due to the optimal path selection at the source node with limited redirection by the intermediary nodes executing trust filters. In contrast, nodes executing the trusted AODV and TORA protocols make routing decisions for each forwarded data packet, and so the path diverts from the available shortest possible path. TORA, due to its link reversal mechanism, prefers local route fixation over discovery of alternate optimal routes in the network. In doing so, TORA lowers the routing overhead due to extraneous route discoveries but loses upon the path optimality of the network.

The energy consumption of the nodes is primarily affected by the throughput and the control packet overhead of the network. However, we observe that the energy consumption of the nodes decreases with the increase in the number of malicious nodes. This is primarily due to the low throughput of the network, which indirectly reduces the number of transmissions and receptions. Nodes executing the trusted AODV, DSR and TORA protocols require sustained promiscuous mode receive operation for the TUI and receive all MAC frames whether or not they are destined for that node [27]. However, this increase in the energy consumption affects all three protocols in a somewhat similar manner. The exorbitant routing overhead of the TORA-20 and TORA-30 protocols adds to the energy consumption of both benevolent and malicious nodes. The silent dropping of data packets, which acts as an energy saver, relieves the selfish nodes of transmission costs. Consequently, the selfish nodes, have been able to save energy as compared to the benevolent nodes that participate diligently in the operation of the network.

As the test was carried out at a relatively higher mobility of 20 m/s, the nodes had frequent interactions with each other. This enabled the trust levels of nodes to be evaluated by their neighbouring nodes. The probability of detection improves with the traffic loads and we see that most of the malicious nodes were successfully detected by one or more nodes executing the trusted protocols. The higher probability of detection in the AODV and TORA protocols is attributable to their specific working, in which intermediary nodes make trust-based routing decisions at

the forwarding stage to create reliable but longer routes. This, although lowers the path optimality, permits increased interactions with unknown nodes in the network. DSR, on the other hand, selects optimal paths only at the source node. The source node may continue to do so until the time it encounters direct interactions with an intermediary malicious node present on the path to the destination. Similarly during the simulation there were few nodes whose trust level was not evaluated until the end of the simulation period. These nodes were not involved in any of the data connections and hence their behaviour was not evaluated by other nodes in the network.

7.3.3 Test 3 : Varying Node Pause Times

In Test 3, we have used the parameters as listed in Table 18.

Table 18: Test 3 Specific Simulation Parameters

Maximum speed	20 m/s
No. of malicious nodes	20
Minimum pause time	0 seconds
Maximum pause time	1000 seconds

The results of Test 3 (shown in Fig. 16 and 17) indicate that the throughput of all protocols degrades with the increase in pause time. Higher pause times essentially limit the number of interactions with other network nodes. This in turn reduces the detection probability, permitting selection of malicious nodes for routing purposes. The malicious nodes are thus able to increase the packet loss, thereby reducing the throughput of the network. The maximum throughput of the protocols is achieved at 0 pause time (constant mobility) with 40% malicious nodes in the network. AODV and DSR maintain an unvarying throughput under different pause times with dissimilar traffic loads. The TORA-10 protocol has the highest throughput under varying pause times while TORA-20 and TORA-30 portray degraded performance at lower pause times due to the creation of positive feedback loops.

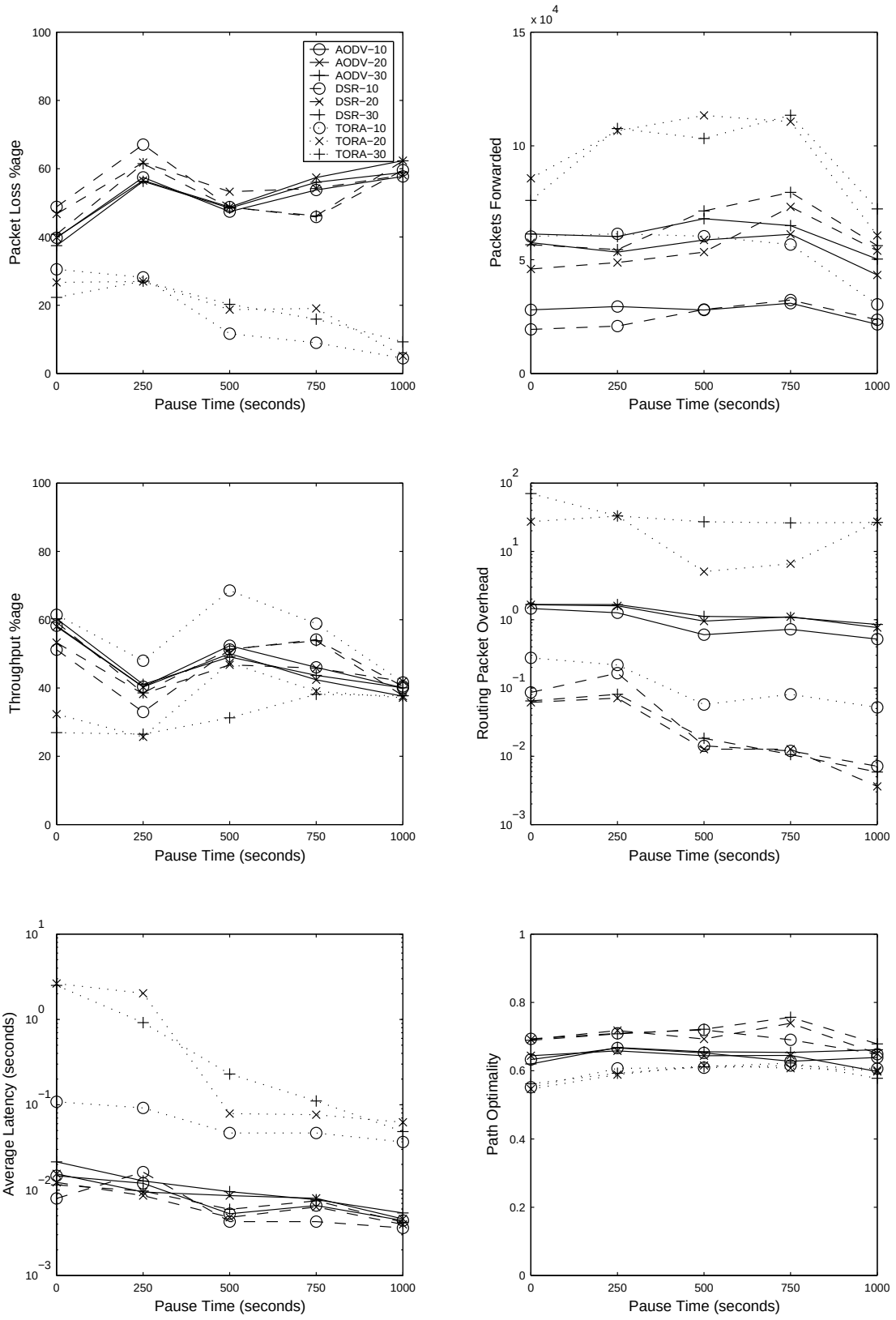


Figure 16: Test 3 : Performance Results

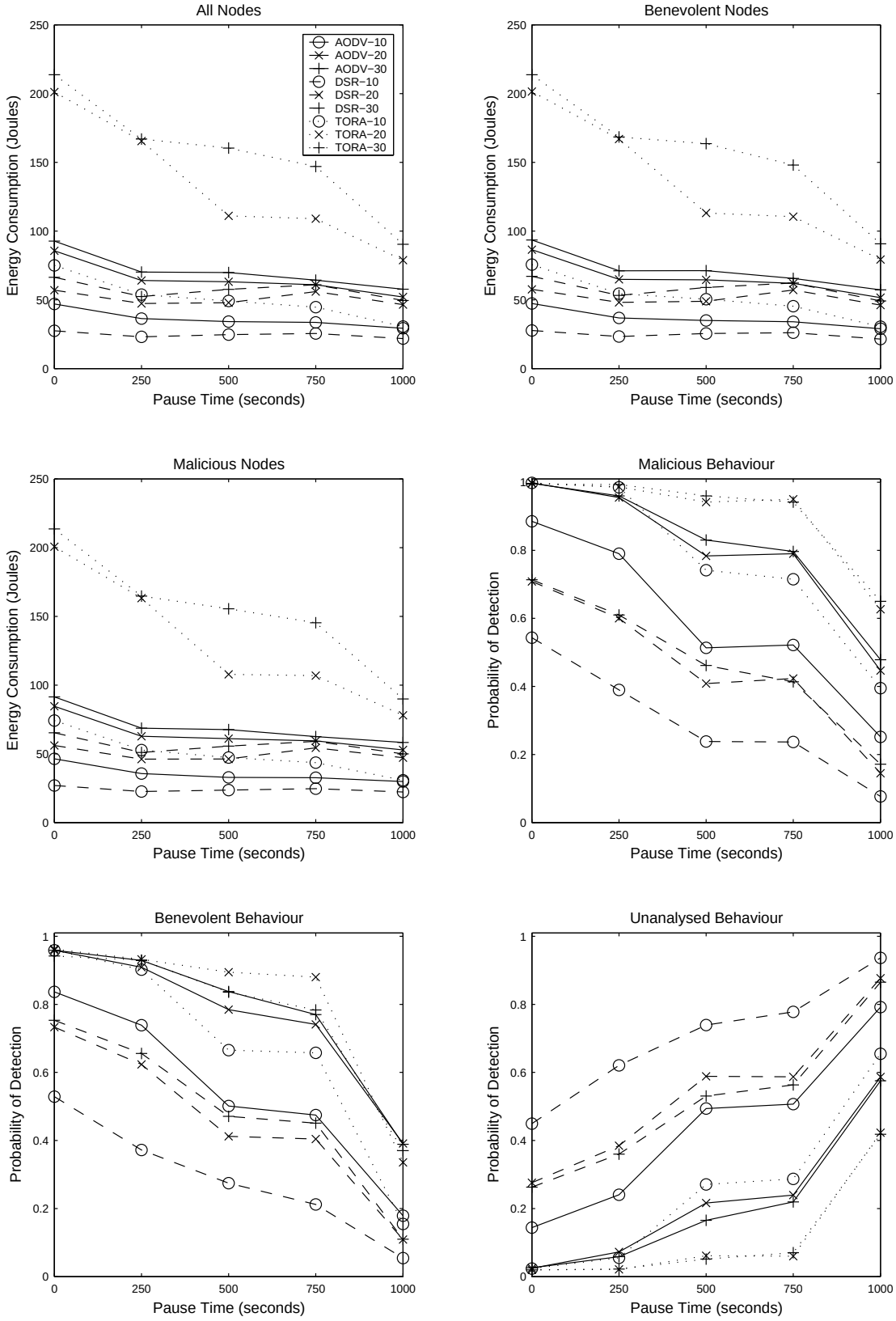


Figure 17: Test 3 : Energy and Detection Results

At higher pause times, the frequency of link creation and breakage reduces significantly. This ensures that the data packets make it to their destinations without being queued for a long time at the source or intermediary nodes, awaiting route discoveries. Thus the routing packet overhead for all three protocols shows a gradual downward trend with increasing pause times. At lower pause times, TORA-10 limits its route discoveries to the local region [49], which in turn limits the overall routing packet overhead. AODV and DSR also generate less overhead with increasing pause times due to the stabilisation of topology, which enables the routes to last longer. This permits nodes to send or forward data packets without initiating new route discoveries. Thus the data packets can be directly transmitted on previously known routes without any noticeable delay, which helps in lowering the overall latency of the network. All three trusted routing protocols aim to increase the aggregate trust level and decrease the number of hops in the path between the source and destination. As a result, when the nodes pause for large intervals, the least cost paths are selected, which are comparable with the shortest possible paths. Thus we observe an improvement in the path optimality with the increase in node pause times.

The energy consumption of the benevolent nodes lowers with the increase in pause times. At higher pause times, the routes in the table or cache remain comparatively stable, minimising the number of additional route discoveries. This helps to lower the routing overhead, which in turns improves the energy consumption. Selfish nodes also undergo the same routing overheads but are able to slightly save upon their energy by not forwarding the data packets. As seen in the results, nodes executing the DSR-10 and AODV-10 protocols consume minimal energy under varying pause times mainly due to their lower packet overhead.

The probability of detection of all three protocols increases at lower pause times primarily due to the increased number of interactions. TORA depicts the highest detection probability even when the network is virtually static. This is due to the fact that TORA retains multiple paths to a single destination, and is able to

evaluate the trust of different neighbouring nodes more effectively. AODV also works in a similar manner but has limited available paths to a destination due to certain optimisations that avoid routing loops. DSR has the least detection capability due to the limited perspective of the source nodes. Each source node, using DSR, selects the most trusted nodes in the path based upon the currently available information. Generally, this information is restricted to only a few nodes in the neighbourhood. Thus, the source node may create a route containing one or more malicious nodes. As the immediate nodes can only make limited routing decisions using trust filters, the data connections continue to traverse the fixed set of nodes, which were initially selected by the source node. This essentially limits the number of nodes that interact with each other, thus lowering the probability of detection. The lower detection probability also suggests selection of optimal paths in terms of number of hops (fall back to native DSR).

7.3.4 Test 4 : Varying Node Maximum Speeds

The parameters listed in Table 19 were used in Test 4.

Table 19: Test 4 Specific Simulation Parameters

Pause time	10 seconds
No. of malicious nodes	20
Minimum possible speed	0 m/s
Maximum possible speed	20 m/s

The results of Test 4 (shown in Fig. 18 and 19) indicate that the throughput of all three protocols improves with the increase in node speeds. This improvement can be attributed to the improved probability of detection of node behaviour due to the higher number of inter-node interactions. TORA-10 makes use of its inherent multi-path feature, which supports multiple link-disjoint paths to every known destination. In TORA, the established destination-oriented DAG can be severed due to a link failure. However, TORA doesn't react to such scenarios until the time it has at least one outgoing link pointing to the same destination. Hence, each data

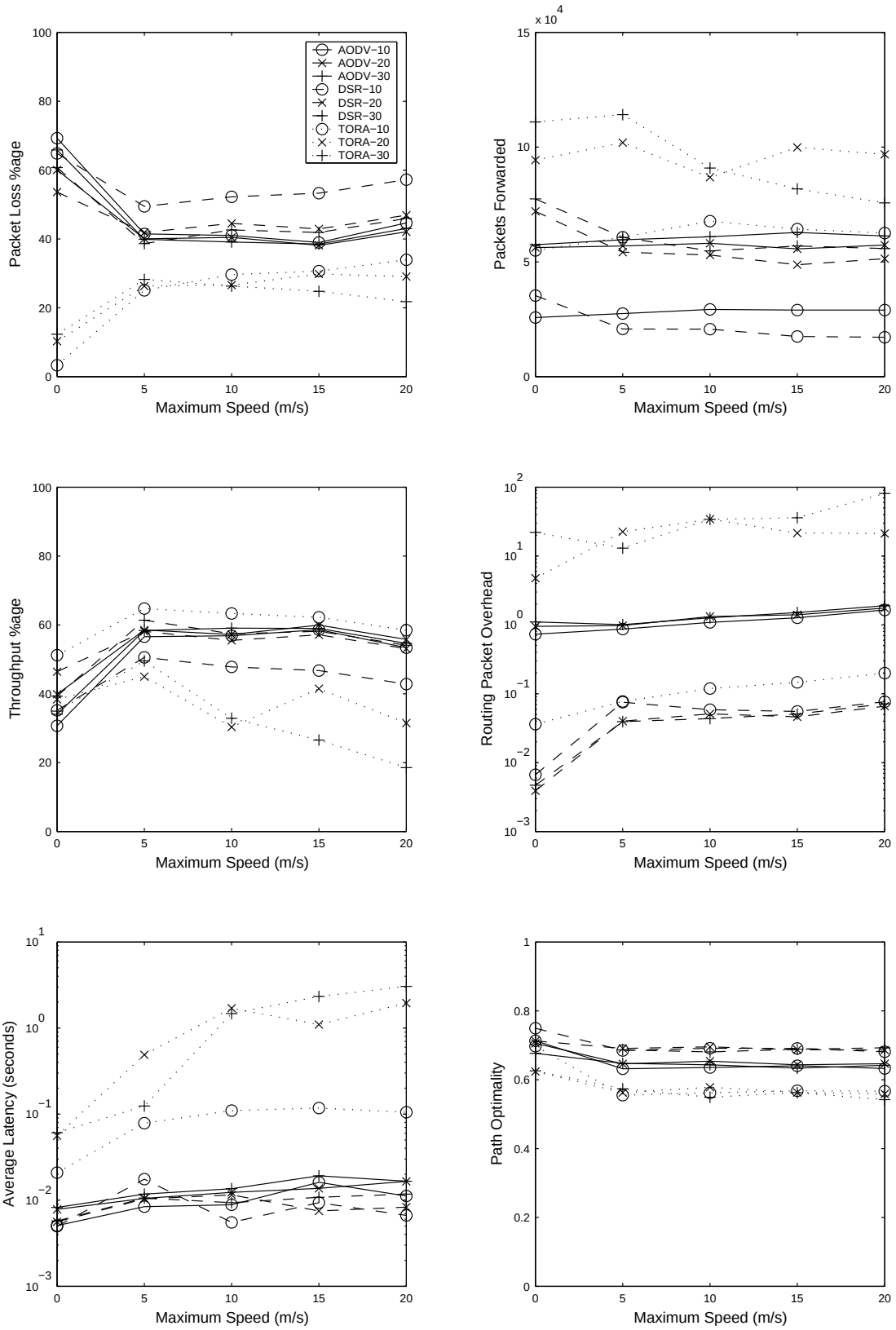


Figure 18: Test 4 : Performance Results

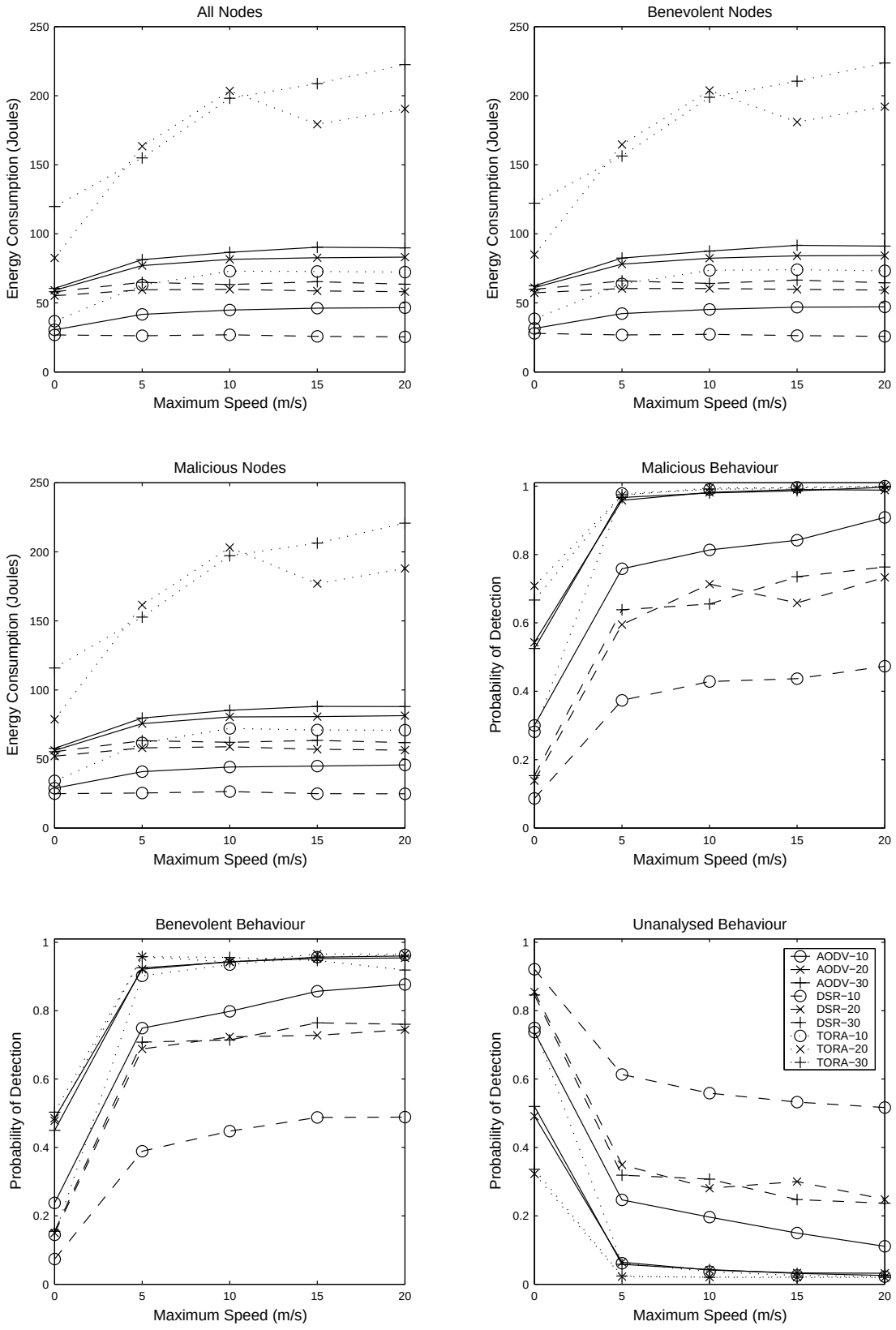


Figure 19: Test 4 : Energy and Detection Results

packet being forwarded by the intermediary nodes, is supported by this multi-path feature, which elevates the probability of successful delivery to a trusted node under varying network speeds. In contrast, nodes executing AODV only maintain limited routes to a destination, and are so unable to aid in packet delivery in the event of unavailability of a trusted next hop leading to a destination. DSR also has no option for an enroute correction of ongoing data connections, however single data packets are salvaged using trust filters. The link connectivity information in the DSR LINK CACHE is used by source nodes to form paths to previously known nodes, which helps in limiting the need for additional route discoveries.

