Trust Based Dynamic Source Routing in Mobile Ad Hoc Networks

John Keane

A dissertation submitted to the University of Dublin in partial fulfilment of the requirements for the degree of Master of Science in Computer Science

September 16th 2002

Declaration

I declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed: _____

John Keane

Date: September 16th 2002

Permission to lend and/or copy

I agree that Trinity College Library may lend or copy this dissertation upon request.

Signed: _____

John Keane

Date: September 16th 2002

Abstract

The notion of an ad-hoc network is a new paradigm that allows mobile hosts (called nodes) to communicate without relying on a predefined infrastructure to keep the network connected. Most nodes are assumed to be mobile and communication is generally assumed to be wireless. Ad-hoc networks are collaborative in the sense that each node is assumed to relay packets for other nodes that will in return relay their packets. Thus, all nodes in an ad-hoc network form part of the network's routing infrastructure. The mobility of nodes in an ad-hoc network means that both the population and the topology of the network are highly dynamic.

In traditional networks, the network infrastructure is provided by a few trusted operators. This is no longer the case in an ad-hoc network where every mobile node becomes part of the network fabric. It is thus necessary to define new security mechanisms for the routing protocols in such networks.

This project involves the analysis, design, implementation, and evaluation of one such mechanism based on the notion of trust. Each node records their experience with routing through particular nodes. Nodes will be regarded as trustworthy, if the relayed packets reach their final destination. An evaluation of the trustworthiness of the other nodes allows the sending node to identify and use the route with the highest probability of packets reaching their destination.

The trust based route selection system developed is evaluated using the NS-2 network simulator. Results have proved that malfunctioning or malicious nodes can be clearly identified by other nodes in the network. However, the route selection algorithm that makes use of this identification information is only outperforming existing routing protocols in a restricted number of instances.

Acknowledgments

I would like to thank my supervisor Christian Jensen for the help and support he offered me throughout the year. I would also like to thank the MSc NDS class for making this year a very enjoyable one.

Contents

List of Figures

Chapter 1

Introduction

1.1 Preamble

Wireless communications technologies are developing rapidly and are undergoing a tremendous rise in popularity. The continuing rise in processing power and the miniaturisation of hardware, coupled with the fact that people want to increasingly retain a high degree of connectivity to the Internet, is leading this substantial shift to mobile computing.

In today's age of pervasive mobile computing, wireless technologies play a key role in connecting ubiquitous devices. The primitive wireless services originally offered by mobile phones and pagers have given way to a sophisticated set of services offered by wireless local area networks and intricate telecommunications networks (GPRS & UMTS/3G) which will soon be accessible by mobile phone and personal digital assistant (PDA).

As wireless devices of all types proliferate and their cost is reduced, users will increasingly demand access to information services and applications even when they are not able to connect to the traditional Internet or it is not necessary to do so. For example, people with laptops at a conference in a hotel may wish to communicate in an ad-hoc manner, without having to make use of any fixed infrastructure network. The concept of such an ad-hoc network is new and it is this type of mobile computing network that forms the basis for this thesis.

1.2 Ad-Hoc Networks

The Internet Engineering Task Force Working Group on Mobile Ad-Hoc Networks [IETF MANET WG] defines an ad-hoc network as:

> A "mobile ad hoc network" (MANET) is an autonomous system of mobile routers (and associated hosts) connected by wireless links – the union of which form an arbitrary graph. The routers are free to move randomly and organise themselves arbitrarily; thus the network's wireless topology may change rapidly and unpredictably. Such a network may operate in a stand-alone fashion, or may be connected to the larger Internet.

The notion of an ad-hoc network is a new paradigm in which a collection of mobile nodes collaborate to form a temporary wireless network. Communication is achieved without reliance on a predefined fixed infrastructure. Both the topology and population of an ad-hoc network are highly dynamic due to the mobility of the nodes involved.

Ad-hoc networks are collaborative in the sense that each node is assumed to forward packets for other nodes and will in turn get its packets forwarded for it. It is thus evident that all nodes in an ad-hoc network from part of the routing infrastructure, i.e. all nodes act as routers. This means that traditional routing protocols are inadequate for ad-hoc networks and a number of new routing protocols have been developed for such networks (see chapter 2).

Figure 1 shows a scenario in which node A transmits data to node C via node B, since A's transmission range is not sufficient to send data directly to C.

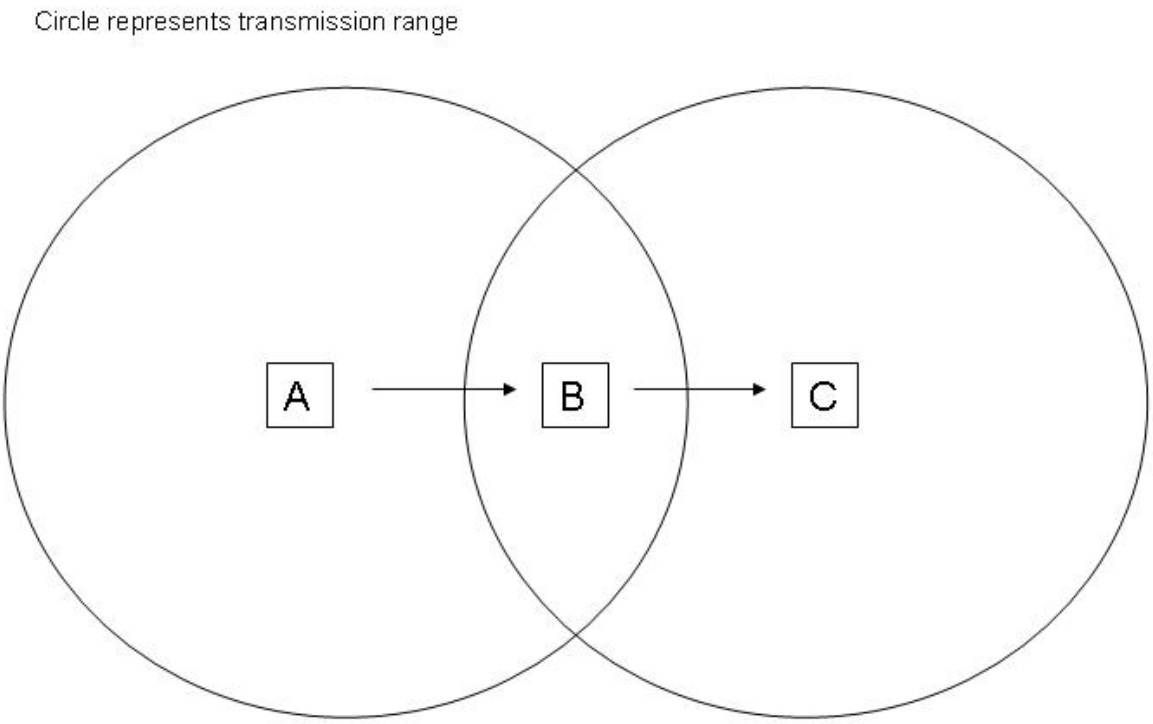


Figure 1: A simple ad-hoc network

This process of routing data through intermediate nodes until the destination node is reached is known as multi-hop. Most packet switching networks including the Internet work in a multi-hop fashion. With traditional networks, both wireline and mobile wireless, the intermediate nodes through which data is sent in order to reach the destination node are routers that are owned and managed by trusted operators. However, in an ad-hoc network the intermediate nodes through which data is sent may be unknown and may potentially be malfunctioning or malicious. Such nodes may disrupt service in the network. This problem is the essential motivation for this thesis and is described and discussed in detail in chapter 2.

There are a large number of potential applications for ad-hoc networks and some of the most cited include: military communications in a battlefield scenario where no infrastructure is present, disaster relief or rescue operations, and conferencing.

1.3 Objective

The basic objective of this project is to design, implement, and evaluate an extension to an existing routing protocol for ad-hoc networks that will address some of the security and reliability vulnerabilities that were mentioned above. These vulnerabilities will be examined in further depth in chapter 2 and at that point both the motivations and the objectives of this project will be more clearly highlighted.

1.4 Document Structure

This document consists of the following chapters:

Chapter 1: Introduction gives a brief insight into mobile communications and ad-hoc networks.

Chapter 2: Literature Review examines key technologies and issues that are relevant to this study.

Chapter 3: Design describes the architecture and operation of the trust based route selection system under study.

Chapter 4: Implementation looks at the manner in which the design is implemented and the technologies that are used.

Chapter 5: Evaluation presents and discusses the results of the simulations that were carried out in order to determine the success of the new system.

Chapter 6: Conclusions summarises the findings of this thesis and suggest future work that could be done in this area.

Chapter 2

Literature Review

The aim of this chapter is to provide a basis and motivation for the design and implementation of a trust based route selection system by exploring and outlining some of the key associated technologies, issues, and problems. Information on relevant background and state of the art technologies is presented. The reliability and security problems associated with routing in ad-hoc networks will be exposed and concepts that will aid in the design of the new system will be examined.

This chapter is organised as follows. Section 2.1 describes traditional security approaches used for both fixed wireline and mobile wireless networks that rely on a fixed infrastructure. Section 2.2 discusses routing issues in ad-hoc networks and describes the most prominent routing protocols. DSR and some of its optimisations are examined in depth. Section 2.3 outlines the security challenges facing ad-hoc networks and exposes the security vulnerabilities present in such networks. State of the art work that addresses problems with routing security is presented. Section 2.4 and 2.5 document the concept of trust and the use of trust management systems respectively. This chapter is then concluded with a summary in section 2.6.

2.1 Traditional Security Approaches

Computer security is primarily concerned with the maintenance of three characteristics: confidentiality, integrity, and availability. Confidentiality, which is often referred to as privacy or secrecy, ensures that the assets of a system are accessible only by authorised parties. This can be achieved by encrypting data and by

controlling access to data. Integrity implies that assets can be modified only by authorised parties and only in an authorised manner. Integrity can be enforced by rigorous control of who can access which resources in what ways. Availability refers to the fact that an authorised party should not be prevented from accessing objects to which legitimate access should be allowed. Denial of service, the opposite of availability, means that assets are inaccessible to authorised parties. Availability is a complex subject and the security community is only beginning to understand what availability implies and how to ensure it. Most of the success in computer security has been in the areas of confidentiality and integrity [Pfleeger].

In traditional networks, such as an infrastructure-based local area network, various security mechanisms are used to achieve confidentiality, integrity, and availability. Most of these mechanisms depend upon either the authenticated identity of the user or some form of credentials that authorise the user to perform certain actions. In many cases these mechanisms rely on a public key infrastructure (PKI).

Security mechanisms implemented by a PKI do provide a capability to build in a certain level of trust into communications. Examples include cryptographic algorithms for privacy and digital signatures, and authentication protocols for proving authenticity. However, these mechanisms do not manage the more general concept of 'trustworthiness'. For example, a signed public-key certificate does not tell you if the owner of the certificate is trustworthy.

The rest of this section will focus on the primary mechanisms used in the traditional approach to achieving the aforementioned three goals of computer security.

2.1.1 Encryption

Encryption is commonly used to provide confidentiality of data. Additionally, because data that cannot be read generally also cannot be altered in a meaningful manner, encryption can be used to achieve integrity. Furthermore, encryption is the

basis for some protocols that ensure availability of resources. Thus, encryption is the core mechanism used to achieve the three goals of computer security [Pfleeger].

In a network environment, encryption can by applied between two hosts or two applications. In *link encryption*, data is encrypted in either the physical layer or the data link layer. The data is thus protected while in transit between two hosts, but is in the form of plaintext while inside the hosts. In *end-to-end encryption,* data is encrypted at the application layer and hence security is provided from one end of a transmission through to the other. If a lower layer should fail to implement security, the secrecy of the data is not endangered. With *link encryption*, encryption is carried out for all transmissions along a particular link. Part of the advantage is lost if a message is only encrypted on some links in a network, and link encryption is therefore used on all links in a network if it is performed at all. By contrast, *end-to-end* encryption is only applied to logical links between processes, and hence the intermediate hosts along the transmission path have no need for cryptographic facilities. Furthermore, the encryption can be done with software and can hence be applied selectively. To satisfy the requirements of an application, there is no need for the lower layers to automatically encrypt all traffic. However, Saltzer et al. [Saltzer] suggest that the communication subsystem may be required to automatically encrypt all traffic to ensure that a misbehaving user or program does not deliberately transmit information that should not be exposed. There is thus a need for both *link encryption* and *end-to-end encryption*, and it is not uncommon for both forms to be applied.

Both symmetric and asymmetric cryptosystems may be used to encrypt data. The Data Encryption Standard (DES) and the Advanced Encryption Standard (AES) are two of the most well know symmetric cryptographic algorithms. However, all such cryptosystems suffer from the complex and costly problem of key distribution. Key distribution is not a problem with asymmetric cryptographic algorithms such as RSA. Public key infrastructures make use of asymmetric public key cryptosystems to allow for confidentiality and for digital signatures. Hybrid cryptosystems such as Secure Socket Layer (SSL) allow the encryption and distribution of a symmetric key via asymmetric encryption. Such cryptosystems are in common use and provide secure communications with little delay. However, despite the confidence placed in such cryptosystems, the necessity to authenticate public keys still persists. A trusted third

party such as a key distribution centre is still necessary and hence the issue of trust is still present.

2.1.2 Authentication

Authentication is the verification of the identity of a remote entity, which may be performed by means of a password, a trusted authentication service, or by using digital certificates. Authentication of an entity allows us to ensure that an unauthorised user does not gain access to data. This is necessary to maintain both confidentiality and integrity of data.

In a small-scale system it is possible for an administrator to assign a unique identifier to each user of the system, but this solution does not scale well. Public key cryptography is used in large-scale distributed systems to create a unique key for every user. We will now examine two public key encryption based systems that are used for authentication in distributed systems.

2.1.2.1 Public Key Certificates

The development of a global key management system became possible as system administrators accepted a trusted third party statement that a particular key belonged to a particular user. The trusted third parties, known as Certification Authorities (CA), create digital certificates that verify that a public key is owned by a particular entity. The issuing of a certificate by a CA implies nothing about the 'trustworthiness' of the key owner involved.

X.509 is the de facto standard digital certificate model that has been adopted on a global scale. It was originally developed by the International Telecommunications Union (ITU) and has since been adopted by the International Standardisation Organisation (ISO). The X.509 standard defines what information is contained in a certificate, and the data format of this information. Public key infrastructures

commonly make use of X.509 certificates, and they are widely visible in web browsers that support the Secure Socket Layer (SSL) protocol.

The X.509 model is a strictly hierarchical trust model. Each entity must have a certificate that is signed by the central certification authority or another authority, which has been directly or indirectly certified by it. This model assumes that certification authorities are organised into a universal "certification authority tree".

The major fault with the X.509 system is that a low-level CA can bind very detailed information about a user in the certificate that they issue, whereas a higher-level CA cannot capture much detailed information. Despite the fact that the CA is used to form a trust relationship, it can never be adequate as an authority for everyone in a large distributed system. Credibility is a major issue, and unfortunately the credibility of a certificate will decrease and its recommendations will increase in uncertainty as the size of the distributed system grows [Khare].