The routing packet overhead for AODV and TORA increases with the network speed. At higher speeds the links are frequently disconnected, and so the nodes initiate additional route discoveries to sustain ongoing data connections. However, the routing overhead of TORA-10 remains comparatively lower than that of AODV due to its multi-path feature. On the other hand, DSR makes use of its caching strategy to lower the routing packet overhead with the increase in node speeds.

The path optimality of AODV and TORA protocols degrades with the increase in speed. This is observed due to the fact that routing decisions are made by intermediate nodes based upon trust levels, and so the actual path may deviate notably from the best available path. These longer routes also increase the latency of the network with the increase in node speeds. In contrast, source nodes executing the DSR protocol make the routing decisions before the start of a data connection, and so the data packets reach the destination with minimal deviation from the originally fabricated optimal path. This ensures that the latency of the network is not substantially raised with the increase in the network speed.

The energy consumed by the benevolent nodes remains slightly higher than the malicious nodes, which have been able to conserve upon the same by dropping packets during packet forwarding. The TORA-20 and TORA-30 protocols conserve significantly higher energy than the other protocols mainly due to the erroneously created positive feedback loops at high traffic loads. AODV protocol depicts higher energy

consumption than the DSR protocol due to its large routing overhead. The DSR protocol uniformly consumes energy with the increase in node speeds. This regular pattern is caused by the gradual decrease in the throughput which is supplemented by the routing packets being generated due to the increased network speed.

The probability of detection of all three protocols improves with the increase in node speeds. This is contributable to the increased number of interactions between unfamiliar nodes leading to better trust evaluation. At high speeds, AODV and TORA depict better detection probability than DSR primarily due to their dissimilar modus operandi. Intermediate nodes executing AODV or TORA, redirect the traffic in real time as per the contemporary trust levels of adjacent nodes. However, DSR supports a limited intermediary redirection facility by using trust filters and, hence, the number of nodes whose behaviour can be evaluated is limited to the actual number of nodes present on a particular data connection.

7.3.5 Test 5 : Simulation with varying Trust Update Intervals

In Test 5, we have used the parameters as listed in Table 20.

Table 20: Test 5 Specific Simulation Parameters

Maximum speed	20 m/s
Pause time	10 seconds
No. of malicious nodes	20
Minimum TUI	1 millisecond
Maximum TUI	20 seconds

The results of Test 5 (shown in Fig. 20 and Fig. 21), indicate that the average packet loss slightly increases with the increase in Trust Update Interval (TUI). It has also been observed that minimal packet loss, minimal packet overhead, and maximum throughput is achieved when the TUI approaches zero. At lower TUI values, congestion and efficiency of nodes is also assessed along with the detection of malicious and legitimate packet drop. Thus we can see an improvement in throughput at a

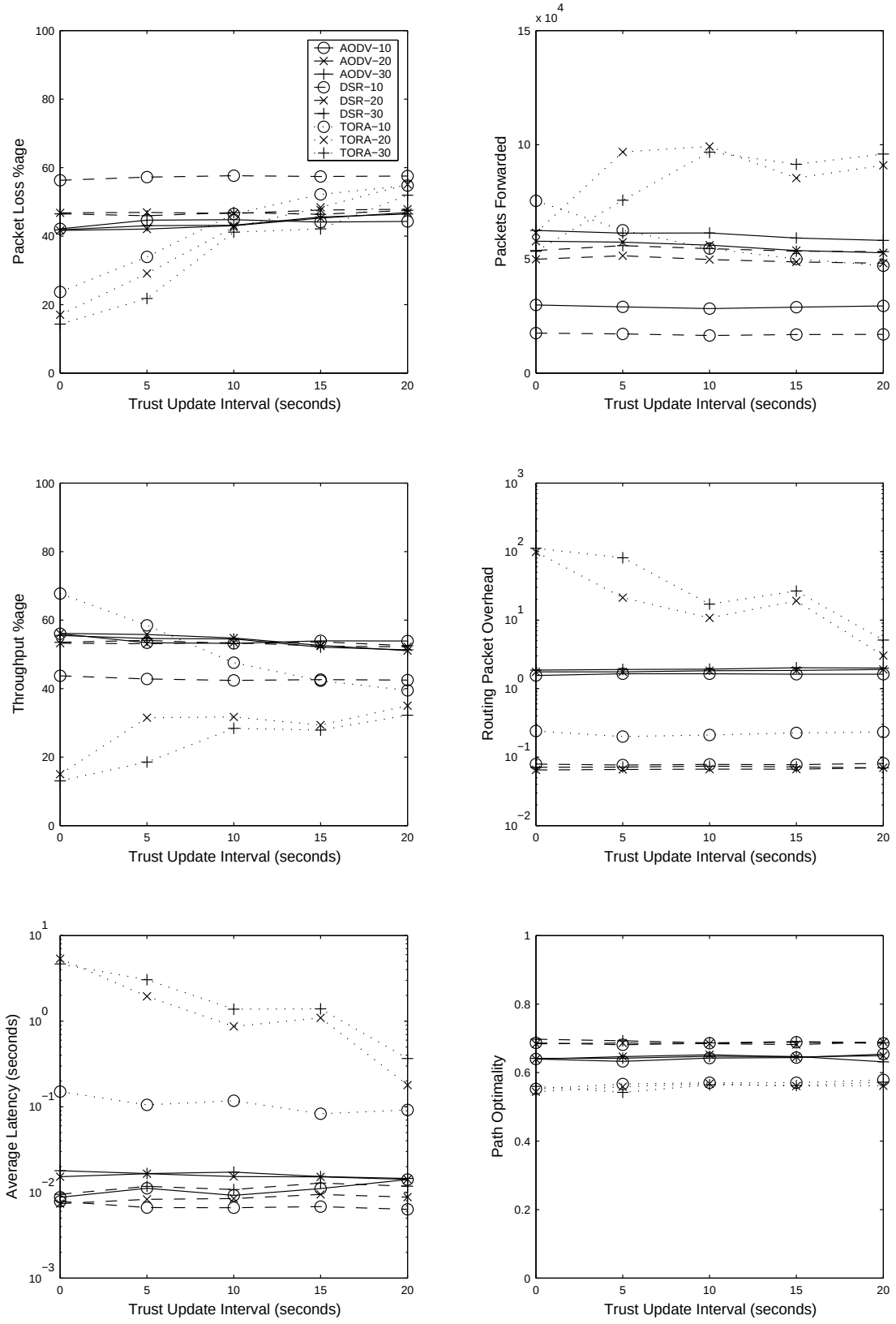


Figure 20: Test 5 : Performance Results

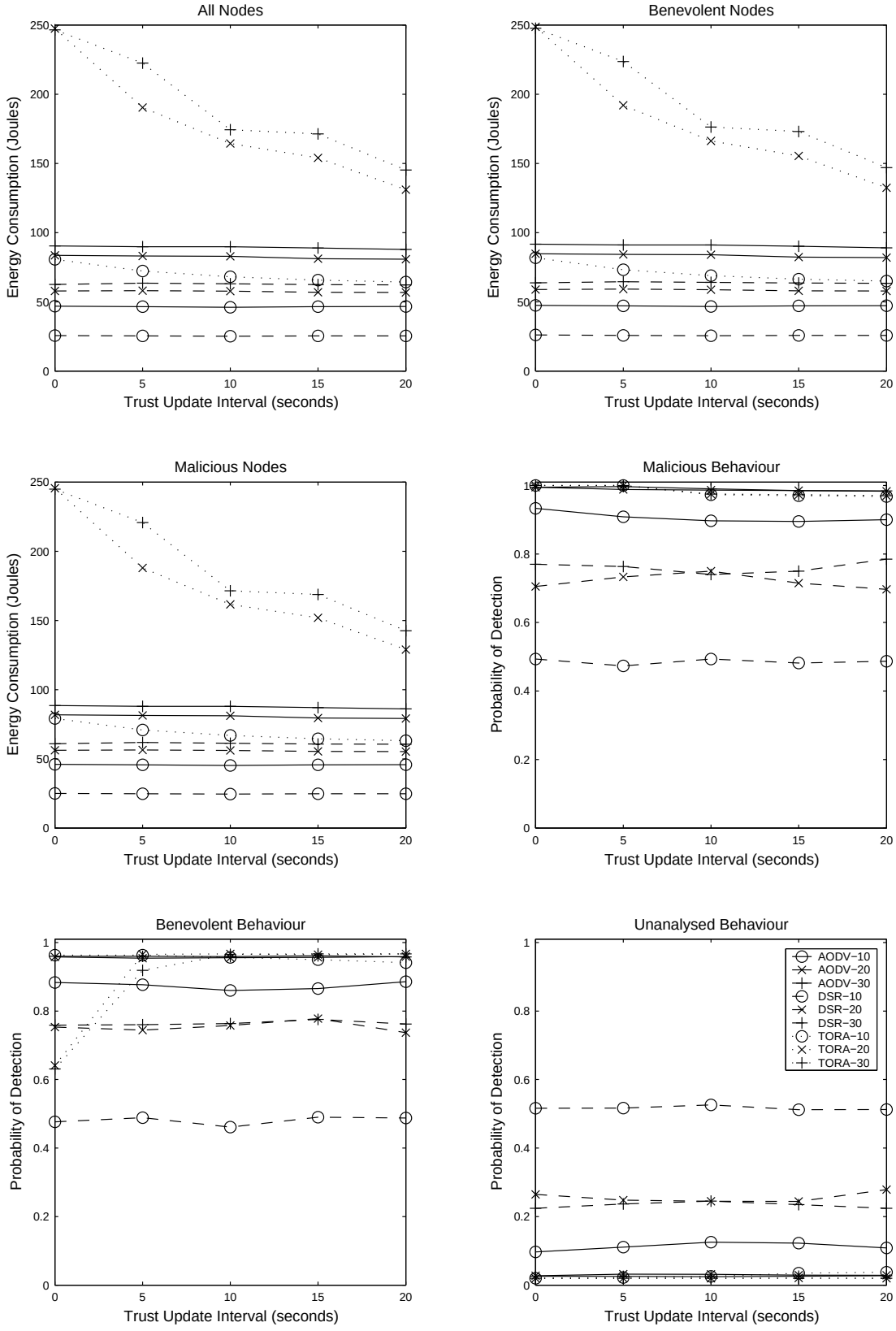


Figure 21: Test 5 : Energy and Detection Results

reduced TUI. However, at lower TUI a sluggish or committed benevolent node may be wrongly categorised as malicious due to its inability to forward packets during the TUI (observable in the probability of detection of TORA-20 and TORA-30). The average latency reduces as the path optimality improves with the increase in TUI. This is due to the fact that with the increase in TUI, the probability of detection of malicious behaviour also reduces. This occurrence causes malicious nodes to be involved in the routing of the data. Thus the packets, which make it to their final destinations, have traversed shorter paths and as a result have lower latency. We see an increase in the packet loss with increased TUI values, which occurs due to delayed trust computation leading to selection of routes containing malicious nodes. The increased promiscuous mode operation at a larger TUI does increase the energy consumption but the inferior throughput of the network reduces the overall consumption by lowering the reception and transmission load on all network nodes.

7.3.6 Test 6 : Comparison between Trusted AODV and ARAN Protocol

In Test 6, we compare the performance of the trusted AODV protocol (T-AODV) with that of the standard AODV and ARAN protocols. In this test we have used the simulation parameters as listed in Table 21. During the test, 15 malicious nodes deceptively decrement the hop count to zero for each forwarded `ROUTE REQUEST` and `ROUTE REPLY` packet. Thus these nodes pretend to be located at a distance of one hop from the source or destination node. During the test we measure the following two metrics:

Average Path Length: The average path length (in hops) of the received data packets.

Permeated Data Packets: The fraction of data packets received that passed through the malicious nodes.

Table 21: Test 6 Specific Simulation Parameters

Examined protocols	AODV, ARAN & T-AODV
Simulation time	250 seconds
Simulation area	1000 x 1000 m
Number of nodes	50
Minimum possible speed	0 m/s
Maximum possible speed	10 m/s
Pause time	30 seconds
Traffic type	CBR (UDP)
Maximum connections	5
Payload size	512 bytes
Packet rate	4 pkt/sec
Maximum malicious nodes	15
Attack	Deceptive decrementing of Hop Count

The results (shown in Fig. 22) indicate that the average path length of the received data packets using the T-AODV protocol is higher than that of AODV and ARAN. This is due to the fact that AODV inherently seeks shortest paths and, in doing so, selects paths which are being portrayed as short by malicious nodes, for routing the data traffic. ARAN also selects the first route discovery packet to reach a node in the shortest time. Both AODV and ARAN, having no facility for sustaining multiple paths, are compelled to route data traffic through the available paths even if they contain malicious nodes. On the other hand, each node executing T-AODV has multiple paths available with it, and so selects them based upon respective trust levels in order to circumvent malicious nodes. However, in doing so, the paths deviate significantly from the shortest possible paths and hence we observe elongated path lengths.

The multiple paths available with T-AODV along with the next hop trust information, enables it to select the optimal paths in terms of trust level and minimal number of hops. ARAN and AODV, both have no facility to re-route data either at the originating or at the forwarding stage. This inadvertently permits a lot of data connections to permeate through the malicious nodes. Nodes executing T-AODV are, however, able to re-route traffic at both the originating and forwarding stage

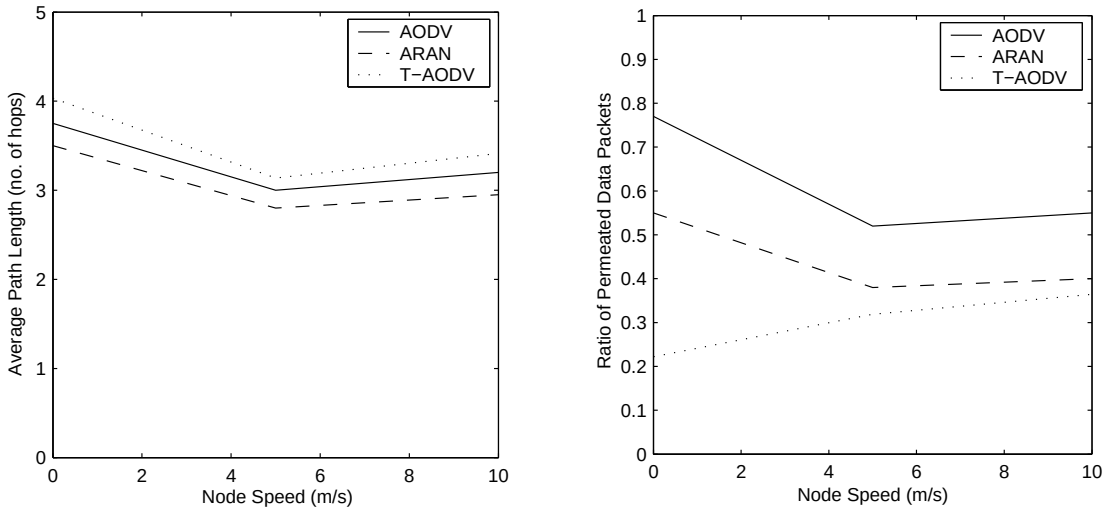


Figure 22: Test 6 : Performance Results

and hence only a small number of data packets pass through the malicious nodes.

7.3.7 Test 7 : Comparison between Trusted and Pathrated DSR Protocol

The parameters listed in Table 22 were used in Test 7.

Table 22: Test 7 Specific Simulation Parameters

Examined protocols	Trusted & Pathrated DSR
Number of connections	10
Maximum speed	20 m/s
Pause time	10 seconds
Minimum malicious nodes	0
Maximum malicious nodes	20

In this test we compare the performance of the trusted DSR protocol with that of the Watchdog/Pathrated DSR [58]. As the watchdog can only detect malicious packet drop using the DSR protocol thus we only evaluate DSR in Test 7 and restrict it to black and grey hole attacks. The Pathrated DSR initially assigns a low trust value to each encountered malicious node. This although advantageous in avoiding malicious nodes, causes certain problems when dealing with grey holes,

as discussed in Section 6.5. The Pathrater scheme also assigns an extremely low trust value (-100) to a detected malicious node, which cuts off the malicious node from the network. This essentially segregates nodes into two possible states i.e. either benevolent or malevolent. With our proposed trust model, there is no cut-off state, and thus trust values transpire in a constant range. Thus all nodes, either benevolent or malevolent, are involved in the routing process based upon their contemporary trust levels. This in effect facilitates engagement of nodes portraying as grey holes or partial forwarders into the routing process, in the event that alternate trusted nodes are not available. This also prevents benevolent nodes, undergoing legitimate packet loss, from being graded as malicious and consequently being isolated from the network. These nodes are simply bypassed in the routing process until the time other trusted nodes are available in the network.

The results (shown in Fig. 23), highlight the effectiveness of the trusted DSR in comparison to the Pathrated DSR. Both protocols are successfully able to detect grey holes. However, the Pathrated DSR immediately absolves such nodes from the routing process and, in the worst case, when no alternate route is available, the packet is dropped at the source. This essentially increases the packet loss and lowers the throughput of the network. On the other hand the trusted DSR endeavours to sustain a best-effort delivery and, if no alternate routes are available, selects the grey holes for the routing process. Thus we see an improvement of up to 8% in the throughput of the network. Due to the improved trust computation and its subsequent application, the trusted DSR makes effective routing decisions, by circumventing malicious nodes, thereby lengthening optimal possible paths and increasing the latency of the network. The routing overhead of the trusted DSR, however, remains consistent with that of the Pathrated DSR due to no extraneous route discoveries.

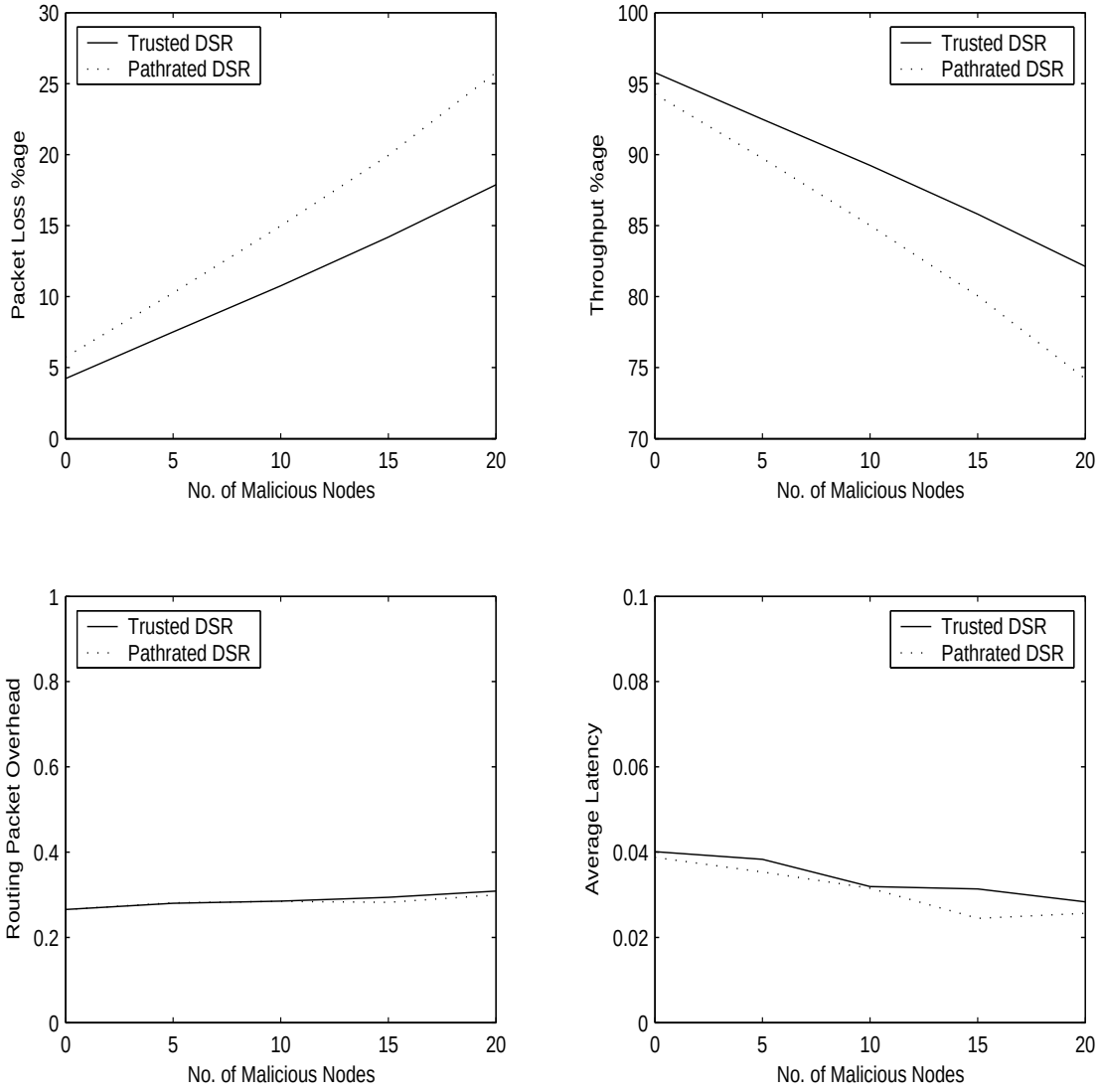


Figure 23: Test 7 : Performance Results

7.3.8 Test 8 : Reputation Exchange with Varying Number of Malicious Nodes

In Test 8, we compare the performance of the trusted routing protocols, which are being assisted by an integrated reputation exchange mechanism, in the presence of malicious nodes. The AODV protocol, making use of only direct trust, is represented as AODV-DIRECT; AODV integrated with reputation exchange through `ROUTE REQUEST` packets is represented by AODV-RREQ and AODV integrated with reputation exchange through `ROUTE REPLY` packets is represented as AODV-RREP. A similar convention is used for the DSR and TORA routing protocols. In this test we have used the simulation parameters as listed in Table 23.

Table 23: Test 8 Specific Simulation Parameters

Number of Connections	10
Maximum speed	20 m/s
Pause time	10 seconds
Minimum malicious nodes	0
Maximum malicious nodes	20

The results (shown in Fig. 24 and 25), indicate that all three protocols have been able to minimise the packet loss using the integrated reputation exchange mechanisms. However, the overall performance gain in each protocol is dependent upon its basic working principle.

AODV-RREQ shows improved performance over AODV-DIRECT, where the reputations are flooded in the network with the help of `ROUTE REQUEST` packets. As AODV relies exclusively on recent routing information, it makes extensive use of route discoveries to find new routes. This helps to disseminate trust information in the network at a higher rate, improving upon the probability of detection and permitting better route switching by all nodes. This in turn improves upon the packet forwarding rate and thereby increases the throughput of the network. However, as `ROUTE REPLY` packets are unicast, AODV-RREP does not show significant improvement over AODV-DIRECT. The packet overhead, latency, path optimality

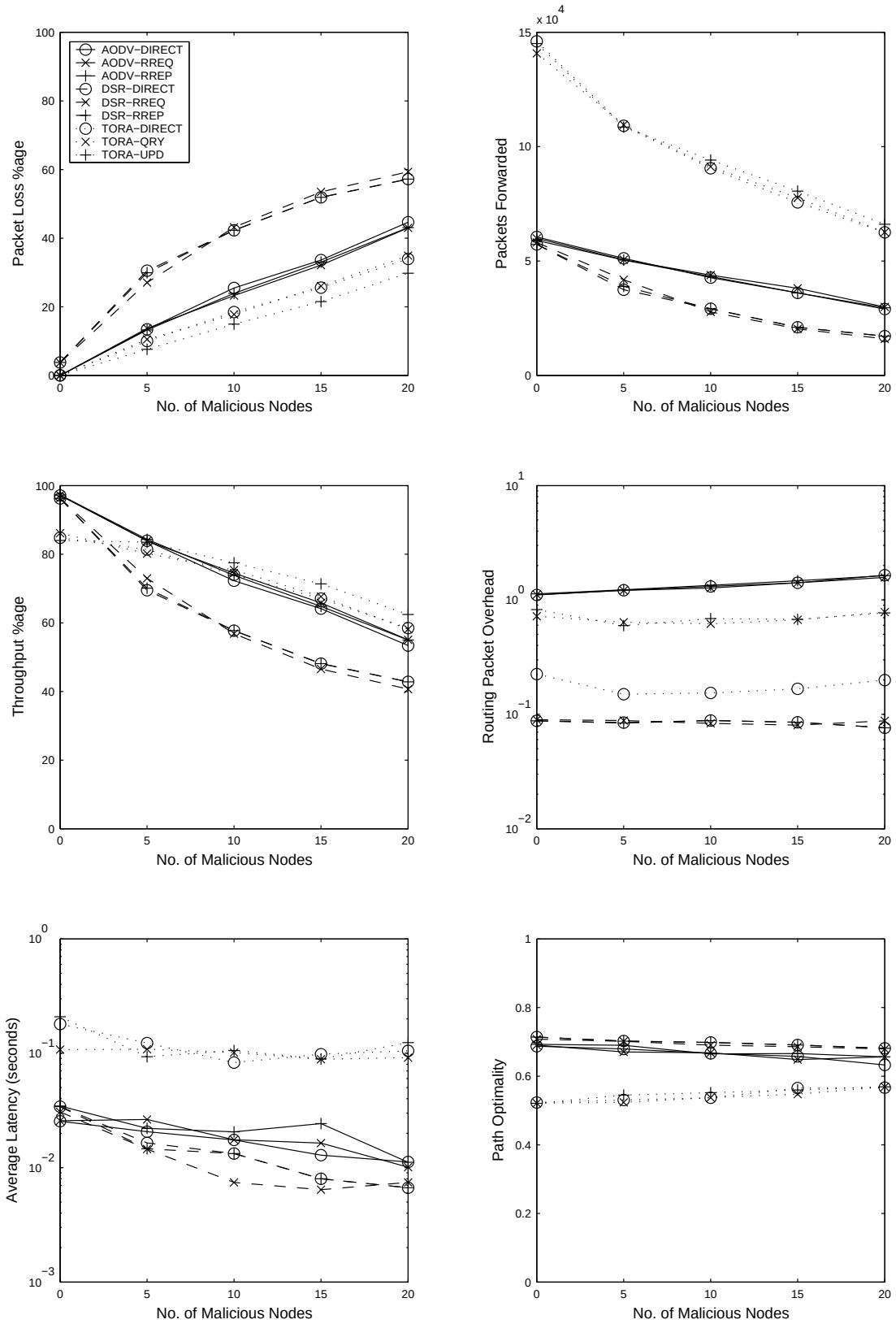


Figure 24: Test 8 : Performance Results

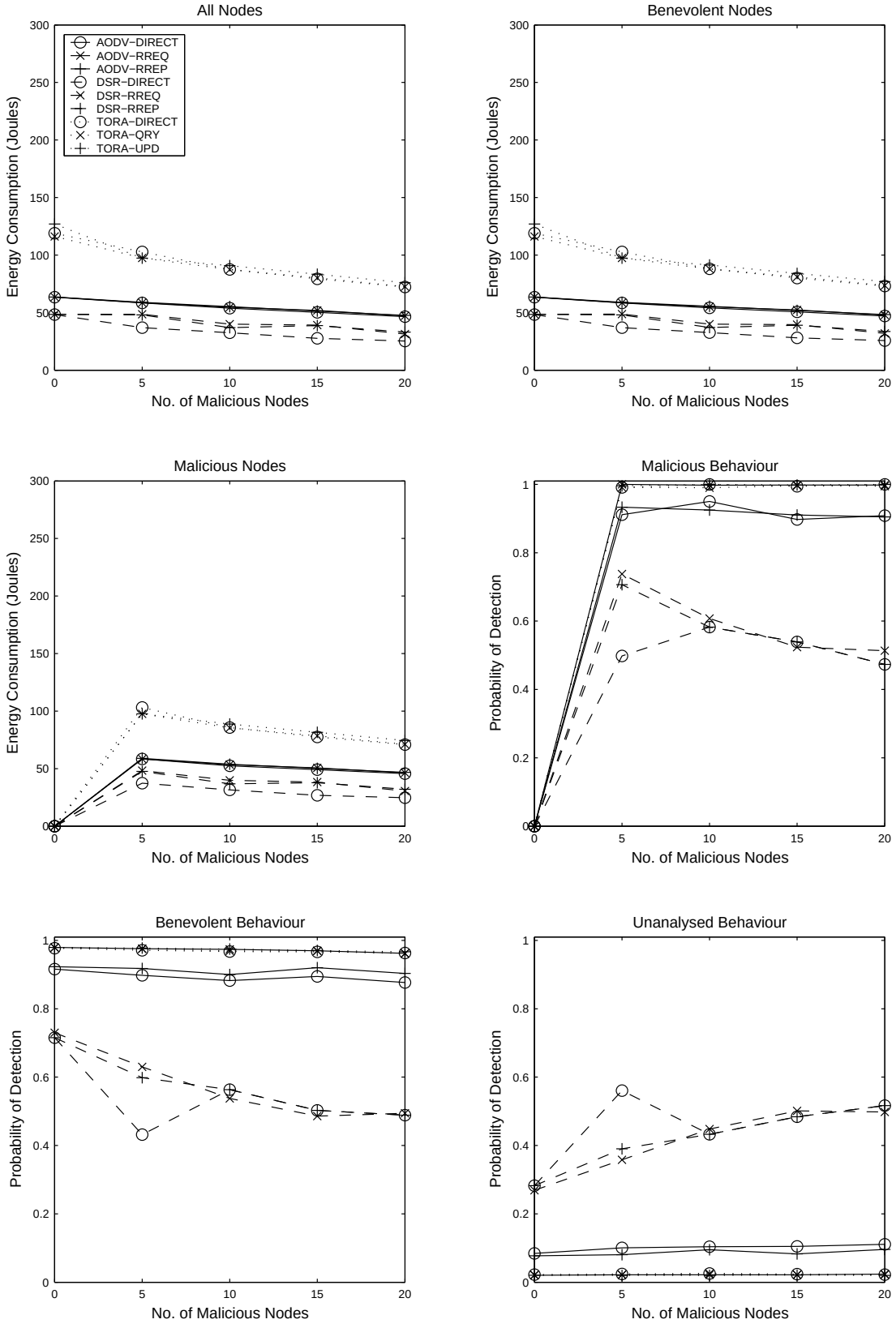


Figure 25: Test 8 : Energy and Detection Results

and energy consumption of AODV-RREQ and AODV-RREP show minimal deviation from that of AODV-DIRECT.

DSR-RREQ shows degraded performance over DSR-DIRECT due to DSR's reliance on cached information rather than fresh route discoveries. This essentially lowers the number of effective route discoveries resulting in poor dispersal of trust information. Thus these reputations may frequently contain incomplete trust information regarding nodes. Accordingly any sending node, which had received such a reputation, may not use a poorly recommended node in any data connection for the network interval. This in turn lowers the packet forwarding rate and consequently reduces the throughput of the network. However, trust dispersal using the `ROUTE REPLY` packets in DSR-RREP improves upon the forwarding rate of the network. These reputations, which are recent as they are acquired just prior to establishing a data connection, contain accurate information and hence permit better route selection at the source node. The packet overhead, latency, path optimality, energy consumption and detection probabilities of DSR-RREQ and DSR-RREP show minimal deviation from that of DSR-DIRECT.

TORA endeavours to minimise route discoveries and in doing so prefers additional link reversals over new route discoveries. These reversals, inject additional `UPD` packets in the network, so as to limit the number of generated `QRY` packets. These supplementary `UPD` packets facilitate in dispersing trust information and thus we observe a lower packet loss with the TORA-UPD protocol. This also helps to establish a better packet forwarding rate and throughput over TORA-QRY and TORA-DIRECT protocols. However, the control packet overhead, of the TORA-QRY and TORA-UPD protocol, surpasses that of the TORA-DIRECT protocol. This is primarily due to the increased link reversals, which are initiated in response to the extended trust information now available with the nodes. However, minimal deviation is observed among the TORA-DIRECT, TORA-RREQ and TORA-RREP for latency, path optimality, energy consumption and detection probabilities.

7.3.9 Test 9 : Reputation Exchange with Varying Node Maximum Speeds

The parameters listed in Table 24 were used in Test 9.

Table 24: Test 9 Specific Simulation Parameters

Number of Connections	10
Pause time	10 seconds
No. of malicious nodes	20
Minimum possible speed	0 m/s
Maximum possible speed	20 m/s

The results (shown in Fig. 26 and 27), confirm that the AODV-RREQ, DSR-RREP and TORA-UPD protocols are able to effectively disperse and exploit the trust information under dissimilar network speeds. This in turn improves upon the probability of detection of malicious and benevolent behaviour. As a result, these protocols have been able to evade malicious nodes, thus lowering the packet loss and improving upon the packet forwarding rate and throughput of the network. AODV-RREQ and DSR-RREP are generally able to reroute traffic on other available paths without generating any extra routing packets. On the other hand, the TORA-UPD and TORA-QRY protocols, upon receipt of trust information, have to undergo significant link reversals in order to maintain a trusted route in the network. Thus we observe a higher control packet overhead for these protocols. However, the latency, path optimality and energy consumption of all reputation assisted protocols remains consistent with that of their direct trust variants.

Although, the overall improvement achieved by the reputation assisted protocols is marginal, we have been able to observe the effect of certain routing packets on the working of the underlying protocols. AODV, due to its higher control packet overhead, is able to effectively spread trust information in the network by flooding `ROUTE REQUEST` packets. On the other hand, DSR tries to minimise new route discoveries and essentially limits the generation of the `ROUTE REQUEST` packets. However, the ensuing `ROUTE REPLY` packets received by a node, initiating a data connection, aid