2.1.2.2 Pretty Good Privacy

The Pretty Good Privacy (PGP) system allows users to create a decentralised trust hierarchy by locally binding user identities to public keys. The key management for PGP is ad-hoc and is based on the notion of a "web of trust" [Abdul]. As a user interacts with other users in the distributed system, they will exchange public keys with individuals that they trust, i.e. believe to be authentic. This is achieved by the signing of certificates. The system allows a user to designate users, that they trust, to act as introducers, and to define a level of trust that they associate with each introducer. When a user interacts with an individual who is unknown to them, but who's certificate is signed by a number of known introducers, the user can have a certain amount of confidence in the authenticity of the unknown users public key, based on the trust relationship that exists with the introducers. Due to PGP's lack of official mechanisms for the creation, acquisition, and distribution of certificates, it is considered unreliable for e-commerce, but appropriate for personal communications [Grandison]. The question of why an individual is trusted is not addressed by the PGP system. It should be noted that no trusted third party is used in PGP.

2.1.3 Access Control

Access control systems enforce the policy that the manager of a resource applies to parties that request access to that resource. The policy refers to a particular resource, and specifies who can have access to the resource and what type of access they can have [O Connell]. Access control systems prevent unauthorised access to system resources, which may compromise both confidentiality and integrity of data. Additionally, unauthorised access may impinge availability by denying service to a legitimate user.

The most prevalent traditional mechanism for implementing access control is the Access Control List (ACL). An ACL is associated with every protected resource. It lists all users who have access to that particular resource, and what level of access they have.

There are a number of problematic issues surrounding the use of access control lists. The fundamental problem relates to the flexibility of the system. Whenever a new user has to be added or a privilege granted, the administrator has to directly modify the list. Access control lists have been used extensively in large-scale distributed systems, primarily because of the familiarity of the concept. However, the ACL system does not scale well and is thus not suited to any environment where there are a large number of users. An ad-hoc network may consist of a large number of users and hence the notion of an ACL in such a network has limited potential. The maintaining of individual mappings of identities to access rights is impractical in such circumstances.

2.2 Routing in Ad-Hoc Networks

In an ad-hoc network, nodes cooperate with each other by forwarding data packets for one another around the network. It is this cooperation that enables nodes to send packets to destinations beyond their physical transmission range. However, this multi-hop routing approach does present problems. A transmitting node must depend

on intermediate nodes to be both reliable and trustworthy, in the forwarding of packets. Many different protocols have been proposed to solve the various multi-hop routing problems in ad-hoc networks. However, there is still no single routing protocol that could be characterised as the de-facto standard in the domain of ad-hoc networking.

Each routing protocol is based on different assumptions and different design choices. The first major design choice is that of on-demand vs. periodic advertisements. An on-demand protocol is reactive in nature, in that routing activities are only initiated when a data packet is in need of a route. Traditionally, routing protocols have been proactive, in which case all nodes find (via periodic message passing) all source-destination routes regardless of the use for such routes. The key advantage of the on-demand approach is the reduction of the routing information load, which is usually significant in low-bandwidth wireless links. The second major design choice is source routing vs. hop-by-hop routing. With source routing the sender enumerates all hops on the route to the destination, whereas in hop-by-hop routing the sender is simply aware of the neighbour to which the data packets are sent.

In the rest of this section we will examine four routing protocols that cover a wide range of design choices. Particular emphasis will be placed on the dynamic source routing (DSR) protocol, as that is the protocol that will be extended in this dissertation. Trust based route selection could potentially be added to most ad-hoc network routing protocols, where the sender is presented with a number of potential routes to the destination. However, trust based route selection is particularly amenable to a source routing protocol such as DSR. With such a protocol, the sending node must have knowledge about all the nodes on the route to the destination, not simply its neighbouring nodes.

2.2.1 Dynamic Source Routing (DSR)

Dynamic Source Routing is a routing protocol specifically designed for wireless mobile ad-hoc networks. Its specification was published originally in [Johnson 1]. The latest version of the protocol was made as an Internet draft [Johnson 2].

The key distinguishing feature of DSR is that it uses source routing rather than hop-by-hop routing [Perkins 1]. The sender knows the complete hop-by-hop route to the destination, and these routes are stored in a *Route Cache*. Each data packet carries the source route in the packet header, i.e. the ordered list of nodes through which the packet must pass so as to reach its destination. Intermediate nodes do not need to maintain up to date routing information in order to route the packets forward, since the packets themselves already contain all necessary routing information. Figure 2 shows an example of how source routing works.

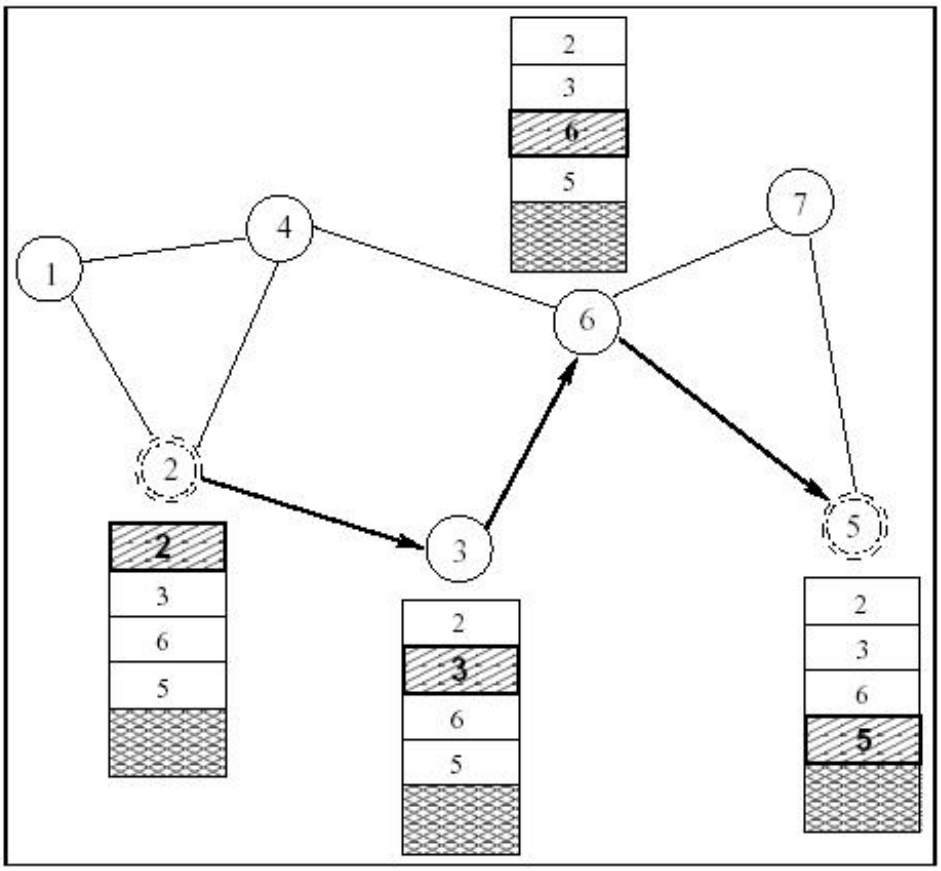


Figure 2: Source Routing in DSR

DSR is an on-demand protocol: it establishes routes dynamically whenever a packet needs to be forwarded to a destination. This is in contrast to distance vector or link state protocols, which try to continuously monitor and update their lists of paths towards possible destinations.

Many assumptions have been made in the design of DSR. It is assumed that all nodes in the network are willing to respect the routing protocol of the network, and in particular that all nodes are assumed to cooperate in the forwarding of packets for other nodes [Johnson 2]. These assumptions generate major security concerns: trusting all nodes in the network to conform to the routing protocol and to forward packets appropriately would expose individual nodes to a wide variety of vulnerabilities. Other assumptions made in the design of DSR include: the speed at which nodes migrate throughout the network is small relative to the rate at which packets are transmitted, communications work equally well in both directions, and nodes have the ability to detect any corrupted packets received.

The DSR protocol consists of two mechanisms: Route Discovery and Route Maintenance. These mechanisms allow for the discovery and maintenance of source routes to a dynamic set of nodes.

2.2.1.1 Route Discovery

Route Discovery is the mechanism by which a node wishing to send a packet to a destination node obtains a source route to this destination node. This process is only carried out if the *route cache* does not already contain the source route. Route Discovery is achieved by the broadcasting of a *route request* to all nodes within communication range. Each *route request* message contains the identity of the initiator of the request, a unique request identification, and a route record of the nodes through which the request has been forwarded. This request is flooded through the network (i.e. continuously relayed from each node to all of its neighbours) and is eventually answered by a *route reply* packet from either the destination node or another node that knows a route to the destination. To ensure that a *route request* does not propagate *ad infinitum* throughout the network, a limit is placed on the number of route requests made for any packet. This limit is embedded in the *route request* message. If the limit is exceeded, the packet will be discarded. Figure 3 illustrates an example where initiator node 2 starts a route discovery to find a path (or multiple paths) to destination node 5.

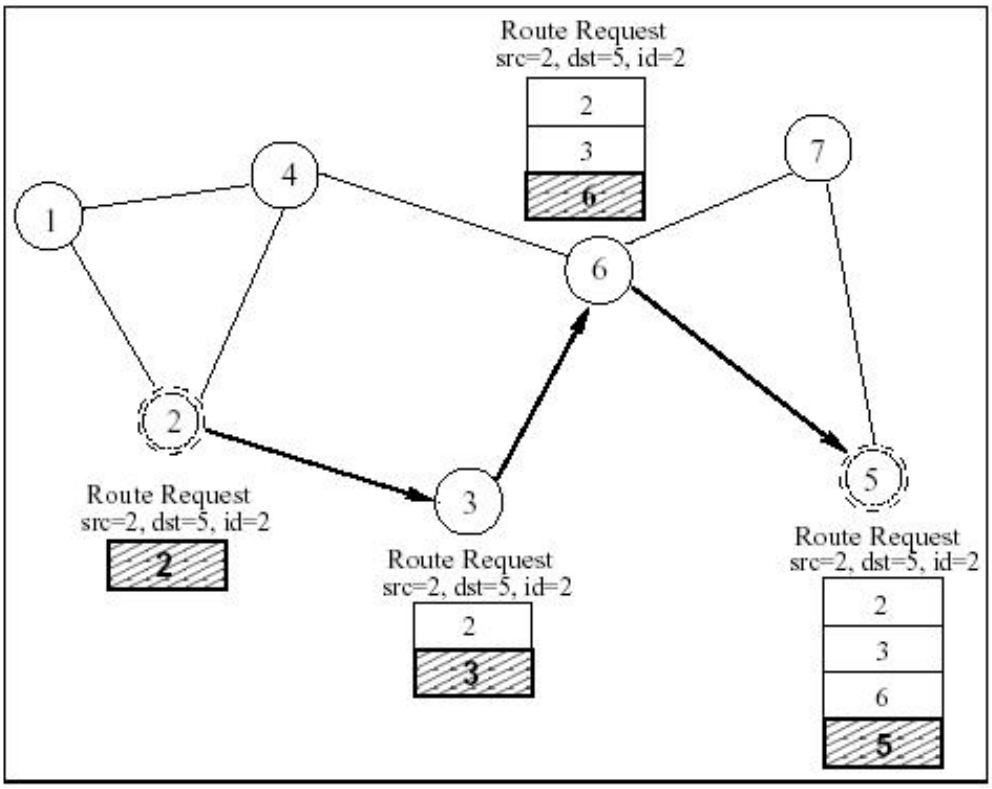


Figure 3: Route Request in DSR

2.2.1.2 Route Maintenance

Route maintenance is the mechanism by which a packet's sender detects if the network configuration has changed so that it can no longer make use of its route to a destination. Each node that transmits a packet is responsible for confirming that the packet has been received by the next hop on the source route. If no confirmation has been received after a set number of attempts, a *route error* message is returned to the original sender of the packet. A *route error* packet indicates that a source route has been broken, and a sender will at this point attempt to use another route to the destination node that is already in its cache, or invoke route discovery to find a new route.

Confirmation can be achieved in a variety of manners. By monitoring failure notifications provided by the data link layer, nodes can determine if the packet was successfully received. If bi-directionality is permitted, confirmation can be achieved when a node overhears the next node transmitting the packet to the next node after it.

Finally, an explicit request for an acknowledgement can be make by embedding an acknowledgment request in the packet header.

2.2.1.3 Optimisation Techniques

Several optimisations to the DSR protocol have been proposed and evaluated to be very effective. The majority of these optimisations rely on DSR's very aggressive use of source routing and route caching. The following paragraphs summarise some of these optimisations.

Salvaging

When a data packet meets a failed link on its source route, the intermediate node at which the link failure exists can use an alternative route from its own *route cache*.

Gratuitous Route Repair

A source node that receives a *route error* packet may piggyback the *route error* packet in its following *route request*. This helps clean up the caches of other nodes in the network that may have the failed link in one of the cached source routes.

Promiscuous Listening

When a node overhears a packet not addressed to itself it checks to see whether or not the packet could be routed through itself so as to shorten the overall route. If this is the case, it sends a gratuitous *route reply* to the source node informing it of the shorter route. Promiscuous listening also allows DSR to "snoop" on any packet it receives and extract information from the packet's header. This allows for a node to learn different routes without being directly involved in the routing process.

2.2.2 Destination-Sequenced Distance Vector (DSDV)

DSDV is a hop-by-hop distance vector routing protocol requiring each node to periodically broadcast routing updates. Loop freedom is guaranteed by DSDV and this is its key advantage over traditional distance vector protocols [Broch].

Each DSDV node maintains a routing table, which lists the next hop for all destinations. All routes are tagged with a sequence number. A route with a higher sequence number is considered more favourable than one with a lower sequence number, and in the case where two routes have the same sequence number, the route with the lower metric is considered more favourable. Each node in the network periodically advertises a monotonically increasing even sequence number for itself. If a node decides that its route to a particular destination node has broken, it advertises the route to that destination with an infinite metric and a sequence number that is one greater than its sequence number for that route (i.e. an odd sequence number). Any node that is routing packets to the destination node, via the node that decided that the route had broken, incorporates the infinite metric route into its routing table, until it discovers a route to the destination node with a higher sequence number.

2.2.3 Temporally-Ordered Routing Algorithm (TORA)

TORA is an on-demand routing protocol based on a "link reversal" algorithm. It is designed to discover multiple routes to a destination quickly and to minimise communication load by localising algorithmic reaction to configuration changes whenever possible. Routing via the shortest path is not considered of primary importance. TORA frequently makes use of longer routes in order to avoid the communication overhead that is required to discover new routes.

Broch et al. [Broch] describe the actions taken by TORA in terms of water flowing downhill through a network of tubes. The tubes represent the links between the nodes in the network, the junctions of tubes represent the nodes, and water in the tubes represents data packets. The routing protocol computes the height of each node with respect to the appropriate destination. If a tube between a sending node and a destination node becomes blocked, the height of the sending node is set to a height that is greater than any of its neighbours. This will cause water to flow back out of the sending node, and towards other nodes that had been routing packets to the destination node, via the sending node in question.