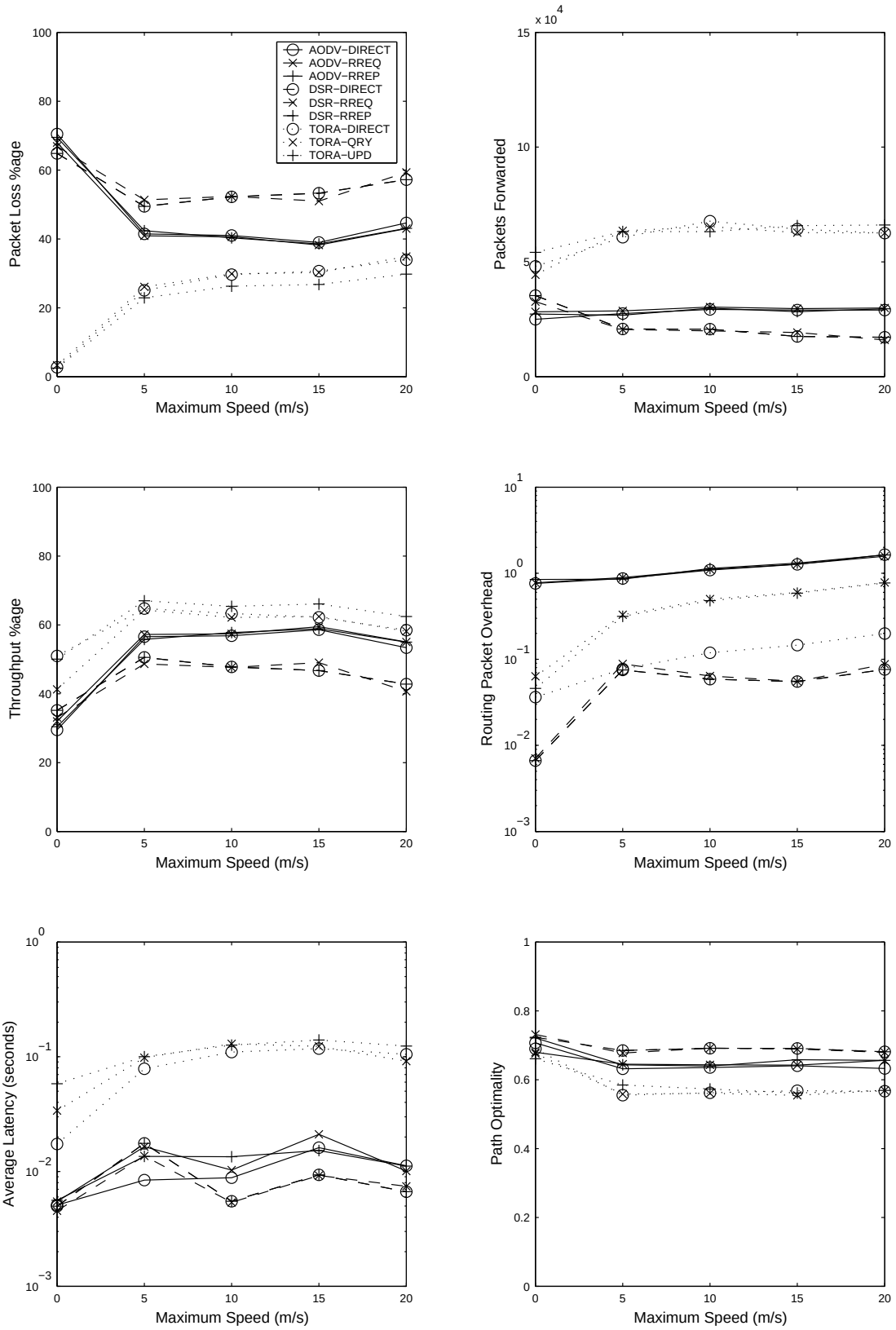


Figure 26: Test 9 : Performance Results

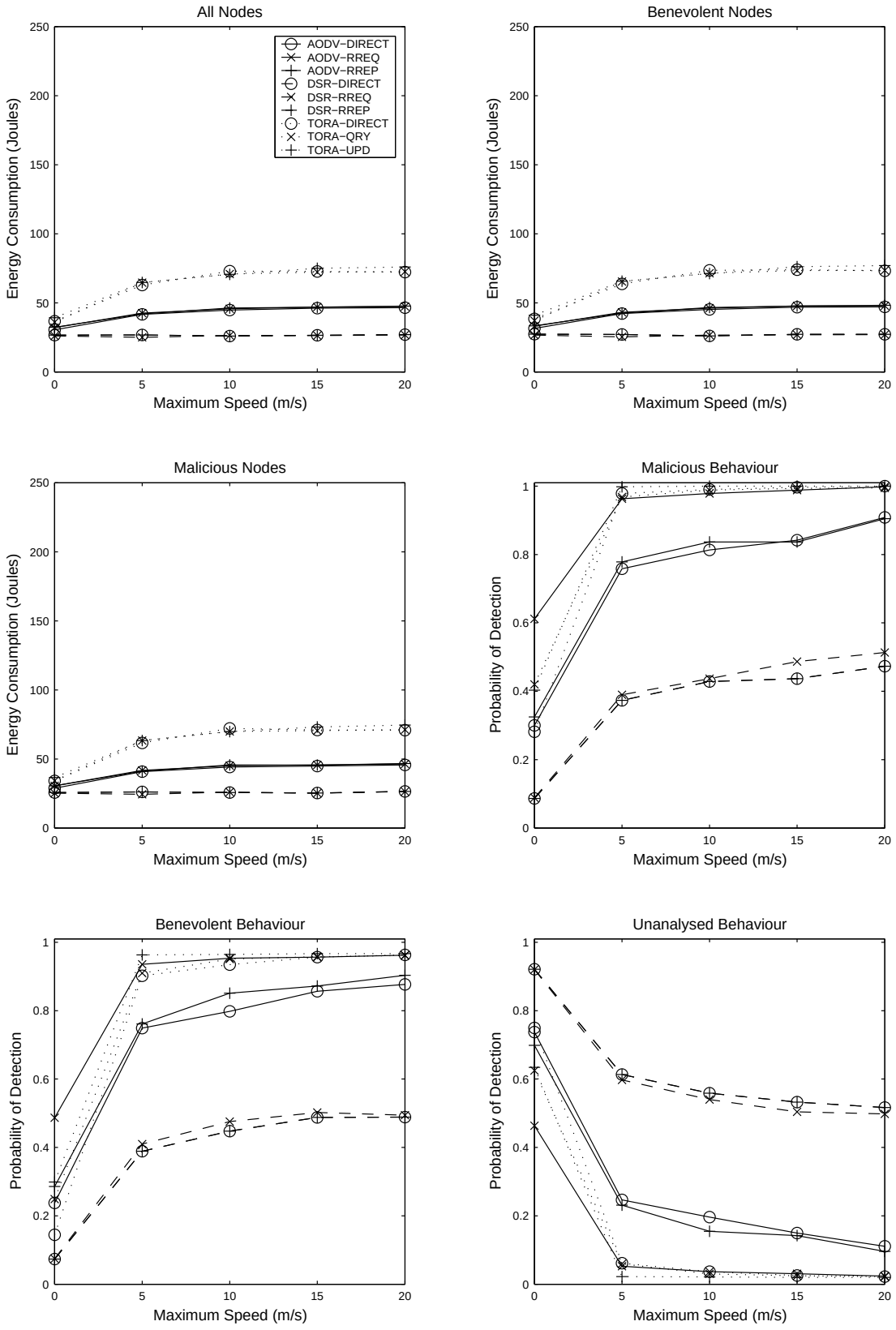


Figure 27: Test 9 : Energy and Detection Results

it in selecting an appropriate trustworthy path through the network. Similarly, in TORA, where the emphasis is on minimal new route discoveries, trust dispersal through `UPD` packets helps to improve upon the network performance in the presence of malicious nodes.

7.4 Analysis

The simulation results indicate that the trusted TORA protocol performs exceptionally well in the presence of malicious nodes under low traffic and diverse mobility conditions. However, due to the inherent feedback loop problem of the TORA protocol, it fails to sustain the same performance at higher traffic loads. The multi-path feature of the TORA protocol permits intermediary nodes to make localised routing decisions based upon trust levels. These decisions help to improve the overall throughput of the network by avoiding malicious nodes. TORA has a relatively higher packet overhead due to its IMEP reliable delivery mechanism and thus induces higher energy consumption for both benevolent and malicious nodes. The path optimality of TORA remains significantly constant even with an increased number of malicious nodes due to its localised route fixation mechanism. However, the path optimality is lower than the other two protocols as TORA endeavours to fix routes locally rather than finding the optimal path to a destination. This feature although lowers the packet overhead, increases the latency due to sub-optimal paths. The behaviour detection probability of TORA improves with the increase in node speeds at lower pause times.

Nodes, executing the trusted DSR protocol, consume the least energy due to the minimal routing overhead generated by the protocol. The DSR protocol makes maximum use of its caching scheme to limit the number of search requests for known nodes. This scheme helps to minimise the latency of the network under varying speeds and node pause times. However, the overall throughput of the DSR protocol in the presence of malicious nodes, remains lower than that of the TORA

and AODV protocols under varying mobility conditions. This is attributed to the fact that a malicious node, present on a link in the LINK CACHE scheme of the DSR protocol, will be used until the time its trust is evaluated by the source node. Thus a source node will continue sending packets on the path containing a malicious node until the time the path between the source and destination is severed. Only upon the breakage of the path, an alternate route can be found to the same destination either from the cache or through a new route discovery. Whereas in TORA and AODV, the point to point routing process inadvertently helps to avoid malicious nodes in the data connections. The behaviour detection probability of DSR remains the least of the three protocols primarily due to its source driven working principle.

AODV exhibits consistent throughput under different traffic loads with varying number of malicious nodes. The throughput improves with the decrease in node pause times and with the increase in mobility. The routing packet overhead of AODV increases with the node speeds. AODV and TORA, functionally operate in a similar manner where the intermediary nodes pass the data packets to the next trusted hop on the path to the destination. However, AODV does not suffer from routing loops at higher traffic loads as does TORA, and so is able to maintain a stable throughput. In lower traffic loads, AODV engages its unabridged route fixation to the destination rather than localised route recovery and thus depicts improved latency and path optimality over TORA. However, in doing so, it generates more overhead as compared to that of TORA. The behaviour detection probability of AODV improves with the increase in node speeds at lower pause times.

A source routing protocol like DSR can benefit the most from reputations, where each sending node has the option to select trustworthy nodes at the start of the connection. However, in the event that the sending node has incomplete trust information about all the nodes in the path, malicious nodes may still be included in the routing process. Reputation can thus help in such situations by providing trust information beyond a single hop. In contrast, in hop-to-hop routing protocols like TORA and AODV, the sending and forwarding nodes can only make the next-hop

routing decision, for which direct trust is considered more accurate than received reputations. Cryptographic message digests are frequently employed to ensure the integrity of reputation messages. However, such mechanisms depend upon certain certifying authorities, which are considered trustworthy both by the message sender and the recipient. However, the exchange of reputations in an extemporised environment, without a trusted third party, is vulnerable to tampering and deception. It also increases the routing overhead of the network and requires extended state information to be retained at each node.

A smaller TUI value helps to evaluate trust accurately in networks with high mobility, i.e. high rate of link creation and breakages. The results indicate that a network having a maximum speed of 20 m/s has an optimal TUI value close to 5 seconds. When the TUI is kept extremely small, the possibility exists that a number of committed benevolent nodes may be incorrectly graded as being inefficient. Similarly, a large TUI value may lead to false detection, when nodes go out of range of each other within the TUI. Packet overhead generally remains constant with different TUI values ensuring even energy consumption. The probability of detection of malicious or benevolent behaviour remains higher at lower TUI values and gradually decreases with the increase in TUI.

A subsidiary advantage of such a trust-based routing scheme is that any node not capable of sustaining the required traffic flow is automatically downgraded when it drops packets, and some other node having a higher trust level is selected for the routing process. This feature helps to reduce traffic congestion on trustworthy nodes by transferring the traffic load to other available nodes in the neighbourhood [111, 52]. Actually, the trust model does not take into account the amount of traffic generated by a node; it only endeavours to sustain a best-effort delivery for the generated traffic.

The current trust model detects malicious and selfish nodes and does not include them in the routing process. Such scenarios may help a selfish node to save upon its battery power. However, as the trust level of such nodes is known to other nodes in

the network, a punishing scheme can be implemented in which such nodes are not provided the requisite network services. Such a mechanism may help to invigorate cooperation rather than selfishness in the participating nodes [101].

The method of verifying the sincerity of the immediate neighbour through promiscuous mode is an effective way of detecting selfish and malicious nodes. However, this technique has certain drawbacks, which have been highlighted by Marti *et al.* [58]. One of them is the ambiguous collision problem in which node a cannot hear the broadcast from neighbouring node b to node c, due to a local collision at node a. The other is the receiver collision problem, in which node a overhears node b broadcast a packet to node c but cannot perceive the collision which takes place at node c. In the same way, if the nodes have different transmission power ranges or directional antennae, the mechanism of passive acknowledgments might not function as desired [1]. However, the impact of dissimilar transmission ranges and hidden-nodes can be minimised by using MAC layer acknowledgements, as provided by the IEEE standard 802.11 Distributed Coordination Function, along with passive acknowledgements.

For connection-oriented services or asynchronous transceivers, a similar approach can be followed where the REVERSE ROUTE may be the same or different from the FORWARD ROUTE. As the trust levels are computed independently, T_{xy} may not be always equal to T_{yx}. This ensures that all intermediate nodes make independent decisions regarding the routing of the packets, resulting in dissimilar FORWARD ROUTEs and REVERSE ROUTEs. However, as the route discovery procedures are confined to situations when no path is available, such a drawback is not considered significantly harmful to the performance of connection-oriented services [31, 67].

Although maintenance of consistent MAC to IP binding offers reasonable protection again spoofing attacks, it may be argued that a node may spoof both its MAC and IP addresses. Such an alternation is not directly perceivable to any of its adjacent nodes, which may consider the spoofed node to be just another network node. However, such a scenario is likely to originate IP address collisions, where

interactions with the spoofed and legitimate node may break the MAC and IP bindings. Such, spoofing attacks can be avoided either by engaging an IP address duplication avoidance scheme [106] or through some other suitable identity theft protection scheme [18].

In order to deceive the trust mechanism, any malicious node may initially perform in a benevolent manner, so as to gain a higher trust rating in the network. However, as soon as the malicious node starts to modify or drop packets, its corresponding direct trust levels maintained by its immediate neighbours start to decay. Consequently, the malicious node is bypassed in all subsequent route discoveries. However, the isolation of malicious nodes, which exhibit complex behaviour like launching attacks in collusion or by operating at the verge of benevolent and malevolent behaviour, is quite intricate. We recommend using Intrusion Detection systems such as those proposed by Zhang *et al.* (2000) and Kachirski *et al.* (2002) for isolating such nodes in ad-hoc networks.

7.5 Summary

In this chapter we have carried out simulations for three trust-based routing protocols in the presence of malicious and selfish nodes in the network. In order to achieve an accurate measure of the protocols' performance, we have undertaken a number of tests involving disparate mobility and attack patterns. We have also compared the performance of the trusted routing protocols with that of some existing protocols. The results of the simulations indicate that the performance of the trusted routing protocols is superior to that of the native protocols in an antagonistic environment. Nodes executing the trust model are able to detect malevolent nodes with a high probability of success. These nodes are thus able to circumvent malicious and selfish nodes by adopting alternate routes in the network. This improves the overall throughput of the network by up to 20% despite the presence of 40% malicious and selfish nodes in the network. In the next chapter we offer concluding remarks and identify areas for possible future work.

Chapter 8

Conclusions and Future Work

This chapter provides the overall conclusions of the thesis and potential areas for future research.

8.1 Conclusions

Ad-hoc networks are formed with the help of a large number of wireless nodes, generally with limited energy, computation and transmission powers. The prime advantage of such networks is their capability to operate without any fixed or virtual infrastructure. Each node helps every other node in the network by forwarding their packets in return for a similar favour from them. All is well if such an altruistic attitude is upheld by all participating nodes. However, as these nodes often operate in a physically insecure environment, they are vulnerable to capture and compromise. In addition, the communication medium being wireless, restricts enforcement of rigourous node memberships, and so a number of malicious nodes may also participate in the network. These nodes, in order to snoop or sabotage, can carry out a variety of attacks against the network. To counter such nodes in an ad-hoc network, cryptographic or trust-based schemes are generally used. Cryptographic schemes, although considerably secure, impose a number of prerequisites

upon the network establishment and operation phase, including the requirement of a trusted third party and pre-configuration of nodes. This in turn impedes their application to pure ad-hoc networks, which are established in a spontaneous and impromptu manner.

In order to maintain the makeshift nature of ad-hoc networks, in this thesis, we have developed a novel approach, which enforces trust in networks executing standard routing protocols. We have moved from the common mechanism of achieving trust via security to enforcing dependability through collaboration. Each node in the network monitors the behaviour of its surrounding nodes with the help of an effort-return trust model. The model derives and computes the direct trust in adjacent nodes based upon their *modus operandi*, using a set of context specific categories. The number of such categories varies according to the type of routing protocol employed in the network. Two common categories, which have the maximum impact on trust development are Passive Acknowledgements (P_A) and the Packet Precision (P_P). The category P_A primarily looks at the node selfishness level by observing its packet forwarding capability. As a result, the category P_A, in addition to detecting selfish nodes, is also able to detect nodes that are experiencing congestion. The category P_P observes the integrity of the forwarded traffic and is hence able to detect modification attacks against the network traffic. The trust derived through these trust categories is then associated with the routing information available in the form of tables or cache. This permits each node to dynamically adapt its routing mechanism according to the observed sincerity of the network nodes rather than simply using the default shortest path routing. This enables nodes to retrieve dependable routes from the routing tables or cache during subsequent route discoveries.

We have examined the performance of three trust-based reactive routing protocols in a network consisting of a high percentage of malicious and selfish nodes. The results from the simulations indicate that these protocols outperform the standard protocols under varying attack, mobility and traffic conditions. However, the performance of each trust-based routing protocol varies significantly under similar

attack conditions due to its peculiar working mechanism. At lower traffic loads, the Temporally Ordered Routing Algorithm (TORA) protocol performs better than the other two protocols in the presence of malicious nodes. Ad-hoc On-Demand Distance Vector (AODV) routing protocol performs second best but surpasses TORA at higher traffic loads. The Dynamic Source Routing (DSR) protocol, although provides lower throughput as compared to either TORA or AODV, generates the least routing overhead, has the lowest latency and consumes minimal energy.

8.2 Future Work

Some of the potential areas for future research are described in this section.

8.2.1 Extension to Proactive Routing Protocols

Most of this research is directed towards reactive routing protocols. However an application to a proactive routing protocol would provide further insight into the trust model's performance. Proactive routing protocols generally share topology information on a periodic basis using link tables or distance vectors. The trust distribution can thus be integrated with the underlying routing protocol, where direct trust values may be passed along with the routing information. However, the precise impact of trust dispersal on the protocol's performance may vary depending upon its working methodology.

8.2.2 Integration of Trust Model with Cryptographic Mechanisms

The trust model has been developed keeping in mind the improvised nature of pure ad-hoc wireless networks. However, as elaborated earlier, the model improves the dependability of the network in the presence of malevolent nodes and is not to be

considered an alternate to security schemes. Any security mechanism working in conjunction with the trust model, would enforce dependability as well as provisioning of the requisite security services in the network. A desirable requirement of such an integration would be to minimise the transformation from a pure ad-hoc network to a managed ad-hoc network.

8.2.3 Alternate Trust Dispersal Mechanisms

Reputations help to establish trust in unknown nodes and reinforce trust in already known nodes. For effective dispersal of trust in pure ad-hoc networks, special measures are required to retain the integrity and freshness of reputations. Although a number of cryptographic solutions exist for accurate trust dispersal, their usage is generally restricted to managed networks. Alternate integrated or independent trust dispersal mechanisms may be studied, which permit effective and timely reputation exchange without being vulnerable to tampering.

8.2.4 Trust Decay over Time

Presently, we are not considering aging of trust information in our trust model, primarily due to the limited lifetime of ad-hoc networks. However, for long life ad-hoc networks, a trust decay policy may be developed. Such a policy should take into consideration the timeliness of the trust information during the computation of aggregate trust in another node. For accurate utilisation of this information, supplementary state information would be required to be retained at each node. Accordingly, a critical cost-benefit analysis may be required to evaluate the net performance gain in relation to the additional memory and processing requirements.

8.2.5 Application to Load Balancing

In addition to isolating malicious and selfish nodes, the trust model can successfully detect congested nodes in the network. This information can be used to supplement a load balancing scheme, which would support ad-hoc networks that have to function in a benevolent or malevolent environment. Congested nodes, which portray a behaviour profile similar to selfish nodes, can be bypassed with the help of the trust model with a high probability of success. This would ensure that the load is shifted from congested to less congested nodes in the network. This load transfer mechanism is likely to oscillate, in which case load is transferred onto relatively vacant nodes, until the time homogeneous load distribution is achieved in the network.

Bibliography

[1] I. Aad, J. Hubaux, and K. W. Knightly. Denial of Service Resilience in Ad hoc Networks, 2004.

[2] N. Asokan and P. Ginzboorg. Key Agreement in Ad-hoc Networks. *Computer Communication Review*, 23(17):1627–1637, 2000.

[3] R. Au, M. Looi, and P. Ashley. Automated Cross-Organisational Trust Establishment on Extranets. In *Proceedings of the Workshop on Information Technology for Virtual Enterprises*, pages 3–11. IEEE Press, 2001.

[4] A. Back. A Denial of Service Counter-Measure. *http://www.hashcash.org/*, 2002.

[5] D. Balfanz, D. Smetters, P. Stewart, and H. Wong. Talking to Strangers: Authentication in Ad hoc Wireless Networks. In *Proceedings of the 9th Annual Network and Distributed System Security Symposium (NDSS)*, 2002.

[6] T. Beth, M. Borcherding, and B. Klein. Valuation of Trust in Open Networks. In *Proceedings of the Third European Symposium on Research in Computer Security (ESORICS)*, pages 3–18. Springer-Verlag, 1994.

[7] C. Bettstetter. Topology properties of Ad hoc Networks with Random Waypoint Mobility. *Mobile Computing and Communications Review*, 7(3):50–52, 2003.

[8] J. Broch, D. A. Maltz, D. B. Johnson, Y. C. Hu, and J. Jetcheva. A Performance Comparison of Multi-hop Wireless Ad hoc Network Routing Protocols. In *Proceedings of the 4th Annual International Conference on Mobile Computing and Networking (MobiCom)*, pages 85–97. ACM Press, 1998.

[9] S. Buchegger and J.Y.L. Boudec. Performance Analysis of the CONFIDANT Protocol: Cooperation Of Nodes - Fairness In Distributed Ad-hoc NeTworks. In *Proceedings of the 3rd ACM International Symposium on Mobile Ad hoc Networking and Computing (MobiHoc)*, pages 226–236. ACM Press, 2002.

[10] L. Buttyan and J. Hubaux. Enforcing Service Availability in Mobile Ad-hoc WANs. In *Proceedings of the 1st ACM International Symposium on Mobile Ad hoc Networking and Computing (MobiHoc)*, pages 87–96. IEEE Press, 2000.

[11] L. Buttyan and J. Hubaux. Stimulating Cooperation in Self-Organizing Mobile Ad hoc Networks. In *Proceedings of the ACM/Kluwer Mobile Networks and Applications (MONET)*, volume 8, pages 579–592. Kluwer Academic Publishers, 2003.

[12] D.W. Carman, P.S. Kruus, and B.J. Matt. Constraints and Approaches for Distributed Sensor Network Security. Technical Report 00-010, NAI Labs, 2000.

[13] S. Carter and A. Yasinsac. Secure Position Aided Ad hoc Routing Protocol. In *Proceedings of the IASTED Conference on Communications and Computer Networks (CCN)*, pages 329–334. ACTA Press, 2002.

[14] CMU. Wireless and Mobility Extensions to NS. *The Monarch Project*, 1998.

[15] M. S. Corson and A. Ephremides. Lightweight Mobile Routing Protocol (LMR), A Distributed Routing Algorithm for Mobile Wireless Networks. *Wireless Networks*, 1995.

[16] S. Corson and J. Macker. Mobile Ad-hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations. *IETF MANET, RFC 2501*, 1999.

[17] S. Corson, S. Papademetriou, P. Papadopoulos, V. Park, and A. Qayyum. Internet MANET Encapsulation Protocol (IMEP) specification. *IETF MANET, Internet Draft*, 1999.

[18] M. Cremonini, E. Damiani, S. D. C. D. Vimercati, and P. Samarati. Security, Privacy and Trust in Mobile Systems and Applications. In M. Pagani, editor, *Mobile and Wireless Systems beyond 3G: Managing New Business Opportunities*. IRM Press, 2005.

[19] B. Dahill, B. N. Levine, E. M. Royer, and C. Shields. A Secure Routing Protocol for Ad hoc Networks. In *Proceedings of the International Conference on Network Protocols (ICNP)*, pages 78–87. IEEE Press, 2002.

[20] D. Denning. A New Paradigm for Trusted Systems. In *Proceedings of the ACM New Security Paradigms Workshop*, pages 36–41. ACM Press, 1993.

[21] Y. Desmedt. Some Recent Research Aspects of Threshold Cryptography. In *Proceedings of the First International Workshop on Information Security*, pages 158–173. Springer-Verlag, 1998.

[22] P. Dewan and P. Dasgupta. Trusting Routers and Relays in Ad hoc Networks. In *Proceedings of the International Conference on Parallel Processing Workshops*, pages 351–358. IEEE Press, 2003.

[23] R. Diestel. *Graph Theory, Second Edition*. Springer-Verlag, 2000.

[24] E. W. Dijkstra. A Note on Two Problems in Connection with Graphs. *Numerische Mathematik*, 1:269–271, 1959.

[25] G. Elofson. Developing Trust with Intelligent Agents: An Exploratory Study. In *Proceedings of the First International Workshop on Trust*, pages 125–139, 1998.

[26] L. Eschenauer, V. D. Gligor, and J. S. Baras. On Trust Establishment in Mobile Ad-hoc Networks. In *Proceedings of the 10th International Workshop on Security Protocols*, pages 47–66. Springer, 2004.

[27] L.M. Feeney. An Energy Consumption Model for Performance Analysis of Routing Protocols for Mobile Ad hoc Networks. *Mobile Networks and Applications*, 6(3):239–249, 2001.

[28] E. Gafni and D. Bertsekas. Distributed Algorithms for Generating Loop-free routes in Networks with Frequently Changing Topology. *IEEE Transactions on Communications*, 29(1):11–18, 1981.

[29] S. Garfinkel. *PGP: Pretty Good Privacy*. O'Reilly and Associates, Inc, 1995.

[30] N.E. Hastings and P.A. McLean. TCP/IP Spoofing Fundamentals. In *Proceedings of the IEEE Fifteenth Annual International Phoenix Conference on Computers and Communications*, pages 218–224. IEEE Press, 1996.

[31] G. Holland and N. Vaidya. Analysis of TCP Performance over Mobile Ad hoc Networks. In *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 219–230. ACM Press, 1999.

[32] Y. C. Hu and D. B. Johnson. Caching Strategies in On-demand Routing Protocols for Wireless Ad hoc Networks. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (MobiCom)*, pages 231–242. ACM Press, 2000.

[33] Y-C. Hu, D. B. Johnson, and A. Perrig. SEAD: Secure Efficient Distance Vector Routing for Mobile Wireless Ad hoc Networks. In *Proceedings of the Fourth*

IEEE Workshop on Mobile Computing Systems and Applications, pages 3–13. IEEE Press, 2002.

[34] Y. C. Hu, A. Perrig, and D. B. Johnson. Ariadne: A Secure On-Demand Routing Protocol for Ad hoc Networks. In *Proceedings of the Eighth Annual International Conference on Mobile Computing and Networking (MobiCom)*, pages 12–23. ACM Press, 2002.

[35] Y. C. Hu, A. Perrig, and D. B. Johnson. Rushing Attacks and Defense in Wireless Ad hoc Network Routing Protocols. In *Proceedings of the ACM Workshop on Wireless Security*, pages 30–40. ACM Press, 2003.

[36] J. P. Hubaux, L. Buttyan, and S. Capkun. The Quest for Security in Mobile Ad hoc Networks. In *Proceedings of the 2nd ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pages 146–155. ACM Press, 2001.

[37] IEEE. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications 802.11. 1997.

[38] IETF. The Internet Engineering Task Force. *http://www.ietf.org/*, 1986.

[39] ITU. Security Architecture for Open Systems Interconnection for CCITT Applications. *X800*, 1991.

[40] S. Jacobs and M. S. Corson. MANET Authentication Architecture. *IETF MANET, Internet Draft*, 1999.

[41] D. B. Johnson, D. A. Maltz, and Y. Hu. The Dynamic Source Routing Protocol for Mobile Ad hoc Networks (DSR). *IETF MANET, Internet Draft*, 2003.

[42] C. M. Jonker and J. Treur. Formal Analysis of Models for the Dynamics of Trust Based on Experiences. In *Proceedings of the 9th European Workshop on*

Modelling Autonomous Agents in a Multi-Agent World: MultiAgent System Engineering, pages 221–231. Springer-Verlag, 1999.

[43] A. Josang. The Right Type of Trust for Distributed Systems. In *Proceedings of the ACM New Security Paradigms Workshop*, pages 119–131. ACM Press, 1996.

[44] A. Josang. Prospectives for Modelling Trust in Information Security. In *Proceedings of the Second Australasian Conference on Information Security and Privacy*, pages 2–13. Springer-Verlag, 1997.

[45] A. Josang. A Logic for Uncertain Probabilities. *International Journal of Uncertainity, Fuzziness and Knowledge-Based Systems*, 9(3):279–311, 2001.

[46] A. Josang, E. Gray, and M. Kinateder. Analysing Topologies of Transitive Trust. In *Proceedings of the Workshop of Formal Aspects of Security and Trust (FAST)*, pages 9–22, 2003.

[47] O. Kachirski and R. Guha. Intrusion Detection using Mobile Agents in Wireless Ad hoc Networks. In *Proceedings of the IEEE Workshop on Knowledge Media Networking (KMN)*, pages 153–158. IEEE Press, 2002.

[48] B. Karp and H.T. Kung. GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, pages 243–254. ACM Press, 2000.

[49] Y. B. Ko and N.H. Vaidya. GeoTORA: A Protocol for Geocasting in Mobile Ad hoc Networks. In *Proceedings of the International Conference on Network Protocols (ICNP)*, pages 240–250. IEEE Press, 2000.

[50] P. Lamsal. Understanding Trust and Security. Technical report, Department of Computer Science, University of Helsinki, Finland, 2001.

[51] S. J. Lee and M. Gerla. AODV-BR: Backup Routing in Ad hoc Networks. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1311–1316. IEEE Press, 2000.

[52] S. J. Lee and M. Gerla. Dynamic Load-Aware Routing in Ad hoc Networks. In *Proceedings of the IEEE International Conference on Communications (ICC)*, volume 10, pages 3206–3210. IEEE Press, 2001.

[53] X. Li and L. Cuthbert. Node-Disjointness based Multipath Routing for Mobile Ad hoc Networks. In *Proceedings of the ACM International Workshop on Performance Evaluation of Wireless Ad-hoc, Sensor and Ubiquitous Networks*, pages 23–29. ACM Press, 2004.

[54] Y. Lu, W. Wang, Y. Zhong, and B. Bhargava. Study of Distance Vector Routing Protocols for Mobile Ad hoc Networks. In *Proceedings of the IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 187–194. IEEE Press, 2003.

[55] H. Luo, P. Zerfos, J. Kong, S. Lu, and L. Zhang. Self-Securing Ad hoc Wireless Networks. In *Proceedings of the Seventh International Symposium on Computers and Communications (ISCC)*, page 567. IEEE Press, 2002.

[56] M. K. Marina and S. R. Das. On-Demand Multi Path Distance Vector Routing in Ad hoc Networks. In *Proceedings of the Ninth International Conference on Network Protocols (ICNP)*, pages 14–23. IEEE Press, 2001.

[57] S. P. Marsh. *Formalizing Trust as a Computational Concept.* PhD thesis, University of Stirling, 1994.

[58] S. Marti, T. Giuli, K. Lai, and M. Baker. Mitigating Routing Misbehavior in Mobile Ad hoc Networks. In *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking (MobiCom)*, pages 255–265. ACM Press, 2000.

[59] R. C. Mayer, J. H Davis, and F. D. Schoorman. An Integrative Model of Organizational Trust. *Academy of Management Review*, 20(3):709–734, 1995.

[60] J. Menezes, A, P. C. Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[61] P. Michiardi and R. Molva. CORE: A Collaborative Reputation Mechanism to Enforce Node Cooperation in Mobile Ad hoc Networks. In *Proceedings of the IFIP TC6/TC11 Sixth Joint Working Conference on Communications and Multimedia Security*, volume 228, pages 107–121. Kluwer Academic Publishers, 2002.

[62] P. Michiardi and R. Molva. Simulation-based Analysis of Security Exposures in Mobile Ad hoc Networks. In *Proceedings of the European Wireless Conference*. VDE-Verlag, 2002.

[63] S. Mueller, R. Tsang, and D. Ghosal. Multipath Routing in Mobile Ad hoc Networks: Issues and Challenges. In *Lecture Notes in Computer Science*, pages 209–234. Springer, 2004.

[64] A. Nasipuri and S.R. Das. On-demand Multipath Routing for Mobile Ad hoc Networks. In *Proceedings of the Eight International Conference on Computer Communications and Networks*, pages 64–70. IEEE Press, 1999.

[65] NS. The Network Simulator. *http://www.isi.edu/nsnam/ns/*, 1989.

[66] P. Obreiter, B. Konig-Reis, and M. Klein. Stimulating Cooperative Behavior of Autonomous Devices - An Analysis of Requirements and Existing Approaches. In *Proceedings of the Second International Workshop on Wireless Information Systems (WIS)*, pages 71–82. ICEIS Press, 2003.

[67] R. D. Oliveira and T. Braun. TCP in Wireless Mobile Ad hoc Networks. Technical Report IAM-02-003, University of Berne, 2002.

[68] P. Papadimitratos and Z. J. Haas. Secure Routing for Mobile Ad hoc Networks. In *Proceedings of the SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS)*, pages 1–13, 2002.

[69] P. Papadimitratos and Z. J. Haas. Secure Link State Routing for Mobile Ad hoc Networks. In *Proceedings of the Symposium on Applications and the Internet Workshops (SAINT)*, pages 379–383. IEEE Press, 2003.

[70] V. Park and S. Corson. Temporally Ordered Routing Algorithm (TORA) version 1 functional specification. *IETF MANET, Internet Draft*, 2001.

[71] V.D. Park and M.S. Corson. A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks. In *Proceedings of the Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, volume 3, pages 1405–1413. IEEE Press, 1997.

[72] C. Perkins, E. M. Royer, and S. Das. Ad hoc On-Demand Distance Vector (AODV) Routing. *IETF RFC 3561*, 2003.

[73] C. E. Perkins and P. Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In *Proceedings of the SIGCOMM Conference on Communications, Architectures, Protocols and Applications*, pages 234–244. ACM Press, 1994.

[74] A. Perrig, R. Canetti, D. Tygar, and D. Song. The TESLA Broadcast Authentication Protocol. *RSA CryptoBytes*, 5(2), 2002.

[75] A. Perrig, Y. C. Hu, and D. B. Johnson. Wormhole Protection in Wireless Ad hoc Networks. Technical Report TR01-384, Department of Computer Science, Rice University, 2001.

[76] A. A. Pirzada, A. Datta, and C. McDonald. Propagating Trust in Ad-hoc Networks for Reliable Routing. In *Proceedings of the International Workshop on Wireless Ad-hoc Networks (IWWAN)*. IEEE Press, 2004.

[77] A. A. Pirzada, A. Datta, and C. McDonald. Trust Based Routing for Ad-hoc Wireless Networks. In *Proceedings of the IEEE International Conference on Networks (ICON)*, volume 1, pages 326–330. IEEE Press, 2004.

[78] A. A. Pirzada, A. Datta, and C. McDonald. Trustworthy Routing with the AODV Protocol. In *Proceedings of the International Networking and Communications Conference (INCC)*, pages 19–24. IEEE Press, 2004.

[79] A. A. Pirzada, A. Datta, and C. McDonald. Trustworthy Routing with the TORA Protocol. In *Proceedings of the AusCERT Asia Pacific Information Technology Security Conference*, pages 23–27, 2004.

[80] A. A. Pirzada, A. Datta, and C. McDonald. Incorporating Trust and Reputation in the DSR Protocol for Dependable Routing. *Elsevier Computer Communications Special Issue on Internet Communications Security*, 29(15):2806–2821, 2006.

[81] A. A. Pirzada and C. McDonald. A Review of Secure Routing Protocols for Ad-hoc Mobile Wireless Networks. In *Proceedings of the 7th International Symposium on DSP for Communication Systems (DSPCS) and 2nd Workshop on the Internet, Telecommunications and Signal Processing (WITSP)*, pages 118–123, 2003.

[82] A. A. Pirzada and C. McDonald. Establishing Trust in Pure Ad-hoc Networks. In *Proceedings of the 27th Australasian Computer Science Conference (ACSC)*, volume 26, pages 47–54. Australian Computer Society, 2004.

[83] A. A. Pirzada and C. McDonald. Kerberos Assisted Authentication in Mobile Ad-hoc Networks. In *Proceedings of the 27th Australasian Computer Science Conference (ACSC)*, volume 26, pages 41–46. Australian Computer Society, 2004.

[84] A. A. Pirzada and C. McDonald. Performance Comparison of Reactive Routing Protocols under Attack Conditions. *International Journal on Wireless and Optical Communications (IJWOC)*, 2(2):163–180, 2004.

[85] A. A. Pirzada and C. McDonald. Secure Routing Protocols for Mobile Ad-hoc Wireless Networks. In T. A. Wysocki, A. Dadej, and B. J. Wysocki, editors, *Advanced Wired and Wireless Networks*, pages 57–80. Springer, 2004.

[86] A. A. Pirzada and C. McDonald. Securing Ad-hoc Networks Using Cryptographic Hashes. *Journal of China Information Security*, pages 436–445, 2004.

[87] A. A. Pirzada and C. McDonald. Deploying Trust Gateways to Reinforce Dynamic Source Routing. In *Proceedings of the 3rd International IEEE Conference on Industrial Informatics*. IEEE Press, 2005.

[88] A. A. Pirzada and C. McDonald. Inherent Robustness of Reactive Routing Protocols against Selfish Attacks. In *Proceedings of the International Workshop on Wireless Ad-hoc Networks (IWWAN)*. IEEE Press, 2005.

[89] A. A. Pirzada and C. McDonald. Detecting and Evading Wormholes in Mobile Ad-hoc Wireless Networks. *International Journal of Network Security*, 3(2):188–199, 2006.

[90] A. A. Pirzada and C. McDonald. Reliable Routing in Ad-hoc Networks using Direct Trust Mechanisms. In M. X. Cheng and D. Li, editors, *Advances in Wireless Ad Hoc and Sensor Networks in book series Signals and Communication Technology*. Springer, 2007.

[91] A. A. Pirzada, C. McDonald, and A. Datta. Dependable Dynamic Source Routing without a Trusted Third Party. In *Proceedings of the 28th Australasian Computer Science Conference (ACSC)*, volume 38, pages 79–85. Australian Computer Society, 2005.

[92] A. A. Pirzada, C. McDonald, and A. Datta. Reliable Link Reversal Routing for Mobile Ad-hoc Wireless Networks. In *Proceedings of the IEEE International Conference on Networks (ICON)*, volume 1, pages 234–239. IEEE Press, 2005.

[93] A. A. Pirzada, C. McDonald, and A. Datta. Performance Comparison of Trust-Based Reactive Routing Protocols. *IEEE Transactions on Mobile Computing*, 5(6):695–710, 2006.

[94] A. A. Rahman. The PGP Trust Model. *EDI-Forum: The Journal of Electronic Commerce*, 1997.

[95] A. A. Rahman and S. Hailes. A Distributed Trust Model. In *Proceedings of the ACM New Security Paradigms Workshop*, pages 48–60. ACM Press, 1997.

[96] A. A. Rahman and S. Hailes. Supporting Trust in Virtual Communities. In *Proceedings of the 33rd Hawaii International Conference on System Sciences*, volume 6, pages 1769–1777. IEEE Press, 2000.

[97] R. L. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[98] E. M. Royer, S. J. Lee, and C.E. Perkins. The Effects of MAC Protocols on Ad hoc Network Communication. In *Proceedings of the Wireless Communications and Networking Conference (WCNC)*, volume 2, pages 543–548. IEEE Press, 2000.

[99] E. M. Royer and C. K. Toh. A Review of Current Routing Protocols for Ad hoc Mobile Wireless Networks. *IEEE Personal Communications Magazine*, 6(2):46–55, 1999.

[100] R. Shirey. Internet Security Glossary. *IETF RFC 2828*, 2000.

[101] B. Shneiderman. Designing Trust into Online Experiences. *Communications of the ACM*, 43(12):57–59, 2000.

[102] F. Stajano. The Resurrecting Duckling - What Next? In *Revised Papers from the 8th International Workshop on Security Protocols*, pages 204–214. Springer-Verlag, 2001.

[103] F. Stajano and R. Anderson. The Resurrecting Duckling: Security Issues for Ad hoc Wireless Networks. In *Proceedings of the 7th International Workshop on Security Protocols*, pages 172–194. Springer-Verlag, 1999.

[104] W. Stallings. *Network Security Essentials*. Prentice Hall, 2000.

[105] A. S. Tanenbaum. *Computer Networks*. Prentice Hall, fourth edition, 2002.

[106] A. P. Tayal and L. M. Patnaik. An Address Assignment for the Automatic Configuration of Mobile Ad hoc Networks. *Personal and Ubiquitous Computing*, 8(1):47–54, 2004.

[107] W. Wang, Y. Lu, and B. Bhargava. On Vulnerability and Protection of Ad hoc On-demand Distance Vector Protocol. In *Proceedings of the International Conference on Telecommunication (ICT)*, pages 375–382. IEEE Press, 2003.

[108] Y. Wang and J. Vassileva. Bayesian Network-based Trust Model. In *Proceedings of the IEEE/WIC International Conference on Web Intelligence*, pages 372–378. IEEE Press, 2003.

[109] C. Ware, T. Wysocki, and J. Chicharo. Hidden Terminal Jamming Problems in IEEE 802.11 Mobile Ad hoc Networks. In *Proceedings of the IEEE International Conference on Communications (ICC)*, volume 1, pages 261–265. IEEE Press, 2001.

[110] A. Weimerskirch and G. Thonet. A Distributed Light-Weight Authentication Model for Ad-hoc Networks. In *Proceedings of the 4th International Conference on Information Security and Cryptology*, pages 341–354. Springer-Verlag, 2002.

[111] K. Wu and J. Harms. Load-Sensitive Routing for Mobile Ad hoc Networks. In *Proceedings of the Tenth International Conference on Computer Communications and Networks*, pages 540–546. IEEE Press, 2001.

[112] L. Xiong and L. Liu. PeerTrust : A Trust Mechanism for an Open Peer-to-Peer Information System. Technical Report GIT-CC-02-29, College of Computing, Georgia Institute of Technology, 2002.

[113] P. W. Yau and C.J. Mitchell. Reputation Methods for Routing Security for Mobile Ad hoc Networks. In *Proceedings of the Joint First Workshop on Mobile Future and Symposium on Trends in Communications (SympoTIC)*, pages 130–137. IEEE Press, 2003.

[114] W. Ye, J. Heidemann, and D. Estrin. An Energy-efficient MAC protocol for Wireless Sensor Networks. In *Proceedings of the IEEE Conference on Computer Communications (Infocom)*, pages 1567–1576. IEEE Press, 2002.

[115] S. M. Yen and Y. Zheng. Weighted One-Way Hash Chain and Its Applications. In *Proceedings of the Third International Workshop on Information Security*, pages 135–148. Springer-Verlag, 2000.