2.2.4 Ad-Hoc On Demand Distance Vector (AODV)

AODV may be considered a combination of both DSR and DSDV. It shares with DSR an on-demand characteristic in that it also discovers routes in a reactive manner, i.e. on an as needed basis. AODV employs a similar route discovery process to DSR, but maintains routing information in a very different manner. Whereas DSR adopts source routing, AODV makes use of hop-by-hop routing and various other mechanisms offered by DSDV.

An important feature of AODV is the maintenance of timer-based states in each node, regarding utilisation of individual routing table entries. If a routing table has not been used recently, it is expired. For each routing table entry, a set of predecessor nodes is maintained. These indicate the set of neighbouring nodes which use that entry to route data packets. When a next hop link is broken, these nodes are notified. Each predecessor node, in turn, notifies its own set of predecessors, and this has the effect of erasing all routes that were making use of the broken link. Perkins et al. [Perkins 1] suggests that route error propagation in AODV can be visualised conceptually as a tree whose route is the node at the point of failure, and all sources using the failed link as the leaves.

2.2.5 Comparison of Routing Protocols

Despite the fact that several comparisons of ad-hoc routing protocols have been conducted, there has yet to be adopted a popular standard.

Broch et al. [Broch] conclude from their performance comparison research that each of the DSDV, TORA, DSR, and AODV protocols, perform well in some cases, yet have certain drawbacks in other cases. Perkins et al. [Perkins 1] observe from their simulations that DSR outperforms AODV in low “stress” situations (e.g. small number of nodes, low amount of mobility), but that AODV outperforms DSR in “high” stress situations. Both of the aforementioned studies conclude however, that DSR consistently generates less routing overload than AODV.

2.3 Security in Ad-Hoc Networks

The security characteristics of ad-hoc networks are very different to those of traditional infrastructure based wireless networks. The absence of a fixed infrastructure implies that nodes cannot rely on information from trusted servers such as certification authorities.

Security in ad-hoc networks is particularly difficult to achieve, principally because of the vulnerability of the links, the erratic nature of connectivity, the limited physical protection of the nodes, the dynamically changing topology, and the absence of a trusted third party such as a certification authority.

When one examines security in relation to ad-hoc networks, one is obliged to consider the topic of routing mechanisms. Routing mechanisms are much more vulnerable in ad-hoc networks than in traditional networks and thus are a major consideration in any analysis of security.

In this section, we shall examine the security challenges facing ad-hoc networks. Given that routing mechanisms comprise one of these principle challenges, we shall then consider the issue of secure routing.

2.3.1 Security Challenges

The nature of ad-hoc networks poses new security challenges that we will now examine.

The use of wireless links renders an ad-hoc network vulnerable to link attacks, ranging from passive eavesdropping to active interference. Eavesdropping may violate confidentiality of information transmitted across the network, whereas active attacks might allow an adversary to delete messages, to modify messages, to generate erroneous messages, to replay old messages, and to impersonate a node. This problem has been thoroughly researched for wireless networks in general (it is not

specific to ad-hoc networks) and many solutions have been proposed and deployed such as frequency hopping spread spectrum communications.

As opposed to traditional wire-line networks where nodes may be physically secured (for example, by being locked into a cabinet), nodes in an ad-hoc network do not have good physical protection. This lack of physical security is further increased in a hostile environment. It is thus necessary to not only consider malicious attacks from the outside, but to also consider the issue of malicious attacks being launched from within the network. Hubaux et al. [Hubaux] suggest that if a device is at risk of being captured or hijacked, the conventional manner in which it is protected is by implementing it in tamper resistant hardware. Cryptographic information (typically a secret key) could be embedded in a smart card that could be plugged into and removed from the node in question (an example of this is the SIM card based solution for GSM). Routing mechanisms embedded in a node could be protected in a similar manner by storing the related software on a smart card.

One of the fundamental security problems in ad-hoc networks is that the presence of a trusted third party, such as a certification authority, cannot be assumed. Many of the security services offered by a public key infrastructure in the domain of traditional infrastructure based networks are reliant on the presence of a trusted third party. The use of digital certificates, public key cryptography, digital signatures and other traditional security mechanisms can no longer be taken for granted in the ad-hoc domain. In such an environment, the lack of trusted third parties makes it necessary to establish trust in alternative ways.

Due to the fact that each node in an ad-hoc network acts as a relay, the routing mechanisms used in such networks are more vulnerable than their conventional counterparts. An adversary who hijacks an ad-hoc node could paralyse the entire network by disseminating false routing information. Node selfishness is a more subtle form of malicious behaviour where nodes do not relay packets (e.g. so as to conserve their own battery power). Another problem that weaknesses in a routing protocol might present is that of malicious neighbour discovery. Researchers have recently shown how this kind of attack can be performed against a Bluetooth device [Hubaux]. We will now examine in greater detail the issue of secure routing

2.3.2 Secure Routing

A routing protocol requires two properties so as to achieve availability in the network: robustness against a dynamically changing topology, and an ability to cope with malicious attacks. Several routing protocols (see section 2.2) cope well with a dynamically changing topology. However, little research to date has focused on the area of developing or extending a routing protocol so as to include mechanisms that defend against malicious attacks.

Zhou and Haas [Zhou] suggest that there are two sources of threats to routing protocols. The first comes from external attackers who could successfully partition a network or introduce an excessive traffic load into the network by causing retransmissions and inefficient routing. To defend against such a threat, nodes can protect routing information through the use of cryptographic schemes such as digital signatures. However, this defence is ineffective against the second and more severe potential threat that the routing protocol is faced with: compromised nodes. Compromised nodes have the ability to advertise incorrect routing information to other nodes. The detection of this type of incorrect routing information is very difficult due to the dynamic topology of ad-hoc networks: when a piece of routing information is found to be incorrect, it may be incorrect because it was generated by a compromised node, or it may be incorrect as a result of topology changes over time.

Having considered the various threats to routing protocols in ad-hoc networks, Zhou and Haas [Zhou] have proposed a security mechanism that could be used for the benefit of a single organisation. This mechanism ensures confidentiality, integrity, and availability in the network by making use of threshold cryptography and secret sharing. It is assumed that a trusted certificate authority is present in the network, whose public key is known by all nodes in the network. We do not believe these assumptions are realistic in the case of a spontaneously generated ad-hoc network that may span organisational boundaries.

Initial work on mitigating routing misbehaviour in ad-hoc networks is proposed in [Marti]. In this paper, the case in which some nodes agree to forward packets but fail to do so is considered. A node may misbehave in this manner because it is

overloaded, selfish, malfunctioning, or malicious. An overloaded node lacks the CPU cycles, buffer space, or network bandwidth to forward packets. A selfish node is unwilling to use CPU cycles, battery power, or network bandwidth to forward packets. A malfunctioning node may have a software fault that prevents it from forwarding packets. A malicious node may launch a denial of service attack by simply dropping packets intentionally. Two mechanisms are proposed in order to deal with this problem: a watchdog, which is responsible for identifying the misbehaving nodes, and a pathrater, which is responsible for determining the best routes that circumnavigate these nodes. The paper shows that these two mechanisms allow for the total throughput of the network to be maintained at an acceptable level, even in the presence of a high amount of misbehaving nodes (e.g. 40%). In addition, all simulation scenarios assume no *a priori* trust relationships. However, several problems still exist and it is recognised by the authors that this paper simply presents "initial" work in this previously un-researched area.

Buchegger and Le Boudec [Buchegger] suggest that, despite the fact that ad-hoc networks only function properly if the participating nodes cooperate in routing and forwarding, it may be advantageous for individual nodes not to cooperate. They propose a protocol, called CONFIDANT, which is designed to detect and isolate misbehaving nodes, thus making it unattractive to deny cooperation: it is based on selective altruism and utilitarianism. A performance analysis of the Dynamic Source Routing (DSR) protocol fortified by CONFIDANT is presented and compared to the regular defenceless DSR. They show that a network with CONFIDANT and up to 60% of misbehaving nodes behaves almost as well as a benign network, in sharp contrast to a defenceless network. However, it must be noted that the CONFIDANT protocol assumes that nodes are authenticated and that no node can pretend to be another in order to get rid of a bad reputation. We do not believe these assumptions to be realistic in the case of a spontaneously generated ad-hoc network.

Major differences exist between the approach adopted by Buchegger and Le Boudec and that adopted by Marti et al. The approach adopted by Marti et al. does not punish malicious nodes that do not cooperate, but rather relieves them of the burden of routing data for other nodes, and furthermore, continues to forward their messages without complaint. This approach rewards malicious nodes. In contrast, the

CONFIDANT protocol seeks to achieve the opposite. However, whereas the Marti et al. approach assumes no *a priori* trust relationships, CONFIDANT assumes that nodes are authenticated and cannot carry out identity swapping.

2.4 Trust

Trust is a complex subject relating to belief in the honesty, truthfulness, competence, reliability, etc. of an entity. The main problem with trust is that there are many different views of what exactly constitutes trust, and there is no consensus in the literature with regard to what trust is.

We have previously seen that traditional security systems make no attempt to explain why a user is trusted to perform a certain action. This section will demonstrate that trust can be used in a framework that will allow a computer system to both quantify trust and to justify the parameters and contexts on which it has built its trust assumptions about a user.

2.4.1 Defining Trust

Academics in fields ranging from economics to social psychology have studied the notion of trust. Many different views on trust exist, and the multiple definitions of trust presented in this section will illustrate this.

Kini and Choobineh [Kini] examine trust from the perspective of economists, sociologists, personality theorists and social psychologists. They state that trust, as defined in the Webster dictionary, is:

- An assumed reliance on some person or thing. A confident dependence on the character, ability, strength, or truth of someone or something.
- A charge or duty imposed in faith or confidence or as a condition of a relationship.
- To place confidence (in an entity).

Kini and Choobineh create their own definition of trust in a system: "a belief that is influenced by the individuals opinion about certain critical system features". It is evident from these definitions that trust is subjective in nature. Two different entities may view the same system with different levels of trust.

Another definition of trust is taken from Diego Gambetta (1990). "Trust is a particular level of the subjective probability with which an agent will perform a particular action, both before [we] can monitor such an action and in a context in which it affects [our] own action". This definition once again highlights the subjective nature of trust. It also implies that in a situation where all the relevant information is not available, the quality and amount of information that we can measure will have an effect on our level of trust.

Trust has been proposed as a mechanism for reducing the level of risk in a situation. This is possible because of the relationship between trust and risk. In the context of an e-commerce transaction, the level of trust has an approximate inverse relationship to the degree of risk [Grandison].

2.4.2 Properties Of Trust

Trust relationships are not absolute. A trustor will never trust a trustee to carry out any possible action that it may choose. Trust is always associated with a particular action or context. For example, a person may only be trusted to carry out financial transactions dealing with less than €10,000. It is also possible that a trust relationship is not symmetric, i.e. A's trust in B may not be the same as B's trust in A.

There is debate in the literature with regard to whether or not trust relationships should be able to exhibit transitivity. In general, trust is a non-transitive property, but may demonstrate transitivity when one considers the use of recommendations or delegation. The following statement demonstrates this point. "If Ann trusts Bob, and Bob trusts Cathy, then Ann trusts Cathy". This statement is in general not true, but may be considered true if the following conditions are true:

- Bob explicitly tells Ann that he trusts Cathy (a recommendation).
- Ann trusts Bob as a recommender. If Ann does not trust Bob as a recommender then she should ignore information Bob supplies her with.
- Ann must be allowed to make a judgement on the quality of Bob's recommendation.

2.4.3 Trust Values

Very often, there is a *level* of trust associated with a relationship. Some entities may be trusted more to perform an action than others. Although trust has no measurable units, its value can be measured. Trust can be considered as a commodity like knowledge or information. However, the major problem with using explicit values to represent trust is that due to the subjective nature of trust, different entities may associate different levels of trust with the same service.

Whether the trust value associated with an entity should be discrete or continuous is not clear. If discrete values are used, then a trust rating of high, medium, or low may suffice. Some systems support arithmetic operations on trust recommendations, and in this case numeric quantification such as a value between 0 and 1 may be more suitable. However, there is still uncertainty relating to the representation of ignorance (or the unknown) with respect to trust. Several opinion models (e.g. Jøsang's Opinion Model [Jøsang]) have been developed to assign trust values in the presence of uncertainty. However, major weaknesses have been identified with such models and hence their use is limited.

2.4.4 Trust Formation

When no prior knowledge about a node exists, the notion of trust formation comes into play. There are three ways in which trust can be established in an unknown

entity: personal experience, recommendations, and reputation. These methods of forming trust are listed in order of decreasing reliability.

Personal experience in a node is gained when that node is trusted, and this trust is subsequently either honoured or betrayed. When building personal experience, it is important that the potential risk of placing trust in an unknown entity is considered. Personal experience is the most reliable manner in which trust can be established, simply because it is based on first hand experience with a particular node.

A recommendation is an assessment by a node, of another node's trustworthiness. Recommendations accelerate the trust formation process. A particular node may be introduced to other nodes, which trust a recommendation about that particular node. Any node may issue a recommendation, and it is necessary for all nodes that receive this recommendation to make an independent decision about whether or not they trust it. Similarities are present between this mechanism and the "web of trust" mechanism in PGP. There is no requirement for a public key infrastructure to make use of recommendations.

A node's reputation is an amalgamation of the level of trust that all other nodes place in that node. Reputational information need not solely be the opinion of others (i.e. a combination of recommendations from others). Also included is information based on an individual node's own personal experience.

2.5 Trust Management Systems

The Trust Management approach of distributed systems security was developed because of the perceived inadequacy of traditional authorisation mechanisms of distributed systems. Blaze et al. [Blaze 1] define trust management as "a unified approach to specifying and interpreting security policies, credentials, and relationships that allow direct authorisation of security-critical actions".

Most trust management systems focus on protocols for establishing trust in a particular context. Traditionally, all applications implemented their own security mechanisms for specifying access control, checking compliance, and binding user authentication to security-critical operations. This approach was subject to the risk of application developers making mistakes that might compromise the security of the system in question. The developers of trust management systems recognised the risks associated with this approach. The principal motivation in the development of trust management systems is to produce off the shelf security modules that can be integrated into any application. The trust management module can be configured to allow for application specific security policies and credentials.

Trust management systems define languages for expressing authorisation and access control policies, and provide a trust management engine for determining when a particular request is authorised. Many different types of trust management systems have been developed, some focusing on authentication, others for specialised purposes, others for general-purpose authorisation, and others based on logics. For the remainder of this section will shall examine two of the most prominent trust management systems in existence.