[116] S. Yi, P. Naldurg, and R. Kravets. Security-Aware Ad hoc Routing for Wireless Networks. In *Proceedings of the 2nd ACM International Symposium on Mobile Ad hoc Networking and Computing (MobiHoc)*, pages 299–302. ACM Press, 2001.

[117] W. H. Yuen and R. D. Yates. Inter-relationships of Performance Metrics and System Parameters in Mobile Ad hoc Networks. In *Proceedings of the IEEE*

Military Communications Conference (MILCOM), volume 1, pages 519–524. IEEE Press, 2002.

[118] M. G. Zapata. Secure Ad hoc On-Demand Distance Vector (SAODV) Routing. *IETF MANET, Internet Draft*, 2001.

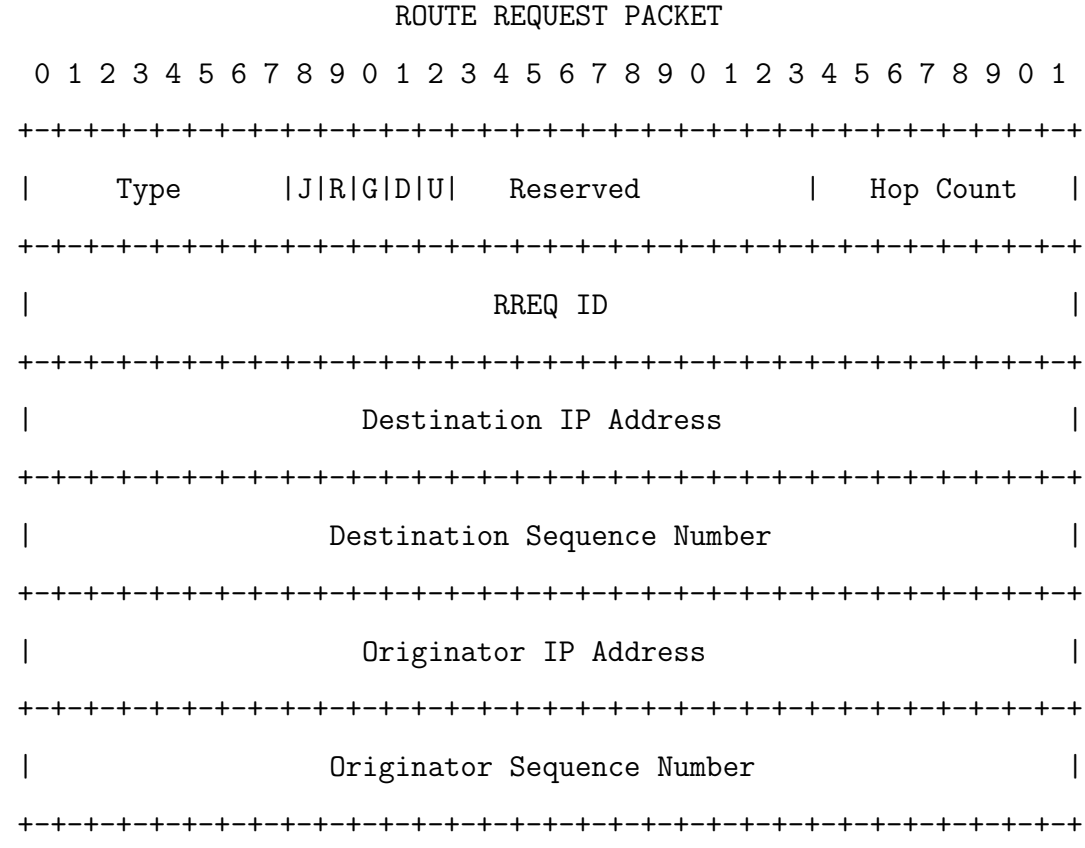
[119] Y. Zhang and W. Lee. Intrusion Detection in Wireless Ad hoc Networks. In *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking (MobiCom)*, pages 275–283. ACM Press, 2000.

[120] L. Zhou and Z. J. Haas. Securing Ad hoc Networks. *IEEE Network Magazine*, 13(6):24–30, 1999.

Appendix A

AODV Packet Headers

This appendix shows the standard AODV `ROUTE REQUEST`, `ROUTE REPLY` and `ROUTE ERROR` packet headers along with the description of their various fields, as given in the IETF RFC 3561 (July 2003) [72].



Type

1

J

Join flag; reserved for multicast.

R

Repair flag; reserved for multicast.

G

Gratuitous RREP flag; indicates whether a gratuitous RREP should be unicast to the node specified in the Destination IP Address field.

D

Destination only flag; indicates only the destination may respond to this RREQ.

U

Unknown sequence number; indicates the destination sequence number is unknown.

Reserved

Sent as 0; ignored on reception.

Hop Count

The number of hops from the Originator IP Address to the node handling the request.

RREQ ID

A sequence number uniquely identifying the particular RREQ when taken in conjunction with the originating node's IP address.

Destination IP Address

The IP address of the destination for which a route is desired.

Destination Sequence Number

The latest sequence number received in the past by the originator for any route towards the destination.

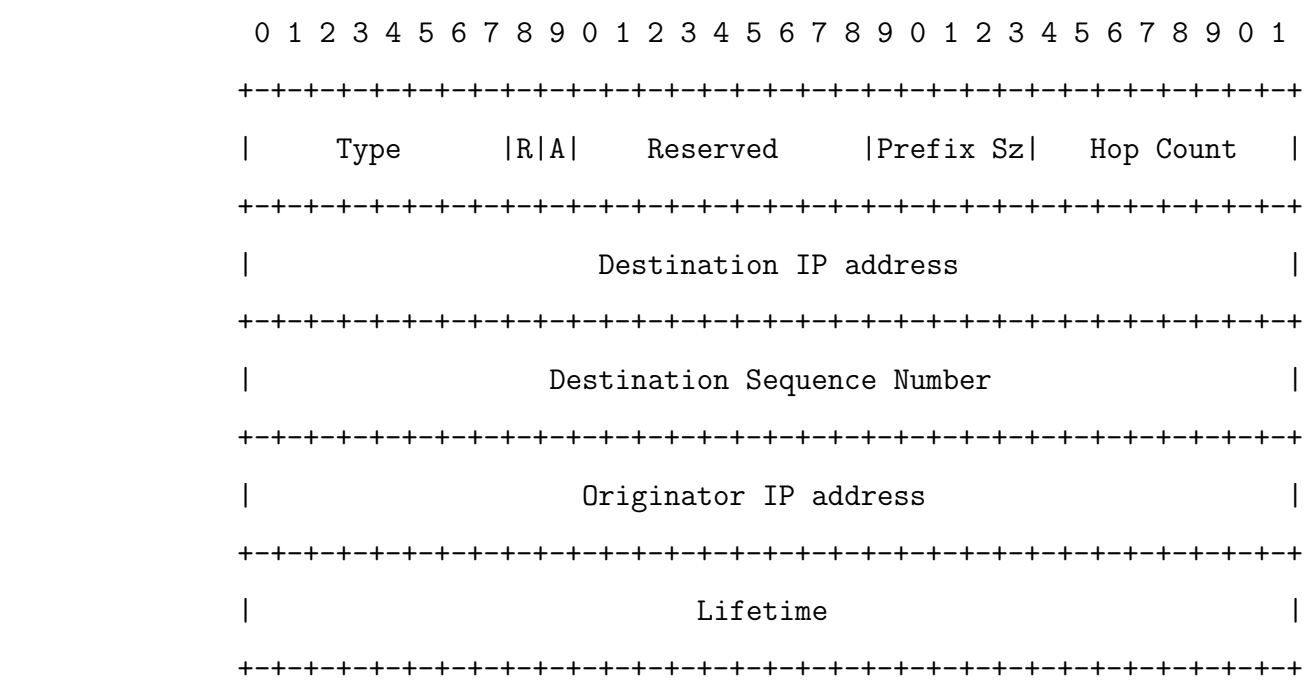
Originator IP Address

The IP address of the node which originated the Route Request.

Originator Sequence Number

The current sequence number to be used in the route entry pointing towards the originator of the route request.

ROUTE REPLY PACKET



Type

2

R

Repair flag; used for multicast.

A

Acknowledgment required.

Reserved

Sent as 0; ignored on reception.

Prefix Size

If nonzero, the 5-bit Prefix Size specifies that the indicated next hop may be used for any nodes with the same routing prefix (as defined by the Prefix Size) as the requested destination.

Hop Count

The number of hops from the Originator IP Address to the Destination IP Address. For multicast route requests this indicates the number of hops to the multicast tree member sending the RREP.

Destination IP Address

The IP address of the destination for which a route is supplied.

Destination Sequence Number

The destination sequence number associated to the route.

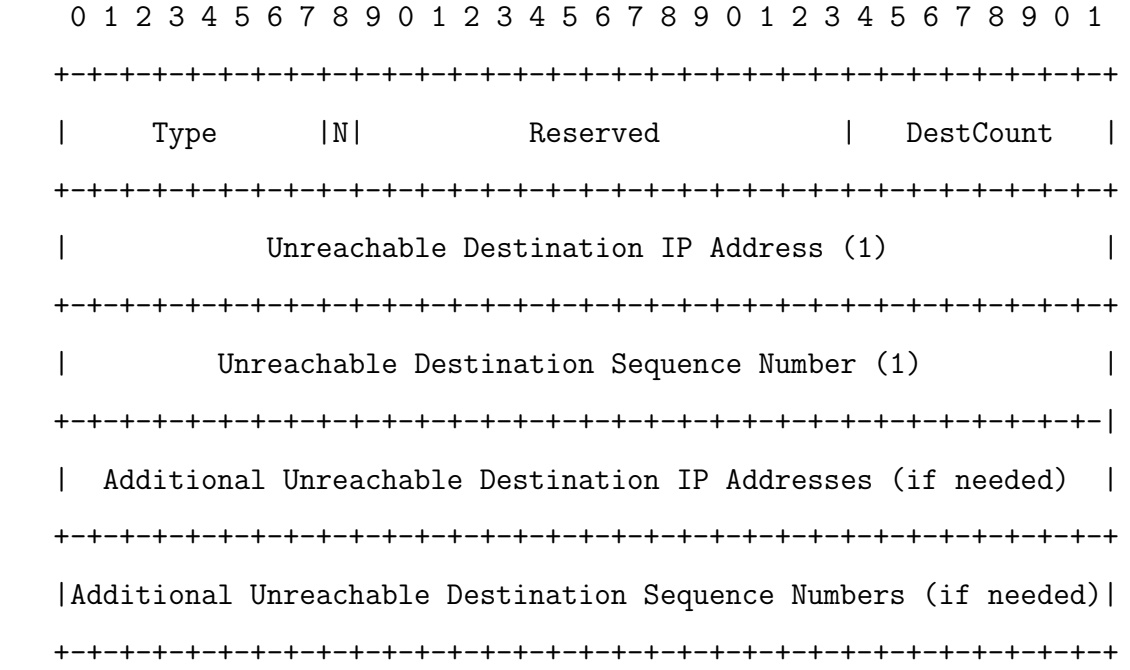
Originator IP Address

The IP address of the node which originated the RREQ for which the route is supplied.

Lifetime

The time in milliseconds for which nodes receiving the RREP consider the route to be valid.

ROUTE ERROR PACKET



Type

3

N

No delete flag; set when a node has performed a local repair of a link, and upstream nodes should not delete the route.

Reserved

Sent as 0; ignored on reception.

DestCount

The number of unreachable destinations included in the message; MUST be at least 1.

Unreachable Destination IP Address

The IP address of the destination that has become unreachable due to a link break.

Unreachable Destination Sequence Number

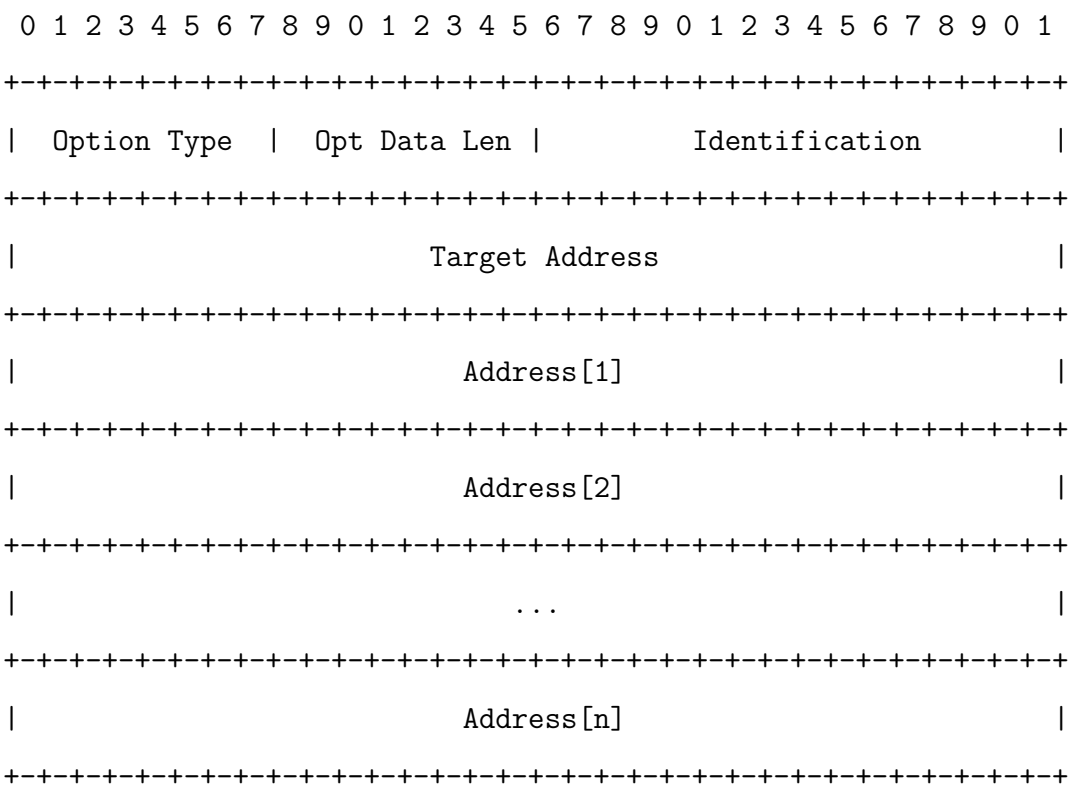
The sequence number in the route table entry for the destination listed in the previous Unreachable Destination IP Address field.

Appendix B

DSR Packet Headers

This appendix shows the standard DSR `ROUTE REQUEST`, `ROUTE REPLY` and `ROUTE ERROR` packet headers along with the description of their various fields, as given in the IETF Internet Draft (draft-ietf-manet-dsr-09.txt, April 2003) [41].

ROUTE REQUEST PACKET



Option Type

2

Opt Data Len

8-bit unsigned integer. Length of the option, in octets, excluding the Option Type and Opt Data Len fields.

Identification

A unique value generated by the initiator (original sender) of the Route Request. Nodes initiating a Route Request generate a new Identification value for each Route Request.

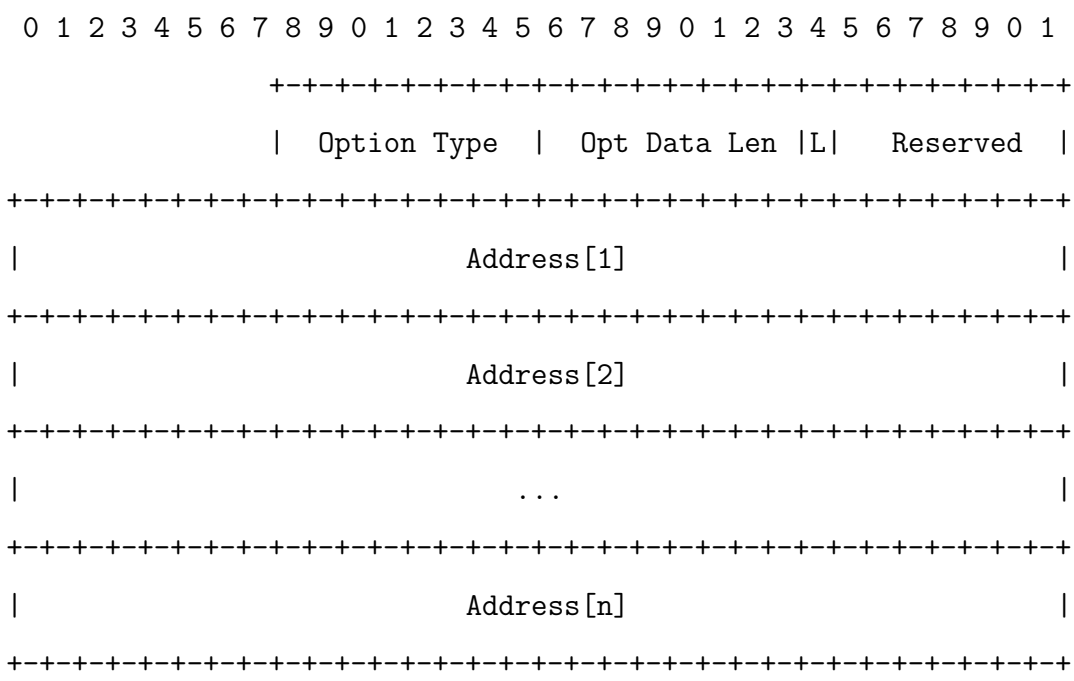
Target Address

The address of the node that is the target of the Route Request.

Address[1..n]

Address[i] is the address of the i-th node recorded in the Route Request option. The address given in Address[1] is thus the address of the first node on the path after the initiator. The number of addresses present in this field is indicated by the Opt Data Len field in the option.

ROUTE REPLY PACKET



Option Type

Nodes not understanding this option will ignore this option.

Opt Data Len

8-bit unsigned integer. Length of the option, in octets, excluding the Option Type and Opt Data Len fields.

Last Hop External (L)

Set to indicate that the last hop given by the Route Reply (the link from Address[n-1] to Address[n]) is actually an arbitrary path in a network external to the DSR network; the exact route outside the DSR network is not represented in the Route Reply.

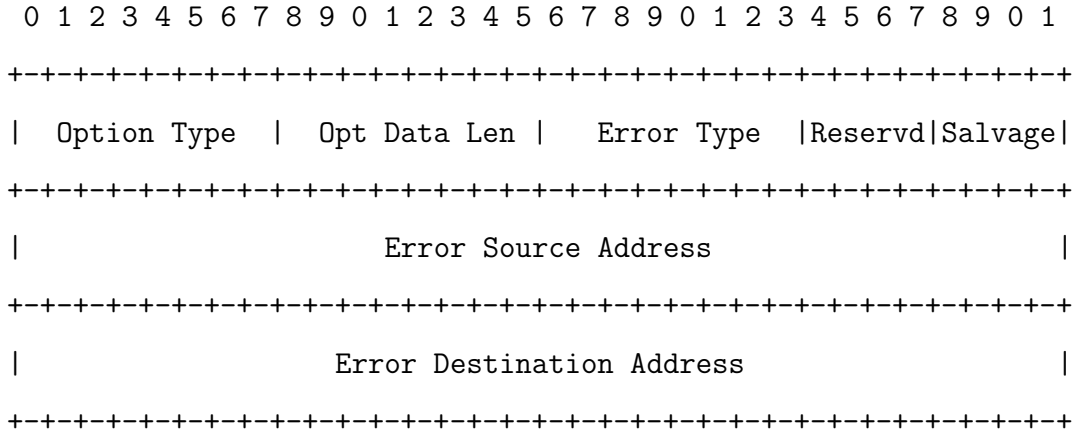
Reserved

Must be sent as 0 and ignored on reception.

Address[1..n]

The source route being returned by the Route Reply. The route indicates a sequence of hops, originating at the source node specified in the Destination Address field of the IP header of the packet carrying the Route Reply, through each of the Address[i] nodes in the order listed in the Route Reply. The number of addresses present in the Address[1..n] field is indicated by the Opt Data Len field in the option.

ROUTE ERROR PACKET



Option Type

Nodes not understanding this option will ignore this option.

Opt Data Len

8-bit unsigned integer. Length of the option, in octets, excluding the Option Type and Opt Data Len fields.

Error Type

The type of error encountered. Currently, the following type values are defined:

1 = NODE_UNREACHABLE

2 = FLOW_STATE_NOT_SUPPORTED

3 = OPTION_NOT_SUPPORTED

Reserved

Must be sent as 0 and ignored on reception.

Salvage

A 4-bit unsigned integer. Copied from the Salvage field in the DSR Source Route option of the packet triggering the Route Error.

Error Source Address

The address of the node originating the Route Error.

Error Destination Address

The address of the node to which the Route Error must be delivered.

Type-Specific Information

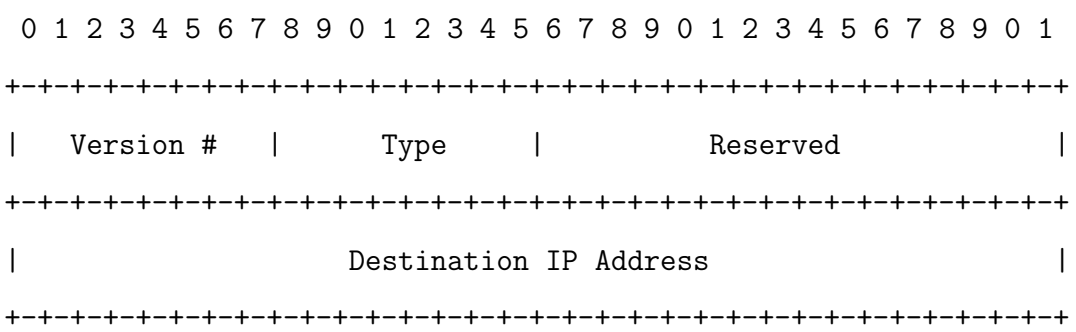
Information specific to the Error Type of this Route Error message.

Appendix C

TORA Packet Headers

This appendix shows the standard TORA Query (QRY), Update (UPD) and Clear (CLR) packet headers along with the description of their various fields, as given in the IETF Internet Draft (draft-ietf-manet-tora-spec-04.txt, July 2001) [70].

QRY PACKET



Version #

The TORA version number. This specification documents version 1.

Type

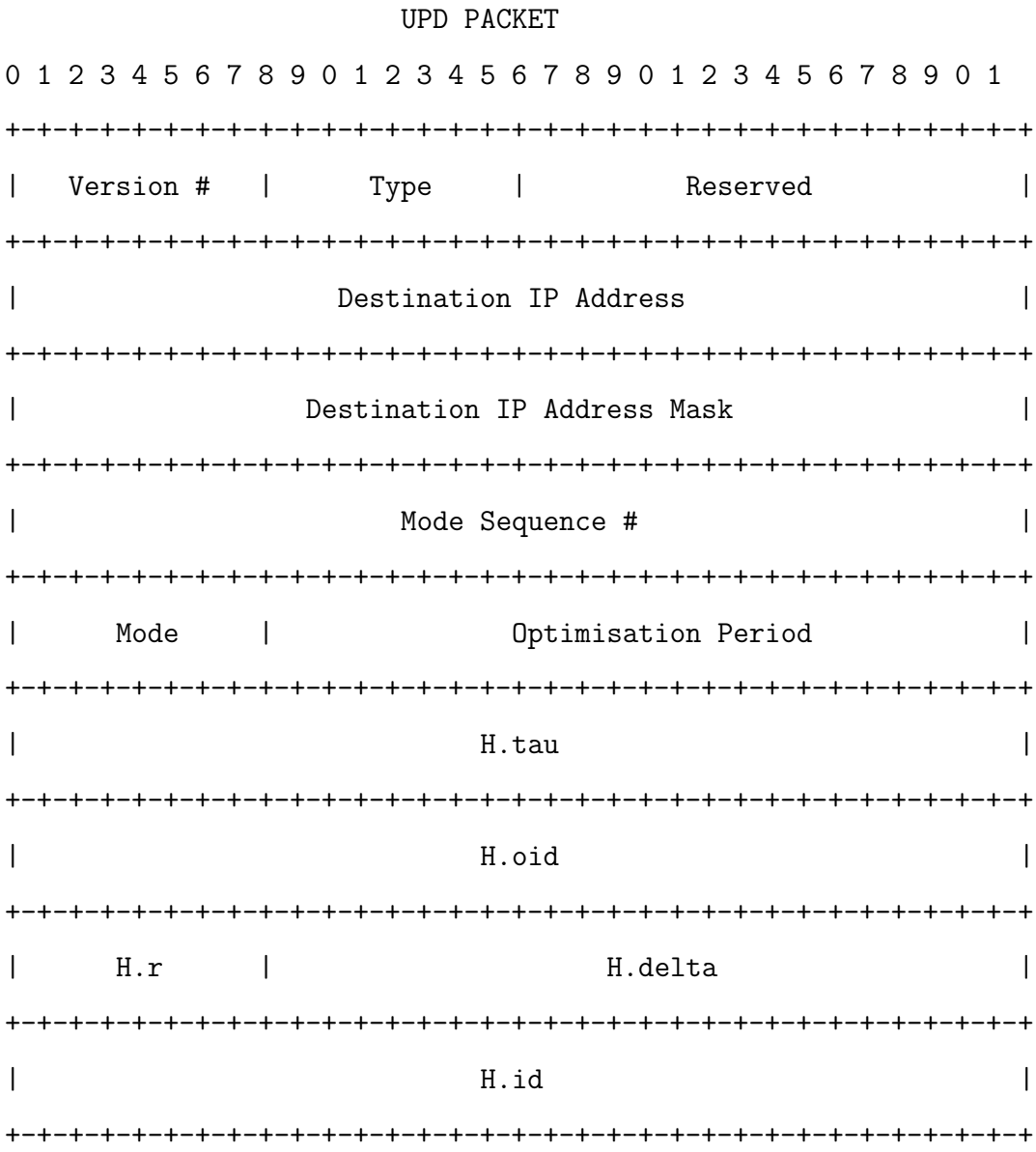
The TORA packet type. For QRY packet this field is set to 1.

Reserved

Field reserved for future use.

Destination IP Address

The IP address for which a route is being requested.



Version #

The TORA version number. This specification documents version 1.

Type

The TORA packet type. For UPD packet this field is set to 2.

Reserved

Field reserved for future use.

Destination IP Address

The IP address for which a route is being requested.

Destination IP Address Mask

The network mask associated with the destination IP address.

Mode Sequence #

Sequence number associated with the subsequent mode and optimisation period fields. Used for propagation of most recent mode state and to ensure each router processes mode information at most once.

Mode

The mode of operation associated with the destination IP address and mask. This field is used to indicate reactive/proactive routing and also the type (if any) of optimisations used for the destination.

Optimisation Period

The period for optimisation packets originated by the destination.

H.tau

The H.tau value, associated with the destination IP address and mask, of the router sending the UPD.

H.oid

The H.oid value, associated with the destination IP address and mask, of the router sending the UPD.

H.r

The H.r value, associated with the destination IP address and mask, of the router sending the UPD.

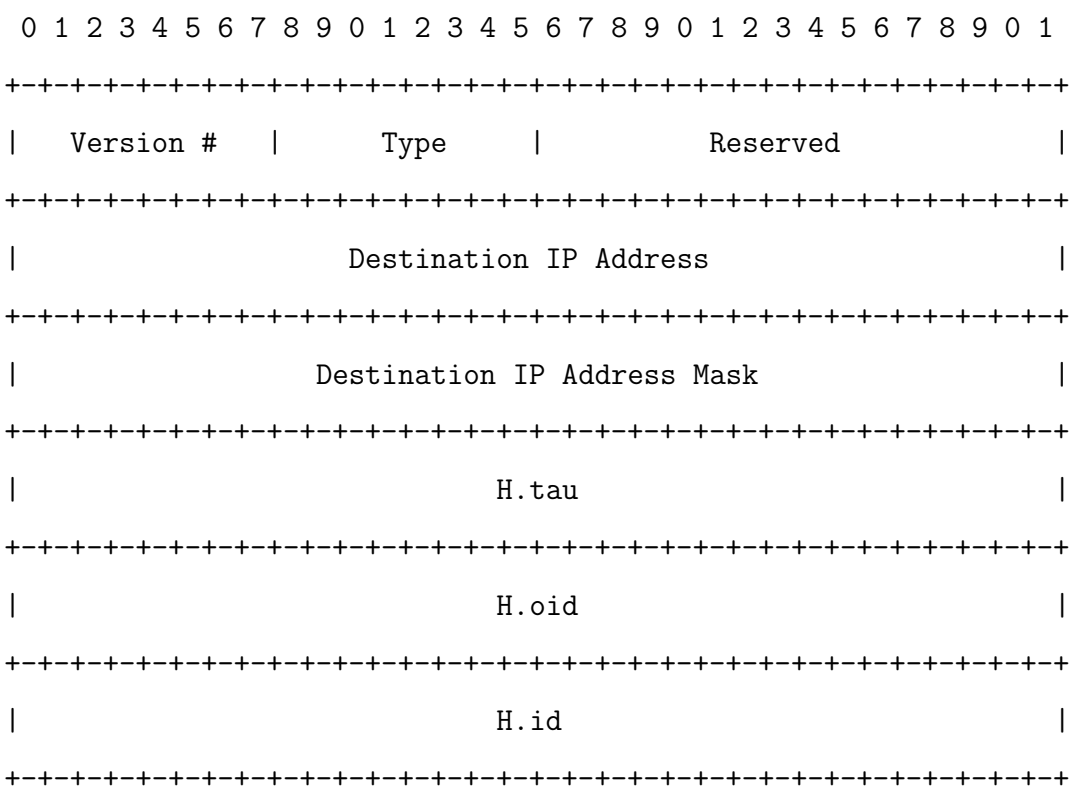
H.delta

The H.delta value, associated with the destination IP address and mask, of the router sending the UPD.

H.id

The H.id value, associated with the destination IP address and mask, of the router sending the UPD (i.e., unique router ID).

CLR PACKET



Version #

The TORA version number. This specification documents version 1.

Type

The TORA packet type. For CLR packet this field is set to 3.

Reserved

Field reserved for future use.

Destination IP Address

The IP address for which a route is being requested.

Destination IP Address Mask

The network mask associated with the destination IP address.

H.tau

The H.tau value, associated with the destination IP address and mask, of the router sending the UPD.

H.oid

The H.oid value, associated with the destination IP address and mask, of the router sending the UPD.

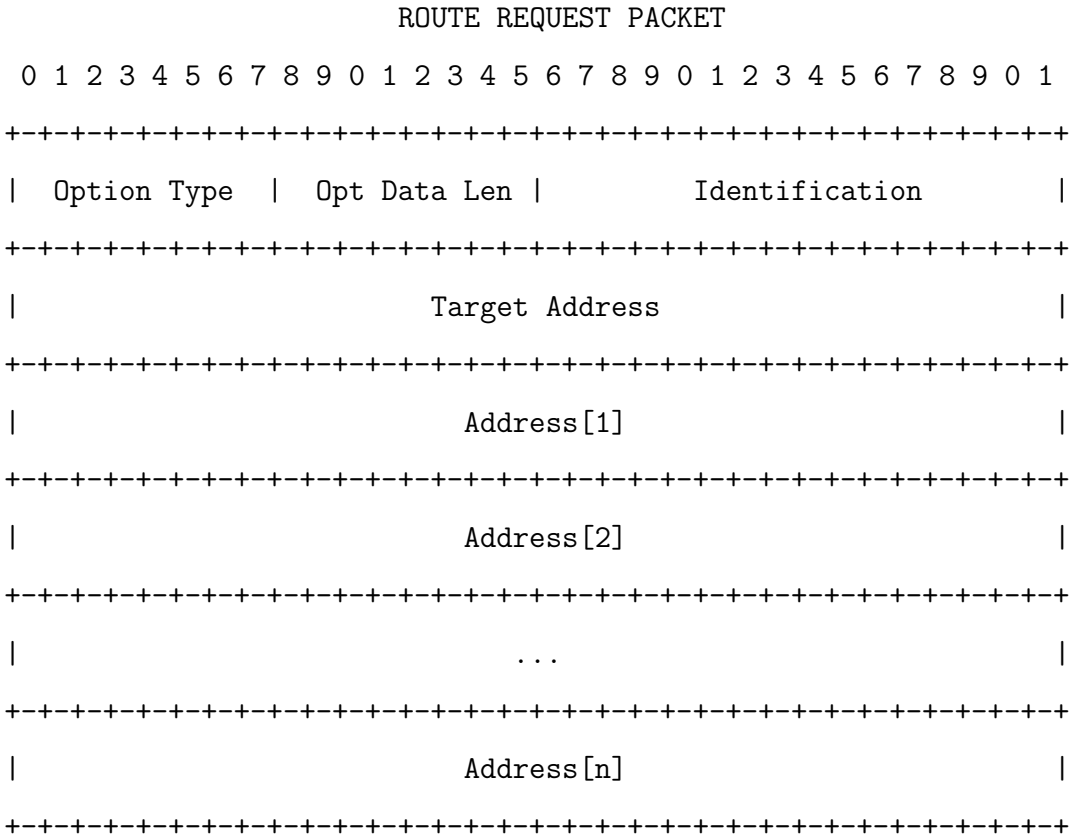
H.id

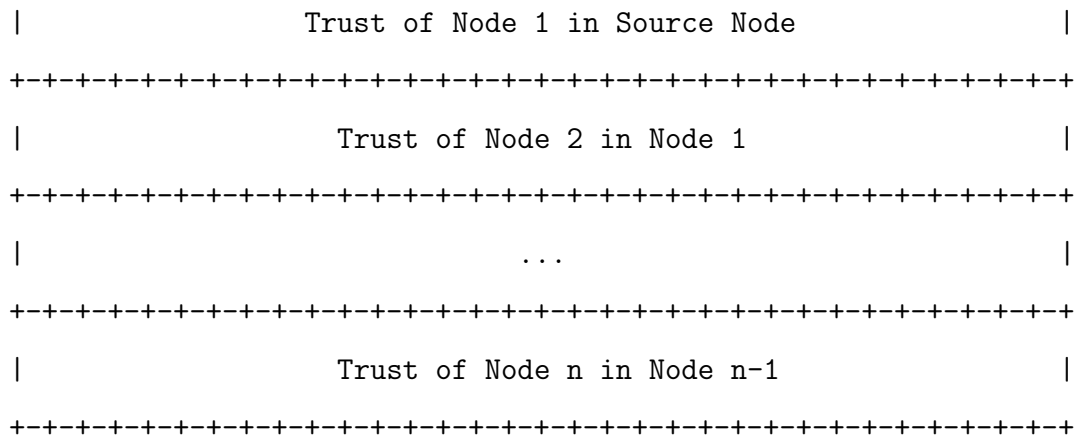
The H.id value, associated with the destination IP address and mask, of the router sending the UPD (i.e., unique router ID).

Appendix D

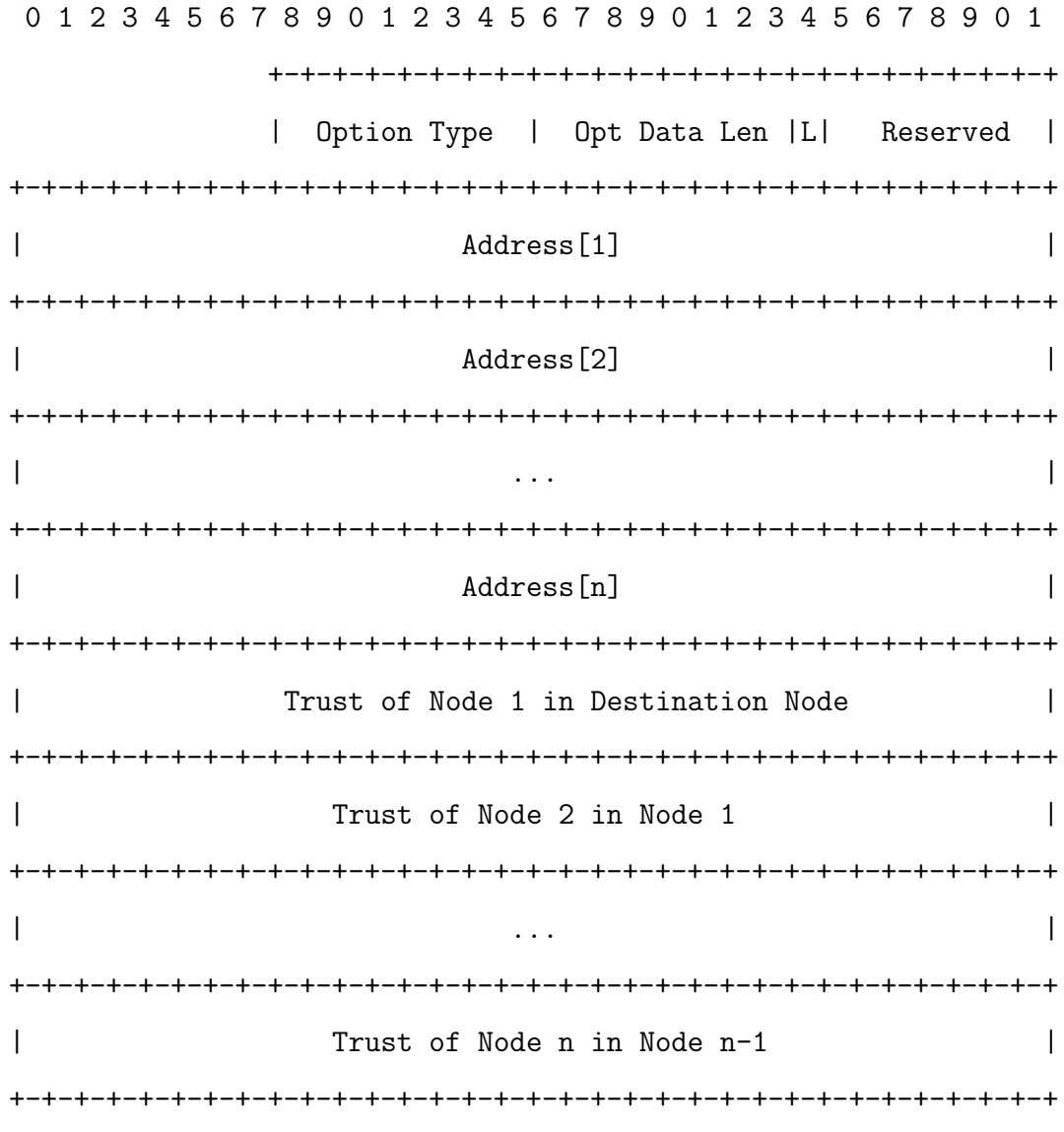
Modified DSR Packet Headers

This appendix shows the integration of the reputation exchange mechanism with the DSR `ROUTE REQUEST` and `ROUTE REPLY` packet headers.