2.5.1 KeyNote

KeyNote, the successor to the PolicyMaker trust management system, was developed by AT&T Research Laboratories to improve on the weaknesses of PolicyMaker. PolicyMaker was the first tool for processing signed requests that embodied trust management principles. Input to PolicyMaker included a set of local policy statements, a collection of credentials, and a string describing a proposed trusted action. The output consisted of a yes/no result or additional restrictions that would allow the proposed action possible [Blaze 1]. KeyNote maintains the design principles of assertions and queries from PolicyMaker, but two additional design goals are included: standardisation and ease of integration. KeyNote assigns more responsibility to the trust management engine, making it easier to integrate into applications [Grandison].

KeyNote provides a single, unified language for both local policies and credentials. These policies and credentials, known as 'assertions', contain predicates that describe the trusted actions permitted by the holders of specific public keys. Assertions are simply small, highly structured programs. A signed assertion, which can be sent over an un-trusted network, is called a 'credential assertion'. Credential assertions, which also serve the role of certificates, have the same syntax as policy assertions but are also signed by the principle delegating the trust. KeyNote, like other trust management engines, does not enforce policy directly, but only provides advice to applications that call it [Blaze 1].

2.5.2 REFEREE

REFEREE (Rule-Controlled Environment For Evaluation of Rules and Everything Else) is a trust management system for making access decisions relating to web documents. It provides both a general policy-evaluation mechanism for web clients and servers, and a language for specifying trust policies.

REFEREE, like PolicyMaker and KeyNote, is a recommendation based query engine, and so it needs to be integrated into a host application. It evaluates requests and returns a value and a statement list. The value is either a true, false, or unknown. True means that the action in question may be taken because sufficient credentials exist for the action to be approved.

All trust decisions are placed under explicit policy control in REFEREE: every action, including evaluation of compliance with policy, happens under the control of some policy. That is, REFEREE is a system for writing policies about policies, as well as policies about cryptographic keys, PICS label bureaus, certification authorities, trust delegation, or anything else [Chu].

2.6 Summary

In this chapter we have examined the operation of some traditional security mechanisms that have a universally recognised importance. We have seen that the cryptographic protocols and authentication services provided by a public key infrastructure do allow for a certain level of trust to be built into communications.

However, the traditional security mechanisms outlined were designed primarily with infrastructure-based networks in mind. In addition to this it is evident that the services provided by a public key infrastructure are dependant on the use of trusted third parties and known identities. In an ad-hoc network, the presence of trusted third parties can no longer be assumed. Hence, the assumption that an underlying public key infrastructure can provide adequate security is not relevant in the domain of ad-hoc networking. Indeed, the notion that the routers in the network are owned and managed by trusted operators is also invalid in an ad-hoc network.

We have examined the principle security threats facing mobile ad-hoc networks and seen that routing protocols in such networks are more vulnerable than those in traditional networks. Much research in the area of mobile ad-hoc networks has concentrated on routing issues, but security on the other hand, has been given a lower priority. No routing mechanisms have to date been developed that successfully implement defences against the various types of potential malicious attacks.

Marti et al. [Marti] have investigated the possibility of identifying misbehaving nodes in an ad-hoc network without *a priori* trust relationships. Despite the fact that some success was achieved in this study, the authors identify some outstanding problems and recognise that their work simply represents 'initial' research into this previously un-researched area. Buchegger and Le Boudec [Buchegger] attempt to make misbehaviour unattractive with the implementation of the CONFIDANT protocol. A performance analysis illustrated the success of the CONFIDANT protocol. However, as with the majority of studies in this area, CONFIDANT assumes that nodes may be authenticated and that identity swapping cannot take place.

Chapter 3

Design

The aim of this project is to design and implement a trust based route selection system for ad-hoc networks that will allow for the identification of misbehaving nodes and the improvement of the throughput in the network in the presence of misbehaving nodes. The design of this system involves the extension of the Dynamic Source Routing (DSR) protocol so as to incorporate the notion of trust. This extension should allow for the resulting trust based route selection system to achieve the aims outlined above.

Section 3.1 will introduce the concept of trust based route selection. The key processes involved in this system will be identified. Section 3.2 will describe and discuss each of the three major components in the trust based route selection system. Finally, section 3.3 summarises the ideas presented in this chapter.

3.1 Overview of Trust Based Route Selection

The process of route selection in DSR involves the selecting of the shortest route to a destination node. In a trust based route selection system implemented in an ad-hoc network in which nodes may be misbehaving, the most reliable and secure path to a destination node is chosen. In such a system, this selection process is based on a local evaluation of the trustworthiness of all nodes on all known source routes to the destination. Hence, it is necessary for all nodes to continuously assess the trustworthiness of other nodes in the network. The *route selection* module is responsible for the implementation of the route selection strategy.

Because the notion of *a priori* trust relationships in ad hoc networks cannot be assumed, a trust based route selection system must have a mechanism for establishing new trust relationships with unknown nodes. The dynamic nature of an ad-hoc network's topology also implies that known nodes will leave the network, that routes may become invalid over time, and that nodes may become overloaded, compromised, selfish, or malicious and may begin to misbehave. It is thus clear that a continuous trust evaluation of all nodes in the network must be carried out so as to reflect the dynamic nature of trust. The reliability of nodes to forward packets will vary over time, and this is reflected by a variable trust value that is associated with each known node in the network. It is the responsibility of the *trust formation* module to implement trust establishment and updating of trust values.

The *trust management* module maintains trust information about all known nodes. It is because of the potential for a large number of nodes to be present that a separate module is required to maintain trust data. The design architecture of the trust based route selection system is shown in figure 4. The extensions to DSR that are required to incorporate the notion of trust have been added as a layer between DSR and the application using the routing protocol.

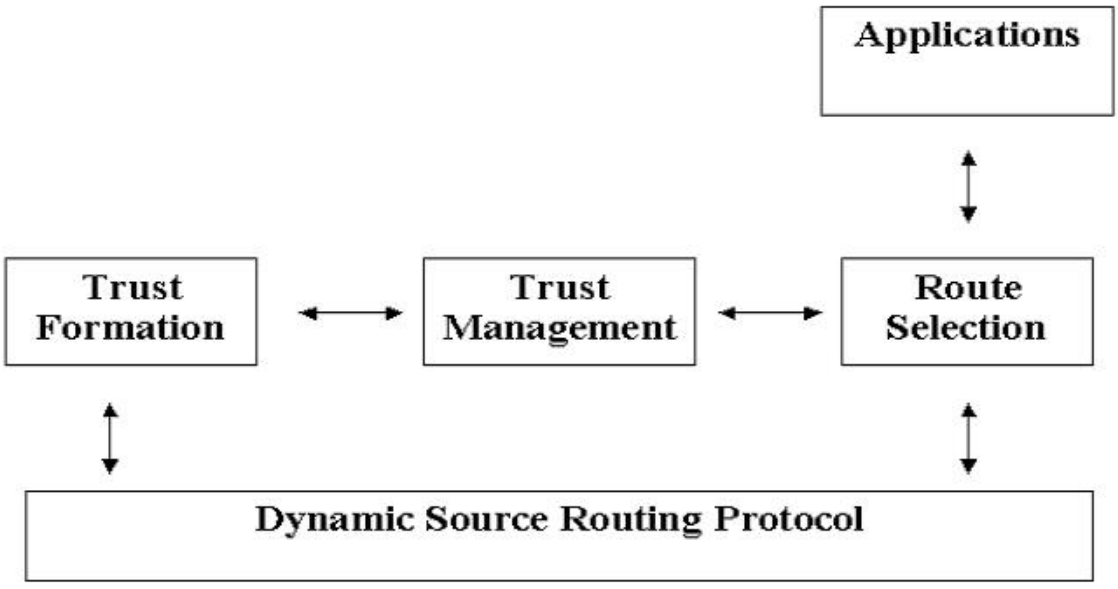


Figure 4: Trust Based DSR Architecture

3.2 Design of Trust Based Route Selection

As were identified in the previous section, the three processes that are involved in trust based route selection are trust formation, trust management, and route selection. In this section each of these three processes will be examined in detail.

3.2.1 Trust Formation

The function of the trust formation process is to establish trust relationships with unknown nodes and to continuously update trust values associated with nodes. These two functions will now be considered in detail.

3.2.1.1 Trust Establishment

In order to establish a trust relationship with an unknown node, it is first necessary to encounter such a node. This can happen in a variety of ways. The principle manner in which this will occur is during the route discovery process of DSR. Following the route discovery process, a node may receive one or more route reply packets. The trust formation module monitors all incoming packets. When a route reply packet is received the source route contained in it is examined to see if there are any unknown nodes in it. This is achieved by the querying of the trust management module so as to determine if a data entry exists for each node in the source route. If no data entry for a particular node exists, the trust management module is informed to create a new data entry for the newly encountered node.

DSR can make use of promiscuous listening that allows the "snooping" of any packet it receives and the extraction of information from the packet's header. This technique may be used to extract source routes from packet headers. Once a source route has been extracted it may be examined as outlined above. This process will allow for further encountering of unknown nodes and further establishment of trust relationships.

3.2.1.2 Updating of Trust Information

Once a trust relationship has been established with a node, it is necessary to update the level of trust associated with that node on a continuous basis. In order to update this trust value, the behaviour of the node must be observed, i.e. personal experience with this node must be obtained. Initially, this involves taking a risk by placing trust in an unknown node. Depending on whether this node subsequently honours or betrays this trust, its trust value will be incremented or decremented accordingly.

To determine if a packet reaches its destination successfully, it is necessary to attach an "acknowledgment request" option to the packet header. If a node receives such a packet for which it is the destination, it is obliged to confirm receipt of the packet by returning an "acknowledgment" to the source node. If the source node receives this acknowledgment, this confirms that none of the nodes on the source route are dropping packets and it increases the trust value of all these nodes.

Conversely, if the source node does not receive this acknowledgment, this implies that the source route has failed. However, the source node is unable to determine where on the source route the failure occurred, i.e. which node on the route is dropping packets. To be able to determine this, it would be necessary to ask every node on the route for an acknowledgment. We do not believe this to be practical because of the increased overhead that would result in the network. Indeed, despite the fact that requesting an acknowledgment for all packets sent is a desirable option, this is also deemed impractical because of the large increase in network traffic that would result. DSR, as with most ad-hoc routing protocols is designed with the intention of minimising the traffic overhead that is needed to operate the protocol.

The only feasible solution to this problem is to decrease the trust value of all nodes on the source route for which the failure occurred. This appears to be somewhat unfair because reliable and trustworthy nodes on the route will get their trust values decremented. However, over time this approach will lead to the identification of misbehaving nodes and behaving nodes. This is partly due to the fact that misbehaving nodes have a high probability of being located at the intersection of source routes on which routing failures occur.

Because the trust formation module monitors all incoming packets to a node, it can also handle the receipt of a route error packet. This type of packet indicates that a link error has occurred on the source route, and at which node this link error has occurred. However, this mechanism operates on the assumption that all nodes act in good faith. It is possible for a malicious node to drop packets on a link, and to subsequently generate and send a route error packet to the source node and claim that the link error occurred at the next node in the link. This presents a problem to the source node which must now make a decision regarding the validity of the route error packet. One solution to this problem is to make this decision by referring to the trust value of the node originating the route error packet. If this node has a high trust value, this route error packet could be trusted. It should be noted that this solution has not been incorporated into the implementation presented in this dissertation.

Another potential reason for the updating of trust values is based on the exchange of recommendations between nodes. If a node issues a recommendation to another node regarding the trustworthiness of another node, the recipient node must decide whether or not to make use of this recommendation. This decision is based on the trustworthiness associated with the issuing node. The use of recommendations could cause a large increase in network traffic. In addition, the use of reputation to establish trust is deemed less reliable than the use personal experience. The implementation presented in this dissertation does not make use of recommendations for the updating of trust values.

3.2.2 Trust Management

It is the responsibility of the trust management module to store trust information relating to all known nodes in the network. Trust management does not directly interact with the DSR protocol, but does interact directly with both the trust formation and route selection modules.

Trust management establishes trust information relating to encountered nodes and updates this information based on instructions from the trust formation process. Trust management also provides the route selection process with information when queried.

Route selection requires this information to determine the most reliable and trustworthy route available to a destination node.

The following information about each known node is recorded by the trust management module: IP address, trust value, time at which relationship was established, and minimum trust value of this node.

The IP address is used to identify the node in subsequent encounters. However, because of the assumption that no trusted third party is present in the network, the problem of identity swapping exists. A node may misbehave, disappear, and then reappear with a new identity, thus negating any bad reputation it has received. To solve this problem, the time when the trust relationship was initially established with the node could be incorporated: the older the trust relationship with the node, the more trustworthy the node should be considered. Alternatively, the times at which misbehaving nodes disappear and new nodes appear could be compared with the intention of identifying identity swapping nodes. If a misbehaving node disappeared at a certain time and a new node appeared subsequent to this in the same geographical area, a certain level of suspicion could be associated with this node. This suspicion could later be cleared depending on the future behaviour of the node.

The trust value is stored as a decimal in the range [0, 1]. 1 represents absolute trust, whereas 0 represents absolute distrust. We assume no prior knowledge about the surrounding nodes, so initially all nodes are assigned a trust value of 0.5. The rate at which a node's trust value is incremented or decremented is exponential. This assures that if a node has a low trust value, positive factors may increase its trust value quickly, whereas if the node has a high trust value, these effects will be less pronounced. Conversely, if a node has a high trust value, negative factors may decrease its trust value quickly. This exponential approach is used to mitigate against nodes that initially behave well so as to gain a high trust value, and then attempt to take advantage of that high trust value to disrupt service in the network.

The storing of the minimum trust value that a node has ever had may be used by the route selection process to implement a security policy, e.g. it may choose to ignore all routes where any node has ever had a trust value of less than 0.3.

The amount of trust information relating to a node that should be maintained by the trust management module is subject to discussion. If only basic trust information is stored, this will reduce the necessary storage requirements. However, this will also limit the potential range of selection heuristics that may be implemented. Johnson et al. [Johnson 2] observe that the limiting factor in ad-hoc networks is not CPU time or local storage requirements, but is network speed. Based on this observation, it seems reasonable to store a large amount of trust information about each node, which would allow for more sophisticated route selection techniques.

It may be desirable to store information about the recent behaviour of a node. If a node has a high trust value, but all of the most recent experiences with this node have been negative, the high trust value (despite the exponential rate of decrease of the trust value) may mask this recent poor behaviour. By storing information about the recent behaviour, a node will have the option of avoiding nodes that have been misbehaving recently, despite the fact that their trust value is reasonably high.

If recommendations are being used in the system, it would be beneficial to store this recommendation information separately from that of observed events, i.e. a node may have a trust value of 0.66 based on personal experience, but a trust value of 0.72 based on recommendations from other nodes.

Because only a finite and limited amount of data can be stored by the trust management module, it is necessary to delete trust information once it has become redundant, i.e. once it has outlived its usefulness. Nodes may eventually stop interacting with one another and nodes may leave the ad-hoc network at any time. To reflect this dynamic relationship between nodes, the trust value of a node that is not being encountered should erode over time until a point is reached where it is assumed that the node has actually left the network. This approach has two advantages. If a node has not been encountered for a long period of time, it will be considered less trustworthy. In addition to this, a node that has migrated from the network will no longer be included in any source routes.

In the implementation of the trust management system, no permanent copy of trust information is being stored between sessions, i.e. when the user shuts down or leaves the network, and subsequently reboots or rejoins the network. However, making use of such permanent storage of trust information would reflect the fact that users of ad-hoc networks may encounter the same nodes on a daily basis. This would also allow for the trust based dynamic source routing system to identify and use reliable and trustworthy routes almost immediately, as opposed to the situation where the system has to spend a certain period of time achieving personal experience with nodes in order to establish and update trust relationships.

3.2.3 Route Selection

When a source node is about to send data to a destination node, it is the responsibility of the route selection module to select the most reliable and trustworthy route to use. The DSR route caches are searched for possible source routes to the destination node. If only one route exists, then the sending node has no choice but to use this route. However, if multiple routes exist, the sending node may select which route to use according to some route selection strategy. The trust management module is queried so as to determine the trust value of all nodes on all routes to the destination node. This is required so as to determine the trustworthiness of all routes.

There are a wide variety of route selection strategies that may be used. The simplest strategy, which has been implemented, is to select the route with the highest average trust value. Because it is generally desirable to minimise the number of nodes in a route, before the final selection is made, the average trust value of all routes is divided by the number of nodes on the route, thus favouring shorter routes. Another possible selection strategy is to avoid un-trusted nodes, preferring to use unknown nodes over untrustworthy nodes. A third possible strategy is to prefer routes through known nodes, regardless of their trustworthiness, in order to increase the predictability of the routing.

The route selection module may also implement a security policy. A simple security policy that may be used is to exclude all routes which have a node with a trust value

of less than 0.3. More complex security policies may be used depending on the amount of trust information being stored about the nodes. For example, any route with a node for which a suspicion of identity swapping exists may be excluded, or any route with a node for which a recommendation has not been received may be excluded.

3.3 Summary

In this chapter we identified and analysed the design architecture of the trust based route selection system. An in-depth analysis of the three major processes involved in trust based route selection was carried out. Finally, some potential problems and some potential optimisations were discussed.

Chapter 4

Implementation

The design and operation of the trust based route selection system was described in the previous chapter. This chapter describes the implementation of this design. Section 4.1 offers both an overview and a detailed discussion of the NS-2 network simulator. NS-2 provides an ideal simulation environment because of its existing implementation of the DSR protocol. Sections 4.2, 4.3, and 4.4 describe the implementation of the extensions to the DSR protocol that are required to implement the trust based route selection system outlined in chapter 3. Section 4.5 concludes the implementation chapter.

4.1 The NS-2 Network Simulator

4.1.1 Overview

The extensions to the DSR protocol that were described in chapter 3 were implemented as extensions to the NS-2 network simulator. The NS-2 network simulator is the latest version of NS, a packet-level, discrete event simulator targeted at networking research. NS-2 provides substantial support for both wired and wireless (local and satellite) networks and is used mainly by the research and academic community for validation purposes.

NS-2 supports networks from the data link layer to the application layer (layer 2 to 7 in the OSI protocol stack). It provides the framework and the implementation of both

unicast and multicast routing protocols. Implementations exist for the transmission control protocol (TCP), the unix datagram protocol (UDP), and constant bit rate (CBR) transmissions.

NS-2 is available for a multitude of operating systems including Linux, Solaris, SunOS, FreeBSD, and Windows 95/98/NT.

4.1.2 Architecture

NS-2 is written in C++ and OTcl (Object Tcl). An object oriented approach allows for maximum scalability and extensibility. The choice of two languages offers a modular design where a separation exists between the low level data manipulation and control operations on the state of the simulation. C++ is used to implement low level algorithms and protocols. This allows for maximum performance. OTcl, a scripting language, is used to define the simulation scenario – network configuration, traffic pattern, movement etc. By changing the OTcl script being used it is possible to change the simulation being run without having to recompile the program.

NS-2 is made up of many components, most of which are implemented as a C++/OTcl class. Agent objects are at the heart of NS-2. They represent endpoints in the network where layer packets are constructed or received. Examples of agent classes implemented in NS-2 include UDP, variations of TCP, and ad-hoc network routing protocols such as DSDV and DSR. A DSRAgent object encapsulates the functionality of the DSR protocol. It acts as the interface between each node and the network. It is thus to the DSRAgent class that the majority of the trust based extensions have been integrated.

Output from NS-2 simulations is written to special objects called traces. Traces store information about each packet that is sent, received, or dropped during run-time. This information is typically written to an output file and post-processed after the simulation has been executed fully.

4.1.3 The CMU Monarch Extensions

Many modifications and additions to the NS network simulator have been made by the CMU Monarch Project. The majority of the CMU extensions made to NS were made in order to support ad-hoc networking. NS did not support ad-hoc routing protocols such as DSR before the advent of the CMU Monarch Project.

The CMU extensions to NS were in the physical, data link, and routing layers. The components implemented in these layers were as follows.

- *Physical Layer:* A radio propagation model, omni-directional antennas and a shared media network interface that makes use of direct sequence spread spectrum (DSSS).
- *Data Link Layer:* The IEEE 802.11 distributed coordination function (DCF).
- *Routing Layer:* The DSR, DSDV, TORA, and AODV routing protocols were implemented.

Another important contribution by the CMU Monarch Project was a set of scripts that allow for the generation of both movement and traffic patterns. The scripts offer either TCP or CBR traffic on the network. The movement patterns are generated using a random waypoint model. Nodes move in a straight line towards some randomly generated destination, at a speed between 0 m/s and a maximum value defined by the user. Between each destination the mobile nodes stop for a certain period of time referred to as the pause time. The smaller the pause time, the more dynamic the network will be. The pause time may be set by the user.

4.2 Trust Formation

As stated in the design chapter, the trust formation module is responsible for the establishing of trust relationships and for the continuous updating of trust information. The implementation of both these processes will be discussed in this section.

The trust formation process must monitor all incoming packets in order to carry out its two functions. Because the DSRAgent class represents the interface between the node and the network, the trust formation module is created as part of a DSRAgent object and is integrated heavily with the DSRAgent class.

4.2.1 Trust Establishment

When a node receives a packet for which it is the intended destination, the handlePacketReceipt function in the DSRAgent class is invoked. This function examines the packet header, and, depending on the type of packet it is, will invoke a particular function to handle the packet. If the packet is a *route reply*, the acceptRouteReply function is invoked. This function will add the source route contained in the packet to the route cache for future use. To determine if any of the nodes on this source route are unknown to the trust management module, the processNewPath function is invoked. This function queries the trust management module to determine if trust information about each of the nodes is already being stored. This is achieved by invoking the isRegistered function in the trust management class. If a node does not have an existing trust binding, the registerNode function in the trust management class is subsequently invoked to add a new entry to the trust management's data structure.

4.2.2 Updating of Trust Information

In order to continuously update trust information, it is necessary to continuously observe the behaviour of the nodes in the network. This is achieved by periodically requesting destination nodes to acknowledge receipt of packets. The existing DSR implementation does not allow for the request of such acknowledgements. It was thus necessary to change the format of DSR packets so as to be able to set a flag which indicates the desire for an acknowledgment from the destination node.

Packets in DSR are implemented as a SRPacket, which is a C++ structure. The header of these packets is implemented as a hdr_sr, which is also a C++ structure. In

order to periodically request an acknowledgment, it is only necessary to modify the format of the packet header. To modify the packet header to include the *acknowledgement* and *acknowledgment request* options, an approach similar to that which was use to implement the *route request* and *route reply* options was used. A value of '1' for the route_request_ field in the hdr_sr structure is used to indicate that the packet is a *route request*. A value of '0' indicates that the packet is not a *route request*. When a *route request* is made a sequence number is attached to it so that subsequent *route replies* can be matched to it. This sequence number is implemented as an integer field in the hdr_sr structure.

Several fields were added to the hdr_sr structure to allow for the request of acknowledgments from destination nodes. A value of '1' for the ack_request_ field indicates that the destination node of this packet is obliged to acknowledge receipt of the packet. A value of '0' indicates that no *acknowledgment* is necessary. Similarly, a value of '1' in the ack_ field indicates that the packet is an *acknowledgment* and a value of '0' indicates that it is not. *Acknowledgments* received are matched against their corresponding *acknowledgment requests* by the use of an *acknowledgment request sequence* number. This is represented as a double field in the hdr_sr structure.

As was previously mentioned, the handlePacketReceipt function in the DSRAgent class is responsible for dealing with each incoming packet for which the node is the destination. This function was modified so that any *acknowledgment requests* and any *acknowledgements* may be caught and processed.

On a periodic basis, an *acknowledgment request* is piggybacked onto a packet. The trust formation process maintains a record of all *acknowledgment request* packets sent so as to be able to subsequently account for any *acknowledgments* received. If a response is not received within a certain time limit, the source route on which the request was sent is deemed to have failed. The trust formation process will then instruct the trust management process to decrement the trust value of all nodes on the source route. Conversely, if an *acknowledgment* is received within the timeout period, the trust formation process instructs the trust management process to increment the trust value of all nodes on the source route.

If an *acknowledgment request* packet is received, a new DSRAgent function called returnAckToRequestor is invoked. This function constructs an *acknowledgment* packet with a randomly generated sequence number and places this packet with the scheduler for transmission.

The percentage of packets sent for which an *acknowledgment* is requested may be varied. Too many *acknowledgment requests* are wasteful of bandwidth. Too few *acknowledgment requests* will not allow enough trust information to be gathered. The optimal number of such requests that should be sent is subject to debate, but may be approximated by experiment.

4.3 Trust Management

The trust management module is responsible for storing trust information which has been gathered by the trust formation module. It is also responsible for supplying the route selection module with this trust information when requested.

Trust information for each known node is stored in a C++ structure which is defined as follows:

```
typedef struct trust_values
{
        double val;                 //Trust value
        unsigned long addr;         //IP address of node
        Time time;                  //Time when node was encountered
        double lowestVal;           //Lowest trust value ever

}trust_values;
```

As was discussed in chapter 3, the potential to store a higher quantity of trust information exists. However, only the basic information required is being made use of in this implementation. All trust information structures are stored in a dynamically sized array.

Several functions exist in the trust management class and will now be discussed. The function isRegistered allows the trust formation module to determine whether or not

the trust management module contains a data entry for a particular node. If no such data entry exists, the function registerNode will create such an entry. Trust management provides two functions which allow for trust values to be updated. The function increaseTrustValue increments the trust value of a node in an exponential manner. The following equation represents this exponential increase:

```
new_trust_value = old_trust_value + ( (1-old_trust_value) /20 );
```

Similarly, the function decreaseTrustValue decrements the trust value of a node in an exponential manner. The following equation represents this exponential decrease:

```
new_trust_value = old_trust_value - ( old_trust_value/20 );
```

The nature of these equations ensures that a node that initially behaves so as to gain a good trust value, and then subsequently misbehaves, will not maintain a good trust rating for a long period of time. However, these equations may be varied to implement a variety of alternative strategies. For example, the trust value of a node may be decreased by a lesser amount if this node has been known for a long period of time, i.e. the time at which the node was encountered may be incorporated into the equation.

Finally, the getEntryValue function which is invoked by the route selection module is responsible for returning the trust value of a particular node. If the route selection module is to make use of more complex selection algorithms, it is possible for the trust management module to supply it with further information when queried.

4.4 Route Selection

When a DSRAgent object wishes to send a packet, it queries the DSR route cache to determine if a source route to the intended destination node is known. The Mobicache class implements the route cache used in DSR. It is by invoking the findRoute function in the Mobicache class that the DSRAgent object queries the route

cache. If the findRoute function does not return any source routes to the DSRAgent object, the route discovery process is initiated. If only a single source route is returned, the DSRAgent object has no choice but to use this route. If multiple source routes are returned, the DSRAgent object will make use of functions implemented in the route selection class so as to determine the most reliable and trustworthy route to the destination node.

If the findRoute function determines that multiple routes to the destination node are known, it will invoke a new function called findRoutes. The findRoutes function will determine the most trustworthy route in both the primary and secondary route caches. The primary cache contains routes that were discovered following the broadcast of a *route request* packet. The secondary route cache stores source routes that were discovered by "snooping" on packets that were being forwarded on to another node. If the best route found is from the secondary cache, it will then get promoted to the primary cache.

In order for the findRoutes function to determine the most trustworthy route available, it invokes the getRouteTrustValue in the route selection class. The getRouteTrustValue function implements the route selection strategy that is being used. In this particular implementation, this involves querying the trust management module to discover the trust value of all nodes on all potential routes, finding the average trust value for these routes, and favouring shorter routes by dividing this average trust value by the length of the route. The getRouteTrustValue returns the most trustworthy source route to the DSRAgent object that then makes use of this route in the subsequent transmission of packets.

A security policy may also be incorporated in the getRouteTrustValue function. In this implementation, a simple security policy was implemented: no source routes that contain a node with a trust value of less than 0.25 are to be considered. More complex security policies may be implemented. However, if the security policy is quite complex, it would be desirable to separate the security policy implementation from the route selection implementation.

4.5 Summary

In this chapter we have discussed the NS-2 network simulator and the manner in which DSR is implemented within NS-2. The implementation of the DSR extensions that are required to build a trust based route selection system have also been discussed in some depth. The next chapter will examine the evaluation of this newly built system.

Chapter 5

Evaluation

After the implementation of the trust based route selection system was completed, the NS-2 network simulator was used to generate a set of results that allows for evaluation of the system. This chapter describes the simulation model used and the results obtained.

Because trust based dynamic source routing is an extension of the DSR protocol, it makes sense to compare the modified version of DSR against standard DSR.

To execute the simulations, we use a PC (500 MHz Celeron Processor with 256 MB RAM) running Mandrake Linux 8.1.

5.1 Measurement Aims

The first objective of the trust based route selection system is to identify behaving and misbehaving nodes, i.e. nodes that can and cannot be relied upon to route packets respectively. Based on this information, the second objective of the trust based route selection system is to improve service in the network, i.e. improve the packet delivery ratios in the network by avoiding misbehaving nodes as much as possible.

In order to determine whether or not nodes in the network are correctly identifying misbehaving nodes, we record the extent to which nodes consider other nodes trustworthy and examine how these trustworthiness levels vary over time. Ideally the

trustworthiness levels of behaving nodes will increase with time, and the trustworthiness levels of misbehaving nodes will decrease with time.

The packet delivery ratio (also known as throughput) is the ratio of the number of packets that are sent to the number of packets that are received by the intended destination. Packet loss can occur due to general network conditions such as congestion, link errors, and unreachable nodes, but packets can also be lost because an intermediate node intentionally drops them. The latter is the only form of packet loss attributable to misbehaving nodes. Because the simulations used are designed to minimise congestion, link errors, and unreachable nodes, the packet delivery ratio provides an indication of the number of packets being dropped by misbehaving nodes in the network. We expect the packet delivery ratios in the network to decrease as the number of misbehaving nodes increases. However, the extent to which the trust based route selection system can cope with misbehaving nodes as compared to standard DSR is what is of primary interest. A comparison of the packet delivery ratios of the modified version of DSR and standard DSR with a varying percentage of misbehaving nodes is thus required.