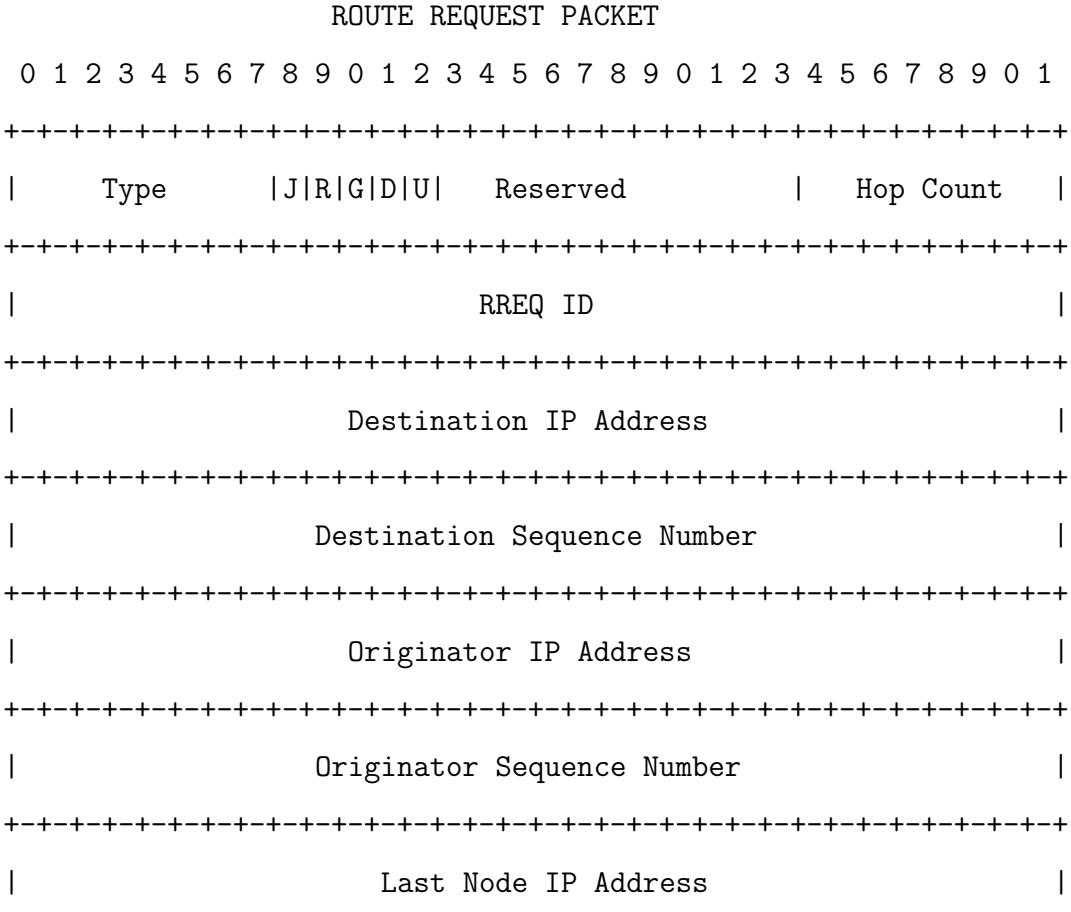
ROUTE REPLY PACKET

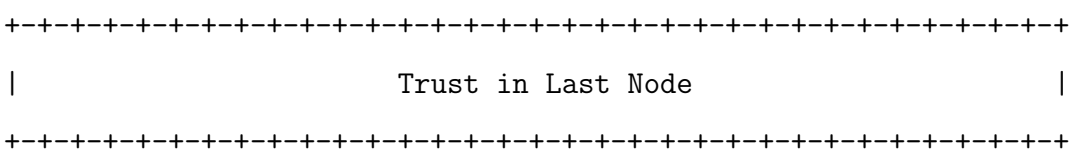


Appendix E

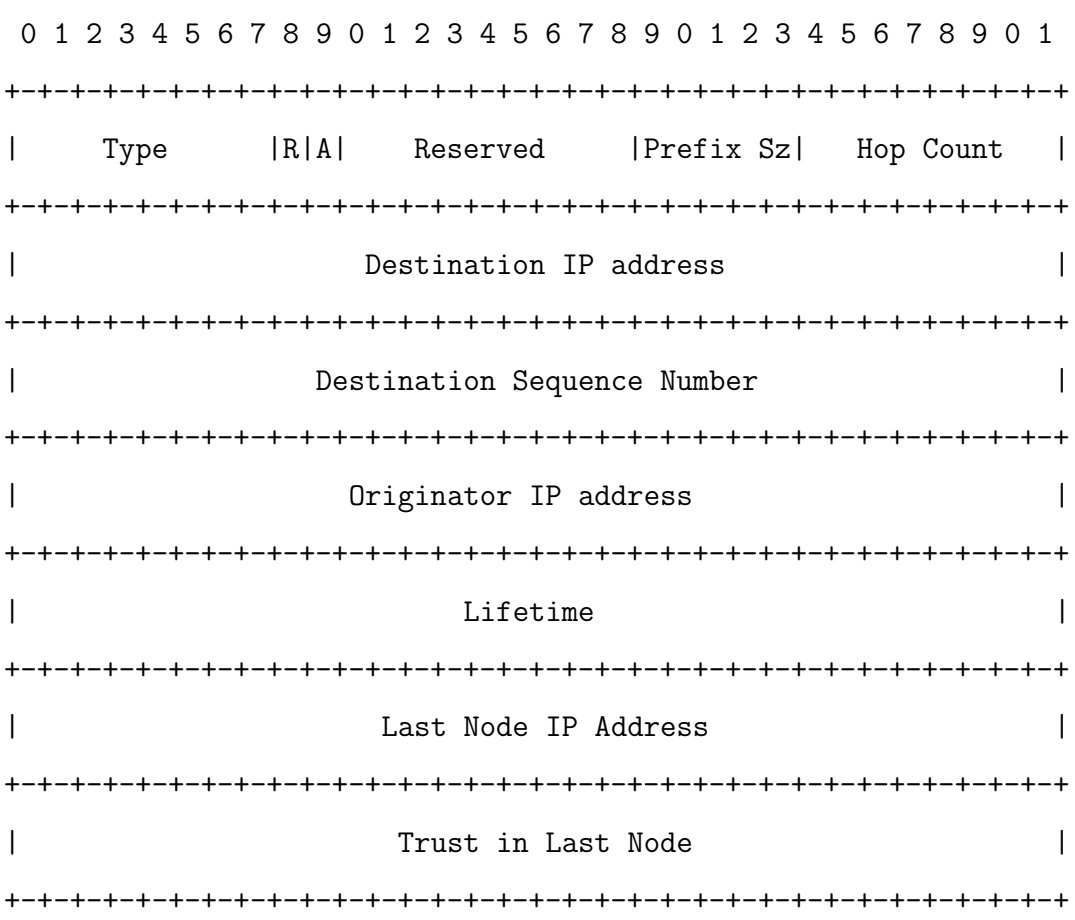
Modified AODV Packet Headers

This appendix shows the integration of the reputation exchange mechanism with the AODV `ROUTE REQUEST` and `ROUTE REPLY` packet headers.





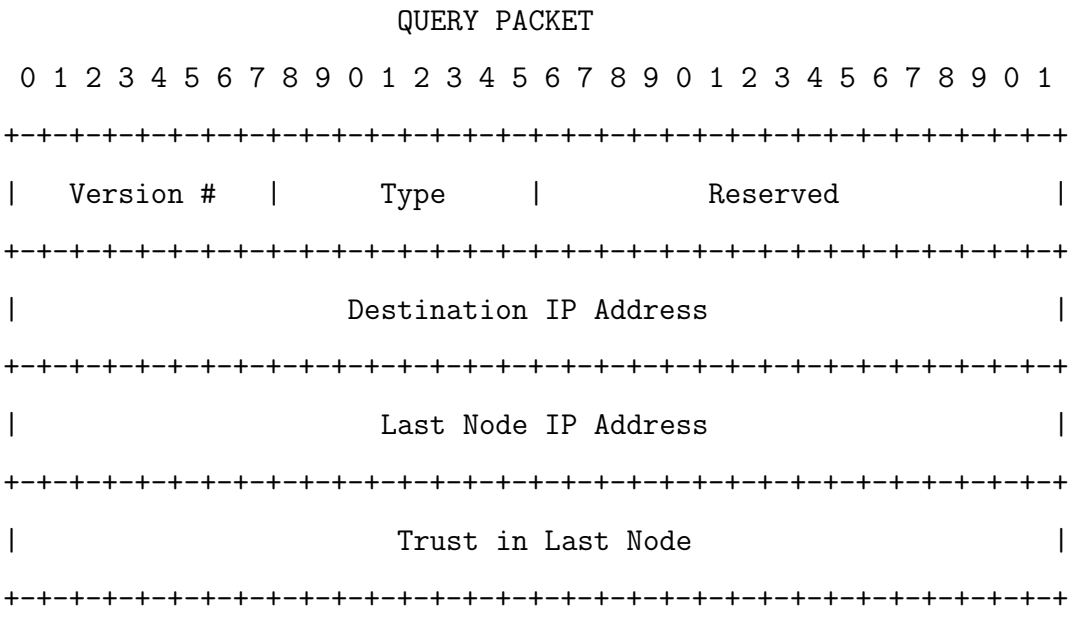
ROUTE REPLY PACKET

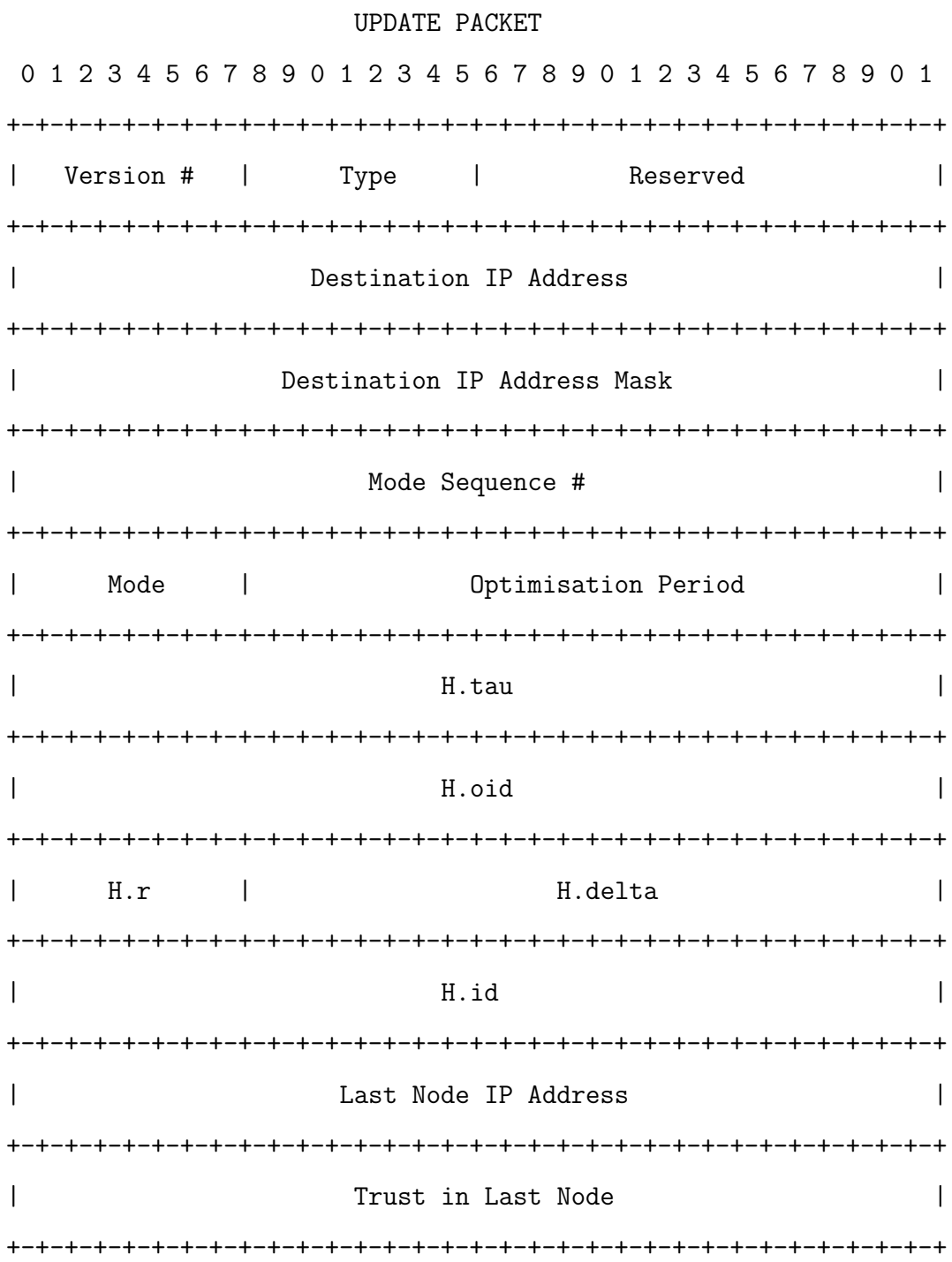


Appendix F

Modified TORA Packet Headers

This appendix shows the integration of the reputation exchange mechanism with the TORA QRY and UPD packet headers.




UPDATE PACKET
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
Version #
Type
Reserved
Destination IP Address
Destination IP Address Mask
Mode Sequence #
Mode
Optimisation Period
H.tau
H.oid
H.r
H.delta
H.id
Last Node IP Address
Trust in Last Node

Appendix G

Modified Dijkstra Algorithm

This appendix shows the pseudo-code for the modified Dijkstra, from the standard shortest path to the most trustworthy path algorithm.

```
t(n)    The trust value assigned to node n by the source node s
p       Set of predecessors for each node on the most
        trustworthy path from the source
S       Set of nodes whose most trustworthy path from
        the source has been found
Q       Set of nodes whose most trustworthy path from
        the source has yet to be determined


initialise(Node s)
for all v nodes in the link-cache
{
    t[v] = 0
    p[v] = NULL
    t(s) = INFINITY
}
```

```
extract-most-trustworthy(Q)
{
    find the most trustworthy node in Q
    remove it from Q
    return the node
}

relax-neighbours(Node u, Node v, Tuv)
{
    if t(v) < t(u) + Tuv
    {
        t(v) = t(u) + Tuv
        p(v) = u
    }
}

dijkstra (Node s)
{
    initialise(s)
    S = Q = {}
    add s to Q
    while Q is not empty
    {
        u = extract-most-trustworthy(Q)
        add u to S
        for each node v which is an immediate neighbour of u
            relax-neighbours(u, v, Tuv)
    }
}
```

Appendix H

AODV Pseudo Code

This appendix shows the NS-2 pseudo-code for the trusted AODV routing protocol.

```
int PP = NEUTRAL;                    // Initial integrity level
int PA = NEUTRAL;                    // Initial selfishness level
int TUI=5;                           // Trust Update Interval

AODV::forward (Packet *p)            //AODV forwarding mechanism
{
    if (p.route = TRUE)              //Find a route for the packet
    {
        buffer_packet(p);
        p.path=find_max_trust_path();//Find the maximum trust path
        schedule forwarding(p);      //Schedule packet dispatch
        Trust_Scheduler(TRUE);       //Trust_Scheduler started
    }
}
```

```
AODV::tap(const Packet *packet)     //AODV promiscuous mode tap
{
    PA++;                           //Increase Pkt Acknowledgement
    if (verify_packet_integrity(p)) //Check Packet Modifications
        PP++;                       //Increase Packet Precision
    else
        PP--;                       //Decrease Packet Precision
    Trust_Scheduler(FALSE);         //Trust_Scheduler disabled
}


AODV::buffer_packet(Packet *p)      //Buffering before forwarding
{
    packet_copy=p->copy();          //Buffer packet
}


Trust_Scheduler::handle(bool status)//Called every TUI seconds
{
    packet_copy=NULL;               //Clear packet buffer
    PA--;                           //Decrease Pkt Acknowledgement
    Trust_Scheduler(FALSE);         //Trust_Scheduler disabled
}


AODV::verify_packet_integrity(Packet *p)//Verify PP
{
    return (p.compare(packet_copy));//Return TRUE if integrity
                                    //check passes
}
```

```
AODV::find_max_trust_path()         //Retrieve maximum trust path
{
    double min_trust_level;         //Trust Threshold
    AODV_path temp_path =NULL;
    for (all paths)
    {
        if (path.Trust > min_trust_level)
        {
            temp_path=path;
            min_trust_level=path.Trust;
        }
    }
    return temp_path;
}
```

Appendix I

DSR Pseudo Code

This appendix shows the NS-2 pseudo-code for the trusted DSR routing protocol.

```
int PP = NEUTRAL;                    // Initial integrity level
int PA = NEUTRAL;                    // Initial selfishness level
int TUI=5;                           // Trust Update Interval

DSRAgent::handleForwarding(SRPacket &p)
                                     // DSR forwarding mechanism
{
    if (p.route = TRUE )             // Find a route for the packet
    {
        buffer_packet(p)             // Buffer packet SR header
        schedule forwarding(p);      // Schedule packet dispatch
        Trust_Scheduler(TRUE);       // Trust_Scheduler started
    }
}
```

```
DSRAgent::buffer_packet(Packet *p)  // Buffering before forwarding
{
    hdr_sr *buff_srh=hdr_sr::access(p); // Buffer packet SR Header
}

DSRAgent::tap(const Packet *packet) // DSR promiscuous mode tap
{
    PA++;                           // Packet forwarded
    if (verify_packet_integrity(p)) // Detect Modification Attacks
        PP++;
    else
        PP--;
    Trust_Scheduler(FALSE);         //Trust_Scheduler disabled
}

DSRAgent::verify_packet_integrity(packet *p)
                                    // Verify Packet Integrity
{
    hdr_sr *srh = hdr_sr::access(p);
    return (srh == buff_srh);       //Return TRUE if integrity
                                    //check passes
}

Trust_Scheduler::handle(bool status)// Called every TUI seconds
{
    buff_srh=NULL;                  // Clear SR buffer
    PA--;                           // Black/Grey Hole detected
    Trust_Scheduler(FALSE);         // Trust_Scheduler disabled
}
```

```
DSRAgent::handlePktWithoutSR(SRPacket& p, bool retry)
                                // Find Source Route for a packet
{
    if (route_cache->findRoute(p.dest, p.route))
        sendOutPacketWithRoute(p, true);
                                // Send out packet with route
    else
        getRouteForPacket(p, retry);
                                // Initiate new route discovery
}

LinkCache::findRoute(ID dest, Path& route)
{
    dijkstra();                 // Execute modified Dijkstra
}
```

Appendix J

TORA Pseudo Code

This appendix shows the NS-2 pseudo-code for the trusted TORA routing protocol.

```
int PP = NEUTRAL;                // Initial integrity level
int PA = NEUTRAL;                // Initial selfishness level
int TUI=5;                       // Trust Update Interval

toraAgent::rt_resolve(Packet *p)//Find route for the packet
{
    TORADestination *td;
    TORANeighbor *tn;
    tn = td->nb_find_most_trusted_next_hop();
    if(tn == 0)                  //No neighbour found
    {
        rqueue.enque(p);         //Queue the packet
        sendQRY(td);             //Send a QRY for the destination
    }
    else
    {
```

```
        forward(p, tn);         //Forward packet to neighbour
    }
}

TORADest::nb_find_most_trusted_next_hop()
                                //Find most trustworthy neighbour
{
    TORANeighbor *tn = nblist.lh_first;
    TORANeighbor *tn_min = 0;
    double tt;                  //Trust Threshold
    for( ; tn; tn = tn->link.le_next)
    {                           //Compare trust levels
                                //and then heights
        if ((direct_trust(tn->index) > tt) ||
        (direct_trust(tn->index) == tt &&
        tn_min->height.compare(&tn->height) < 0 ))
        {
            tn_min = tn;
            tt=direct_trust(tn->index) ;
        }
    }
    return tn_min;
}

toraAgent::forward(Packet *p)   //TORA forwarding mechanism
{
    buffer_packet(p)            //Buffer packet
    schedule forwarding(p);     //Schedule packet dispatch
    Trust_Scheduler(TRUE);      //Trust_Scheduler started
}
```

```
toraAgent::buffer_packet(Packet *p)
                                //Packet buffering before forwarding
{
    packet_copy=p->copy();      //Buffer packet
}


toraAgent::tap(const Packet *packet)
                                //TORA promiscuous mode tap
{
    PA++;                       //Increase Packet Acknowledgement
    if (verify_packet_integrity(p))
                                //Check Packet Modifications
        PP++;                   //Increase Packet Precision
    else
        PP--;                   //Decrease Packet Precision
    Trust_Scheduler(FALSE);     //Trust_Scheduler disabled
}


toraAgent::verify_packet_integrity(Packet *p)
                                //Verify PP
{
    return (p.compare(packet_copy));
                                //Return TRUE if integrity
                                //check passes
}
```

```
Trust_Scheduler::handle(bool status)
                                //Called every TUI seconds
{
    packet_copy=NULL;           //Clear packet buffer
    PA--;                       //Decrease Packet Acknowledgement
    Trust_Scheduler(FALSE);     //Trust_Scheduler disabled
}
```