5.2 Simulation Model

Figure 5 lists both fixed and variable simulation parameters.

In all of our node movement patterns, the nodes choose a destination and move in a straight line towards the destination at a speed uniformly distributed between 0 and 20 m/s. This is the random waypoint model. The speed distribution chosen offers a range of users such as fixed location, walking, or driving a car. Once the node reaches its destination it stops and waits for a certain period of time known as the pause time. The pause time chosen is a variable so as to simulate networks of varying dynamicity.

The nodes communicate using constant bit rate (CBR) node to node connections. CBR is chosen to avoid potential peculiarities associated with more complex

protocols such as TCP. The amount of network traffic chosen (i.e. number of connections made and number of packets sent per connection) simulates an environment in which few packets are being dropped due to network congestion.

Parameter	Value
Application	Constant Bit Rate (CBR)
Radio Range	250m
Packet Size	512 bytes
MAC	802.11
Application Output Rate	4 packets/s
Network Area	*Variable (Mostly 800 x 800m)*
Pause Time	*Variable*
Number of Connections	*Variable*
Transmission Rate	2.0 Mbps
Simulation Time	*Variable (Mostly 600s)*
Node Speed	0 to 20 m/s
Number of Nodes	*Variable (Mostly 15)*
Movement Pattern	Random Waypoint Model
Traffic Pattern	Generated Randomly
% Misbehaving Nodes	*Variable (0 to 40%)*

Figure 5: Simulation Parameters

Some variable percentage of the nodes in the network are configured to misbehave, i.e. drop all packets they receive. This percentage begins at 0% and increases to 40% in small increments. While a network with 40% of misbehaving nodes may seem unrealistic, it is to gain further information about the system and to simulate highly hostile environments that this high figure is used. The same random seed is used

across the 0% to 40% variation range of misbehaving nodes. This ensures that any nodes that misbehave in the 20% case also misbehave in the 40% case, i.e. obstacles present in the lower percentage cases are also present in the higher percentage cases.

Simulations that were executed incorporated the following variable parameters (and fixed parameters as shown in figure 5):

- Simulation time = 600s, Pause time = 600s, No. of nodes = 15, Area = 600 x 600m
- Simulation time = 600s, Pause time = 300s, No. of nodes = 15, Area = 600 x 600m
- Simulation time = 600s, Pause time = 60s, No. of nodes = 15, Area = 600 x 600m
- Simulation time = 600s, Pause time = 30s, No. of nodes = 15, Area = 600 x 600m
- Simulation time = 300s, Pause time = 300s, No. of nodes = 10, Area = 500 x 500m
- Simulation time = 300s, Pause time = 60s, No. of nodes = 10, Area = 500 x 500m
- Simulation time = 300s, Pause time = 30s, No. of nodes = 10, Area = 500 x 500m
- Simulation time = 500s, Pause time = 500s, No. of nodes = 30, Area = 800 x 800m

For all of the above simulations, the percentage of misbehaving nodes varied as described above. The wide range of simulation scenarios outlined above was used so as to gain a reasonable level of insight into the system.

5.3 Results

Two sets of results will now be displayed and discussed. These results correspond to the measurement aims outlined above, i.e. the variation of trust values of nodes over time and the variation of packet delivery ratios with a varying percentage of misbehaving nodes.

5.3.1 Identification of Misbehaving Nodes

In the simulations executed, all nodes in the network recorded their trust ratings of all other nodes. These trust ratings change continuously over time. The resulting amount of output data is much too large to be presented here. We will now examine

and discuss a sample of this output data. The sample chosen is deemed to be representative of all the output data generated.

The following graphs measure the variation of trust values of nodes in the network with time, as perceived by some node in the network. This node is deemed to be representative of all nodes in the network, i.e. is a typical node whose results are typical of other nodes. It should be noted that a trust value for a node will not exist until that node is encountered.

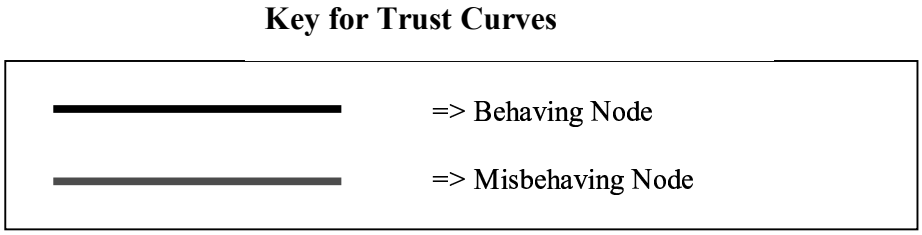


Figure 6: Simulation Time = 500s, Pause Time = 500s, No. of Nodes = 30, Area = 800 x 800m.

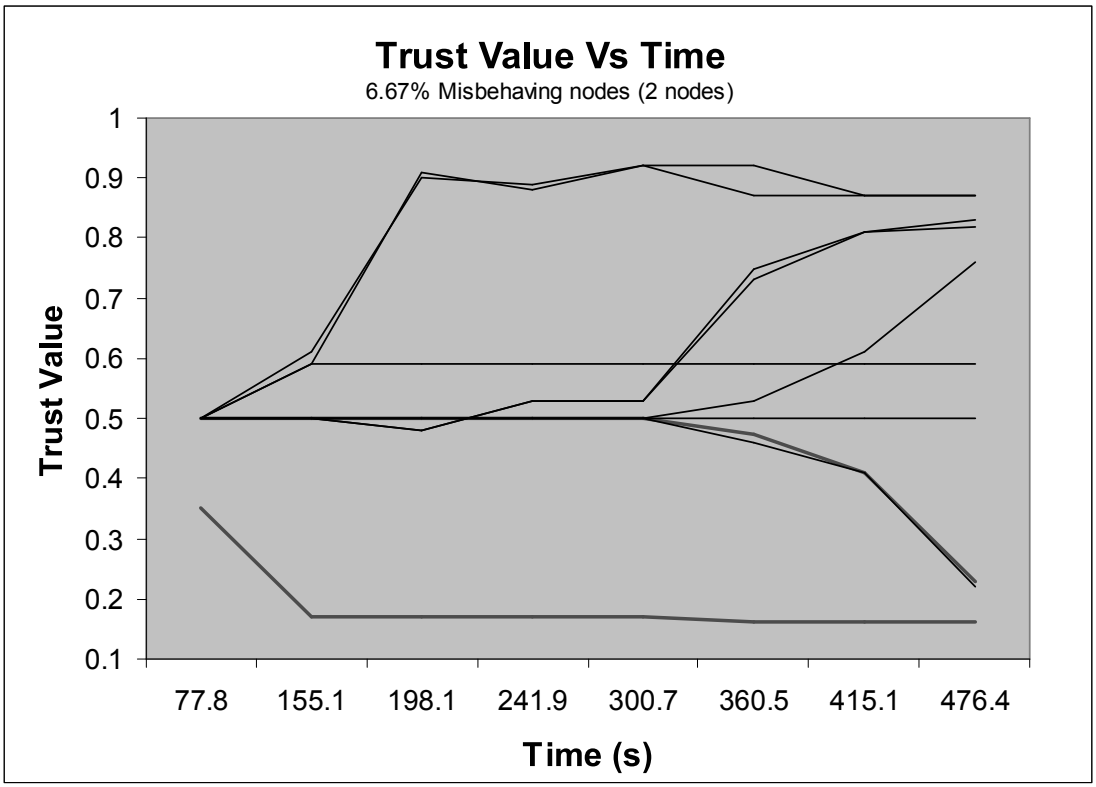


Figure 7: Simulation Time = 600s, Pause Time = 600s, No. of Nodes = 15, Area = 600 x 600m.

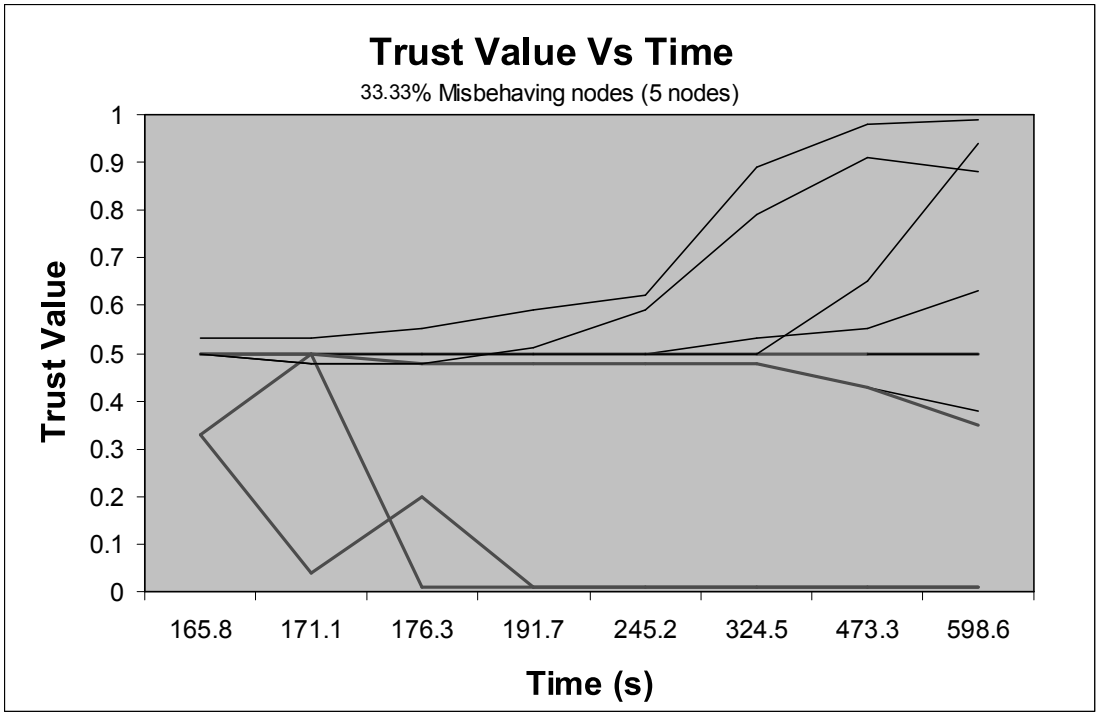


Figure 8: Simulation Time = 600s, Pause Time = 300s, No. of Nodes = 15, Area = 600 x 600m.

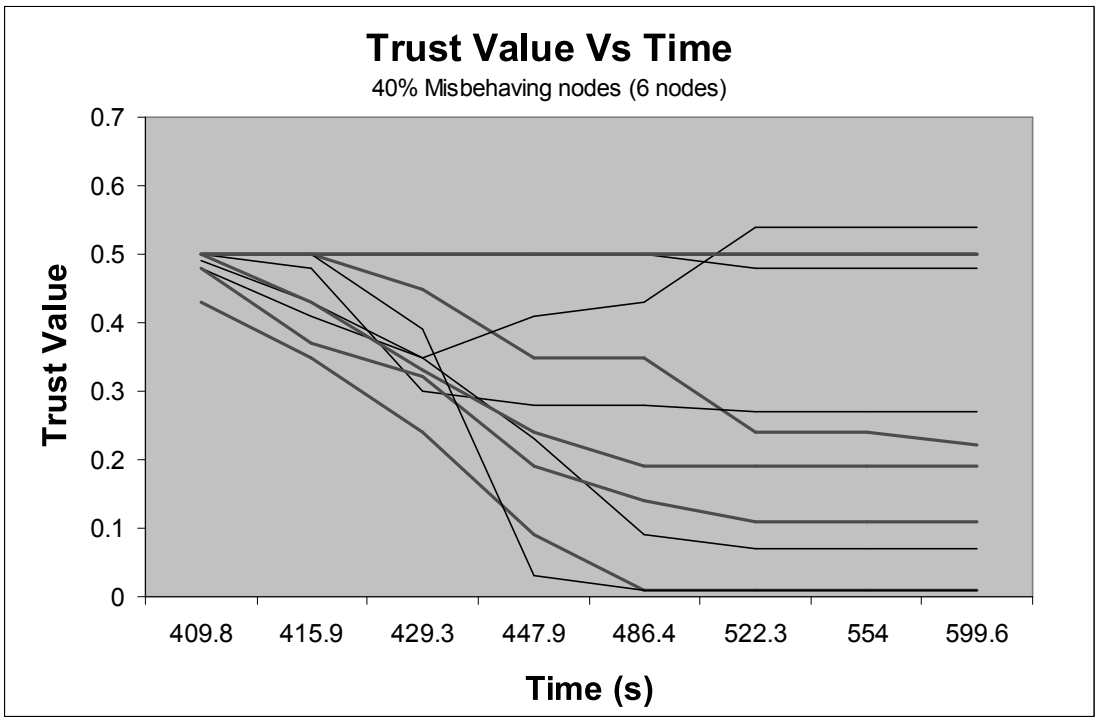


Figure 9: Simulation Time = 600s, Pause Time = 300s, No. of Nodes = 15, Area = 600 x 600m.

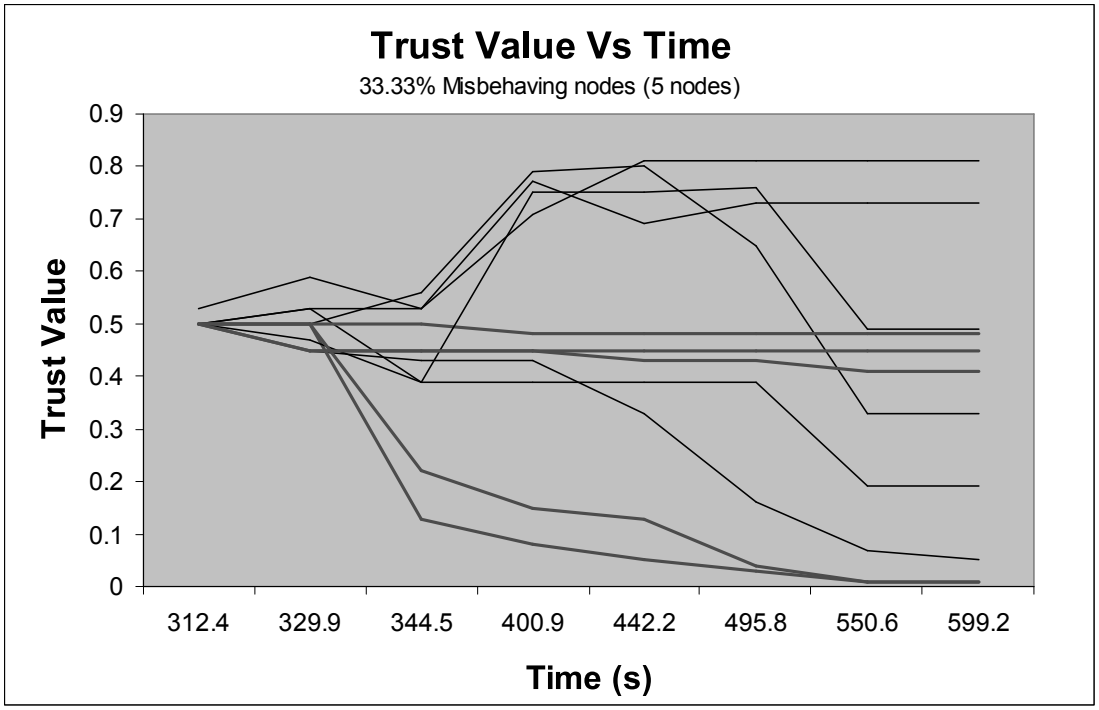


Figure 60: Simulation Time = 600s, Pause Time = 30s, No. of Nodes = 15, Area = 600 x 600m.

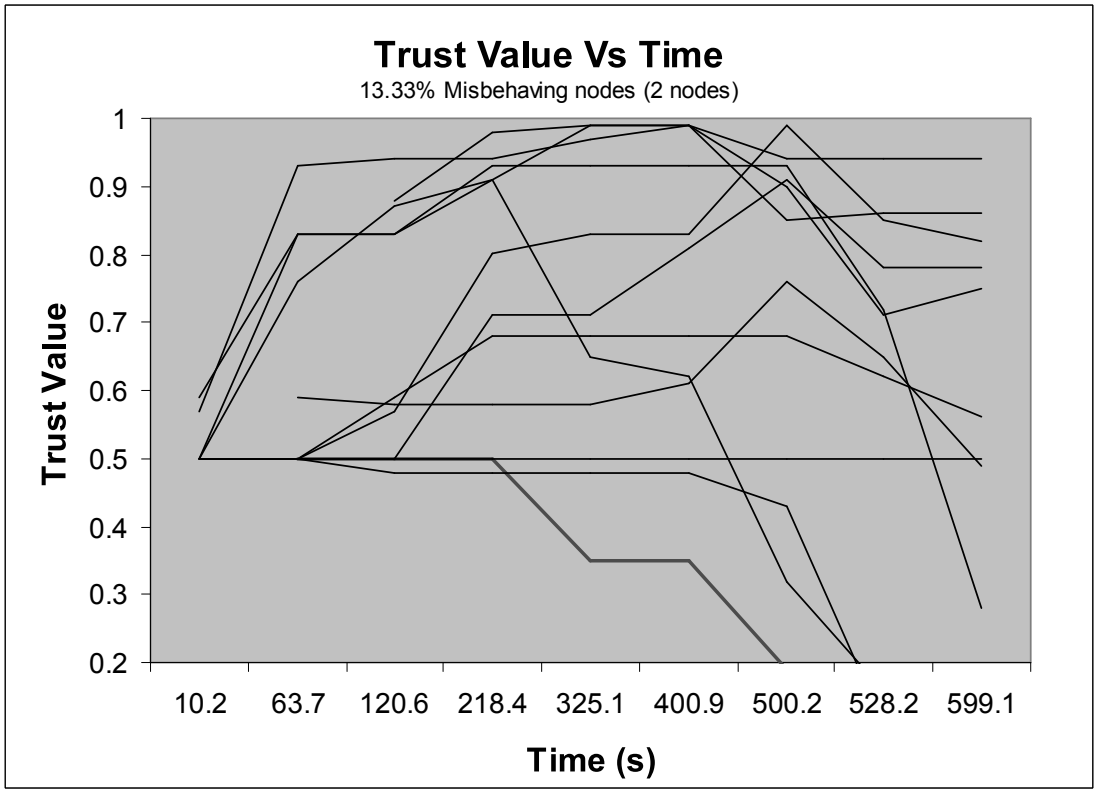
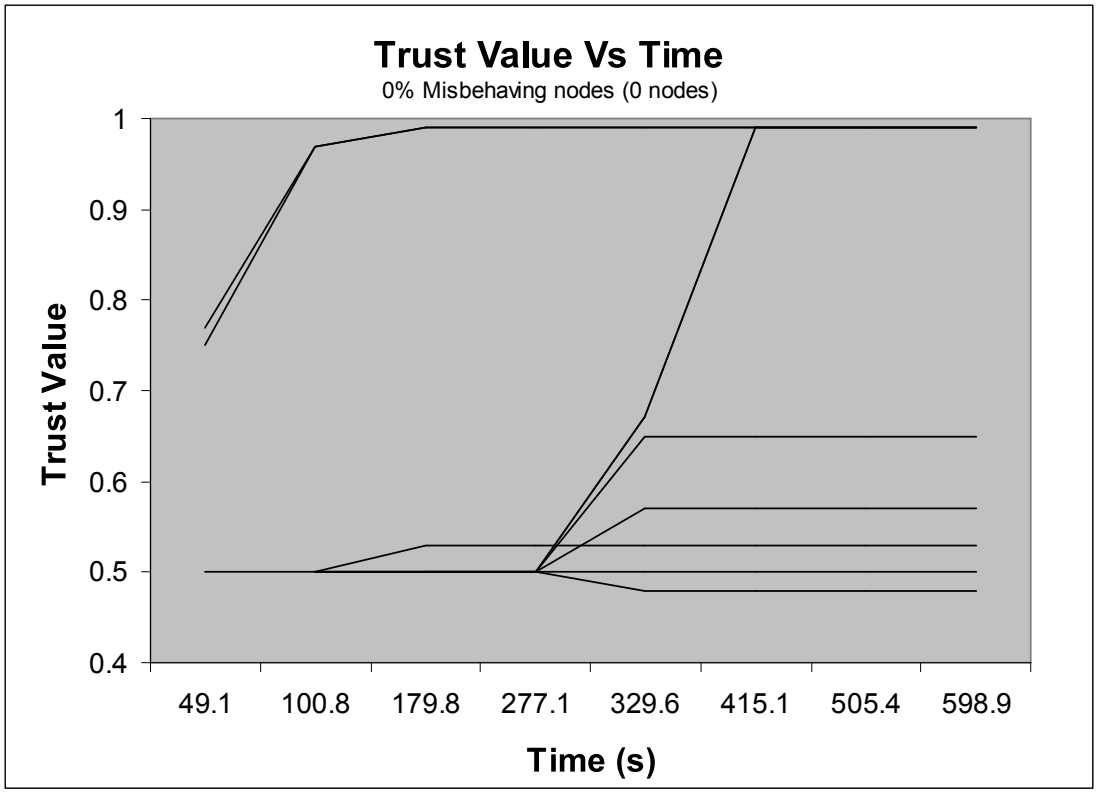


Figure 71: Simulation Time = 600s, Pause Time = 300s, No. of Nodes = 15, Area = 600 x 600m.



It is clear from the above graphs that in the majority of situations, misbehaving nodes are being clearly identified by other nodes in the network. In addition, in the majority of situations, a certain amount of behaving nodes are being clearly identified by other nodes in the network. However, this is not always the case. In many situations behaving nodes are being unfairly assigned a low trust value.

There are several reasons why a behaving node may be deemed to be acting maliciously by other nodes in the network. If a node is unreachable due to its geographical location, it will not be able to receive packets intended for it and will receive a poor trust rating. However, using reasonable simulation parameters offsets the likelihood of this scenario occurring.

If a behaving node is geographically located next to a misbehaving node, the probability that both nodes are on a high number of the same source routes is high. In this scenario it is likely that the behaving node will receive a poor trust rating because of the continued failure of all source routes passing through both nodes. This problem

will be further exacerbated when the amount of movement in the network is low, i.e. when the chosen pause time is high.

When the random traffic pattern is making use of low pause times, there is a potential for good routes (i.e. routes with no misbehaving nodes) to be broken while being made use of. In this situation, the source route will be deemed to have failed and the behaving nodes will have their trust values decremented. However, a possible solution to this problem involves examining route error packets that are received so as to determine the cause of failure of the route. A route error packet indicates a broken link, i.e. an intermediate node did not receive data because it has moved out of range. This scenario is different to that where an intermediate node drops packets. In this case, a route error packet will not be generated because the receipt of packets by the misbehaving node will have been confirmed by the previous node (despite the fact that the misbehaving node will subsequently drop the packets). By examining route error packets it is possible to determine if a good route has been broken and hence possible to not decrease the trust values of the nodes on the relevant route, i.e. it is possible to distinguish between a route being broken as a result of movement of nodes and a route containing a node that is dropping packets. This solution has not been implemented by the author.

In general, the performance of the system is better when high pause times are chosen, i.e. when the network is static. However, it is difficult for trust based DSR to gain a significant advantage over DSR in this scenario because of the fact that DSR will also discover reliable routes quickly in a static network.

5.3.2 Packet Delivery Ratios

Traces are special objects in NS-2 that are used to store information about each packet that is sent, dropped, and received during run-time. This information is written to a file and is post-processed by a perl script so as to extract the required information. By determining the number of packets sent, dropped, and received, it is possible to calculate the packet delivery ratio in the network for a particular simulation.

The graphs below show how the packet delivery ratio varies as the number of misbehaving nodes in the network increases. All graphs depict this information for both trust based DSR (TBDSR) and standard DSR, so that a comparison can be made. Each graph represents a different simulation scenario.

Figure 82: Simulation Time = 500s, Pause Time = 500s, No. of Nodes = 30, Area = 800 x 800m.

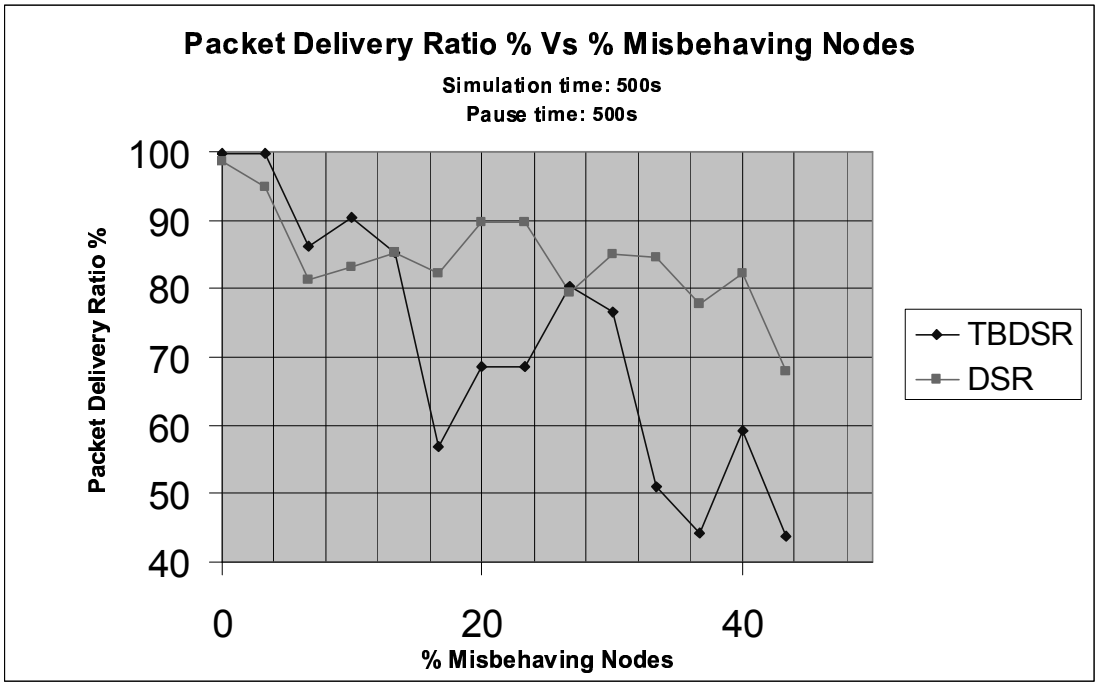


Figure 13: Simulation Time = 600s, Pause Time = 600s, No. of Nodes = 15, Area = 600 x 600m.

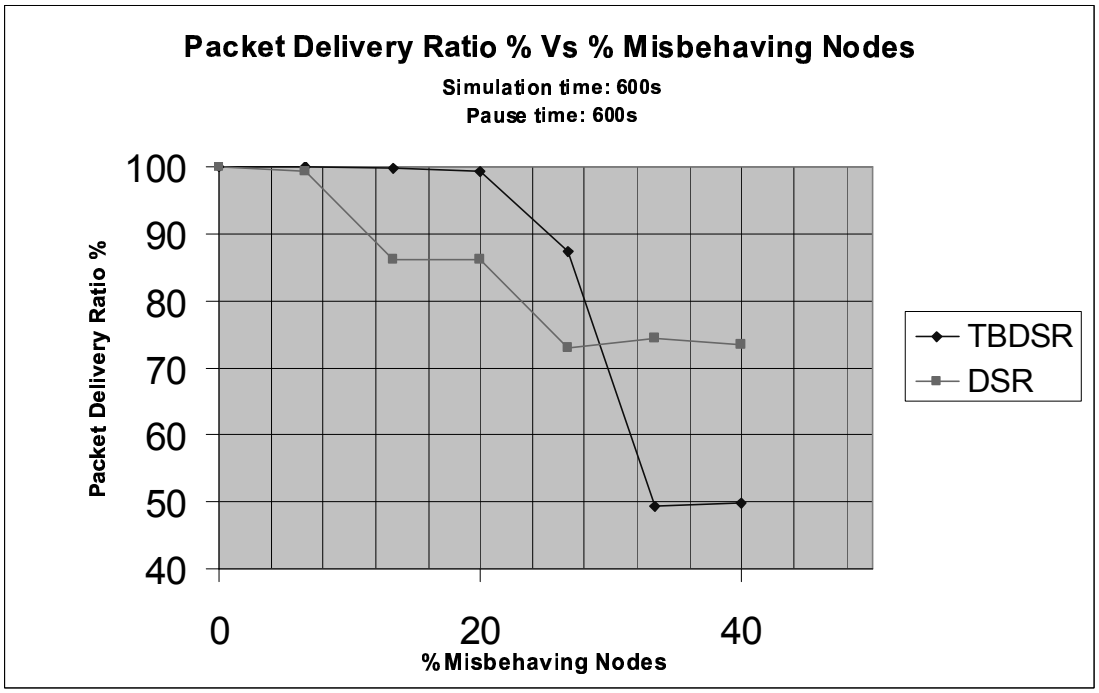


Figure 14: Simulation Time = 600s, Pause Time = 300s, No. of Nodes = 15, Area = 600 x 600m.

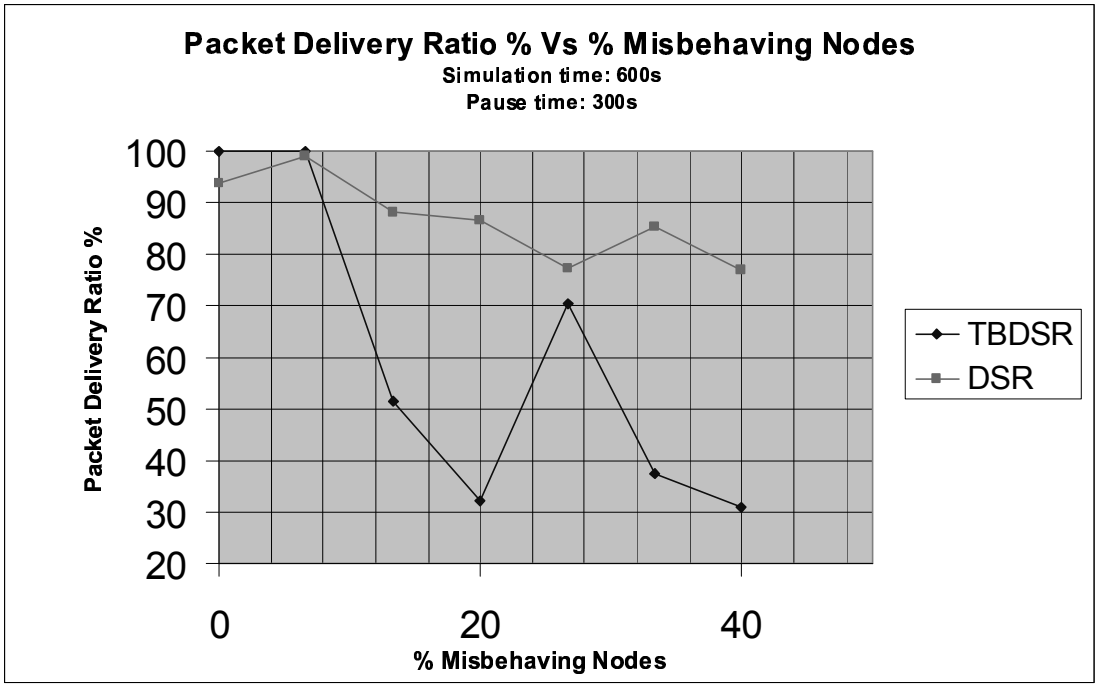


Figure 15: Simulation Time = 600s, Pause Time = 30s, No. of Nodes = 15, Area = 600 x 600m.

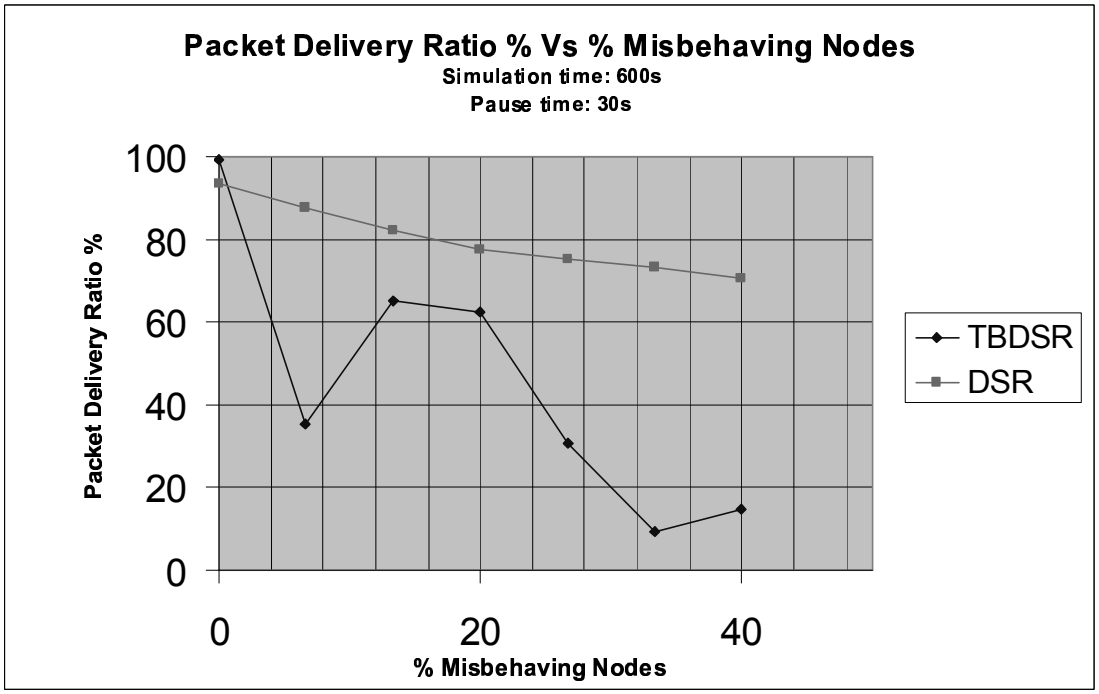
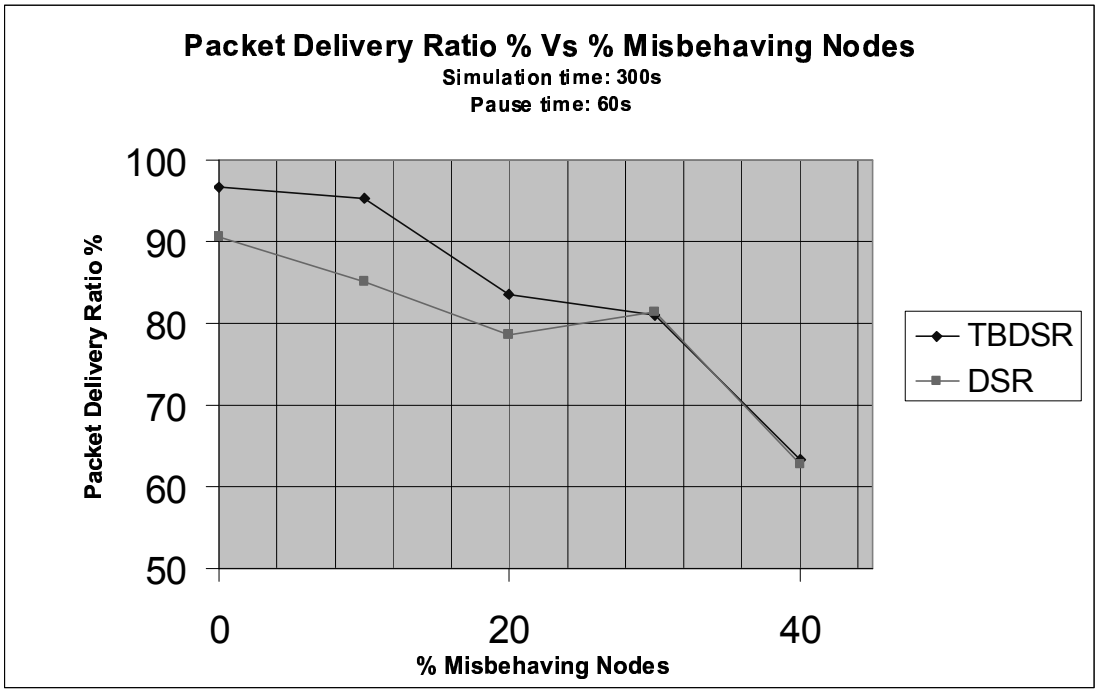


Figure 16: Simulation Time = 300s, Pause Time = 60s, No. of Nodes = 10, Area = 500 x 500m.



The above graphs do provide a certain amount of information regarding the ability of TBDSR to offer an improved service over standard DSR when misbehaving nodes are present in the network. However, there is also a degree of ambiguity resulting from the information presented in these graphs.

TBDSR appears to compete with (and in certain cases outperform) DSR when either the pause time being used is high or the percentage of misbehaving nodes in the network is low. However, when the network is dynamic (i.e. the pause time is low) TBDSR does not offer a packet delivery ratio which is as high as that offered by standard DSR. There are several reasons why this may be the case.

As was discussed in the previous section, in a highly dynamic network there is a potential for good routes to be broken while they are being used. This may result in behaving nodes receiving a poor trust rating. If this occurs, the route selection algorithm cannot function as effectively as it is designed because it may decide to use unreliable routes over reliable routes for transmission.

The implementation of TBDSR outlined in this dissertation does not retain trust information between sessions. This reflects a scenario in which a node moves to a new environment between sessions, e.g. a person living in Dublin arrives in New York and turns on their PDA. We believe that the retention of trust information between sessions would offer TBDSR a considerable advantage over standard DSR. In this case TBDSR would know immediately which routes are reliable. In the present implementation, both TBDSR and DSR may have to try several routes before finding a reliable route, but TBDSR will have to do more work than DSR for the same result. The retention of trust information would reflect a scenario in which a node operates in the same environment on a daily basis, e.g. a person with a PDA goes to work, goes to the coffee shop for lunch, goes to the gym in the evening etc.

5.4 Summary

In this chapter we have discussed the simulation model used to evaluate trust based DSR and we have examined the results of this evaluation. The identification of misbehaving nodes is being achieved with a high degree of success in the majority of simulation scenarios. However, this success does not facilitate TBDSR outperforming DSR in the domain of packet delivery ratios. It is only in limited situations that TBDSR can compete with DSR in the packet delivery ratio achieved. Some of the possible reasons for this and some of the possible optimisations that could be implemented have been discussed.

Chapter 6

Conclusions

This thesis has described the analysis, design, implementation, and evaluation of a trust based route selection system for use in mobile ad-hoc networks. The system was implemented in the NS-2 network simulator as an extension to the existing dynamic source routing protocol.

In chapter 1 an introduction to mobile ad-hoc networks is presented. This is followed in chapter 2 by an in depth discussion of the key technologies, issues, and problems associated with ad-hoc networks. Security and reliability problems and vulnerabilities of ad-hoc networks are highlighted and this provides the motivation for this thesis.

In an ad-hoc network the presence of a trusted third party such as a certification authority and the known identity of nodes in the network cannot be assumed. Blindly trusting nodes that may be malfunctioning or malicious exposes the local node to a wide variety of vulnerabilities. The first objective of the trust based route selection system was to clearly identify nodes in the network that misbehave by dropping packets. The second objective was to incorporate this information in a route selection algorithm with the intention of improving the reliability of service in the network.

From an examination of the results presented in chapter 5 it is clear that the first objective has been achieved. Misbehaving nodes in the network are being clearly identified by other nodes in the majority of situations. However, an improvement in the reliability of service in the network as measured by the packet delivery ratios

being realised is only being achieved in a small portion of cases. To completely fulfil the second objective it is necessary for future work to be carried out in this area.

6.1 Future Work

In order to further enhance the trust based route selection system so as to improve on the reliability and security being achieved in ad-hoc networks which contain misbehaving nodes, work can be directed to the following areas.

- Allow the trust management module to gather and store further trust related information about nodes encountered. Information about recent behaviour of nodes, recommendations from trusted nodes, and suspicions about identity swapping could all be incorporated. This would allow for more complex selection heuristics and security policies to be implemented.

- Retention of trust information between sessions would allow the trust based route selection system to identify reliable routes immediately. This would reflect the scenario in which a node encountered a similar set of nodes on a regular basis, e.g. a person goes from their house to work, to the gym, and back to their house on a daily basis. We believe that the immediate identification of reliable routes would offer trust based DSR a considerable advantage over standard DSR.

- If the above modifications were to be made to the trust based route selection system, it could be integrated into other systems such as the CONFIDANT protocol [Buchegger] outlined in chapter 2. This protocol assumes that nodes are authenticated and that identity swapping is not allowed. Based on these assumptions, the CONFIDANT protocol achieves excellent service in an ad-hoc network in which there are a large number of misbehaving nodes. A modified version of the trust based route selection system could be incorporated into the CONFIDANT protocol so that the assumptions outlined above may no longer be necessary.

- Further evaluation such as micro-benchmarking is required to determine the mechanism overload of trust based DSR and provide further insight into the mechanism.

Bibliography

[Abdul] Alfarez Abdul-Rahman. The PGP Trust Model.

[Blaze 1] M. Blaze, J. Feigenbaum, J. Lacy. Decentralised Trust Management. Proc. IEEE Symposium on Security and Privacy, P.164-173, 1996.

[Blaze 2] M. Blaze, J. Feigenbaum, et al. The Role of Trust Management in Distributed Systems Security. 1999.

[Blaze 3] M. Blaze, J.Feigenbaum, A. Keromytis. KeyNote: Trust Management for Public-Key Infrastructures. 1998.

[Broch] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, Jorjeta Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. ACM 1998.

[Buchegger] Sonja Buchegger, Jean-Yves Le Bouche. Performance Analysis of the CONFIDANT Protocol Cooperation Of Nodes – Fairness In Dynamic Ad-hoc Networks. 2002.

[Chu] Yang-Hua Chu, Joan Feigenbaum, Brian LaMacchia, Paul Resnick, Martin Strauss. REFEREE: Trust Management for Web Applications. AT&T Research Labs, 1997.

[Grandison] Tyrone Grandison, Morris Sloman. A Survey Of Trust In Internet Applications. IEEE Communications Surveys 2000.

[Hubaux] Jean-Pierre Hubaux, Levente Buttyan, Srdan Capkun. The Quest for Security in Mobile Ad Hoc Networks. Proceedings of the ACM 2001.

[IETF MANET WG] http://www.ietf.org/html.charters/manet-charter.html

[Johnson 1] D. Johnson, D. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In Mobile Computing, chapter 5, P.153-181, 1996.

[Johnson 2] D. Johnson, D. Maltz, Yih-Chun Hu, J. Jetcheva. The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks. Internet Draft, IETF MANET Working Group, 2001.

[Jøsang] A. Jøsang. Prospectives for Modelling Trust in Information Security. Australasian Conf. Information Security and Privacy, 1997.

[Khare] Rohit Khare, Adam Rifkin. Trust Management On The World Wide Web. First Monday.

[Kini] A. Kini and J.Choobineh. Trust in Electronic Commerce: Definition and Theoretical Considerations. 31st Annual Hawaii Int'l. Conf. System Sciences 1998.

[Marti] S. Marti, T. Giuli, K. Lai, M. Baker. Mitigating Routing Misbehaviour in Mobile Ad Hoc Networks. Proceedings of MOBICOM 2000.

[Nedos] Andronikos Nedos. Direction Based Routing for Mobile Ad Hoc Networks. Trinity College Dublin, 2001.

[O Connell] Paul O'Connell. Collaborative Ad hoc Applications. Trinity College Dublin, 2000.

[Perkins 1] Charles E. Perkins, Elizabeth M. Royer, Samir R. Das, Mahesh K. Marina. Performance Comparison of Two On-Demand Routing Protocols for Ad Hoc Networks. IEEE Personal Communications 2001.

[Perkins 2] Charles E. Perkins, Elizabeth M. Royer. Ad-hoc On-Demand Distance Vector Routing. Proc. 2nd IEEE Wksp. Mobile Comp. Sys. and Apps., Feb 1999.

[Pfleeger] Charles P. Pfleeger. Security in Computing, 2nd Ed. Prentice Hall, 1997.

[Saltzer] J.H. Saltzer, D.P. Reed, D.D. Clark. End-to-End Arguments in System Design. M.I.T Laboratory for Computer Science.

[Vanhala] Anne Vanhala. Security in Ad-hoc Networks. University of Helsinki, 2000.

[Zhou] Lidong Zhou, Zygmunt J. Haas. Securing Ad Hoc Networks. IEEE Network,