Cooperation Enforcement in Ad-hoc Wireless Networks

Prepared by:
David Waiting

Supervised by:
Neco Ventura

Department of Electrical Engineering
University of Cape Town
2005

This dissertation is submitted to the University of Cape Town
in fulfillment of the academic requirements
for the Degree of Master of Science in Engineering

April 2005

Declaration

I declare that this thesis is my own work. Where collaboration with other people has taken place, or material generated by other researchers is included, the parties and/or material are indicated in the acknowledgements or references as appropriate.

This work is being submitted for the Master of Science Degree in Electrical Engineering at the University of Cape Town. It has not been submitted to any other university for any other degree or examination.

D Waiting

David Waiting

21 April 2005

Date

Acknowledgements

I would like to acknowledge the following people for their assistance with this work:

- Mr Neco Ventura, my supervisor.
- Mr Albert Hasson, for his technical assistance and invaluable criticism.
- The past and present members of the UCT Communications Research Group, for their advice and feedback.

Synopsis

Ad-hoc networks consist of a system of wireless nodes that can freely and dynamically self-organise into a working network topology. This allows people to internetwork seamlessly in areas that have no preexisting communication infrastructure. Nodes are expected to forward the traffic of other nodes in order for the packets to reach their final destination.

It is envisaged that community ad-hoc networks will become widespread in the near future, as they require no administrative support. Nodes will be able to enter and leave the network as they choose. An unfortunate result is that in an open multi-agent system such as this, the role-players tend to be unreliable and self-interested.

Because the nodes in an ad-hoc network often have limited battery and processing power, it is in their best interest to conserve this power. This leads to nodes exhibiting selfish behaviour. Selfish nodes utilise the resources of an ad-hoc network but do not forward the traffic of other nodes. This uncooperative behaviour reduces the quality of network links and may cause the network to fail.

Cooperation enforcement techniques aim to entice cooperation amongst the nodes of an ad-hoc network. A distributed trust model is presented in this research that aims to evaluate the trustworthiness of nodes in an ad-hoc network. If nodes are found to be untrustworthy, then they are punished until they become cooperative. Otherwise they are excluded from the network.

A test-bed framework is created that implements the cooperation enforcement model. Evaluations are performed that determine the effects of selfish nodes on ad-hoc networks and establish whether a cooperation enforcement model can effectively identify selfish nodes.

Furthermore, it is beneficial to exclude selfish nodes from routing decisions. This increases the reliability and throughput of links. A metric is proposed that rates a route according to the trustworthiness of the nodes that make up the route. It is shown that in some scenarios this metric can improve the routing decision and increase the average data throughput between nodes.

Contents

List of Figures

List of Tables

Abbreviations

ARP - Address Resolution Protocol

DSDV - Destination Sequenced Distance Vector

DSR - Dynamic Source Routing

EDGE - Enhanced Data Rates for Global Evolution

GPRS - General Packet Radio Service

GSM - Global System for Mobile Communications

IBSS - Independent Basic Service Set

IEEE - Institute of Electrical and Electronic Engineers

IPv4 - Internet Protocol version 4

ISM - Industrial Scientific Medical

ITU - International Telecommunication Union

LAN - Local Area Network

MAC - Medium Access Control

NAT - Network Address Translation

OS - Operating System

OSI - Open Systems Interconnection

PCI - Personal Computer Interconnect

PCMCIA - Personal Computer Memory Card International Association

PCF - Point Coordination Function

PD - Prisoner's Dilemma

PDA - Personal Digital Assistant

SIG - Special Interest Group

TCP - Transmission Control Protocol

TTL - Time To Live

UDP - User Datagram Protocol

WiFi - Wireless Fidelity

WPAN - Wireless Personal Area Network

Chapter 1

Introduction

The ability to communicate effectively and timeously is fundamental to human interaction. In recent times, huge amounts of communication is facilitated by computers. The proliferation of the Internet, and applications such as email, is proof that humans have accepted computers as the de facto standard for modern communication. We now see computers being used, not only for communicating small amounts of data in the form of hypertext or email, but also for bandwidth hungry applications such as Internet telephony and video conferencing. It has become increasingly necessary to investigate methods for optimising the flow of data between all forms of electronic devices, in order to accommodate the growing needs of the applications running on these devices.

Electronic devices are traditionally connected by wires or cables. Wired connections provide high data rates and reliable information transfer. However, they are restricted in that they require some form of physical human intervention before devices can communicate. For example, laying cables and plugging in wires. This restriction makes wired interfaces inappropriate for computers that are highly mobile, such as PDAs, or devices that are inaccessible by wires, such as satellites. For this reason it is often appropriate to communicate wirelessly using either radio waves, microwaves, infrared or sonar. In this research the investigation is limited to communications via electromagnetic waves.

James Clerk Maxwell forecast the propagation laws of radio waves in 1864. In 1894 British Scientist Oliver Lodge succeeded in transmitting wireless signals over 150 yards, and soon after Guglielmo Marconi devised the first practical system of wireless telegraphy. Since then wireless microwave signals allow for communication between devices that are thousands of kilometres apart. While we will probably never see the end of wired connections, there is an

increasing trend to use wireless interfaces to facilitate communication between machines.

A widespread form of wireless communication is the cellular telephone network. A number of standards have been developed for these types of networks. The most popular being the Global System for Mobile Communications (GSM), and more recently the third generation (3G) mobile phone technologies, covered by the ITU IMT-2000 family [1]. Wireless communication between computers was not thoroughly explored until recently with the growing number of notebook computers and hand-held devices. Currently, the dominating technologies for these devices are the IEEE 802.11 family of protocols for local area networks (WiFi) [2] and the Bluetooth protocol for short-range piconets [3]. The growing demand and popularity of these protocols has resulted in the widespread availability of hardware that implements these technologies.

However, an emerging trend is to connect wireless devices in a peer-to-peer fashion. Hosts are thus able to connect directly to each other without the need for base-stations or access points. The resulting network is known as an *ad-hoc network*. The term *ad-hoc* has been used in many different contexts in the literature. In this research the term is reserved exclusively to describe a network that requires no fixed network infrastructure or administrative support. In addition to this, an ad-hoc network is defined as a network in which no node assumes control over the resources or administration of the network. All nodes are of equal importance in maintaining the functionality of the network.

The characteristics of ad-hoc networks make them well suited for environments in which no infrastructure exists, either due to natural disaster, unsuitability of the surrounding environment, or the temporary nature of the network. Ad-hoc networks lend themselves to the following scenarios:

- Following natural disasters where existing network infrastructure has been destroyed by earthquake or fire;
- Rural locations where the economics of scale do not warrant the installation of base-stations;
- Locations that do not have access to electricity in order to power network infrastructure;
- Conferences and exhibitions;
- Highly mobile devices that change the network topology frequently;

- Community ad-hoc networks.

Community ad-hoc networks provide a scalable and cost-effective mechanism to facilitate communications amongst several users in close proximity. In this research, community ad-hoc networks are discussed in which the participating nodes are computing devices running various applications. These devices may take the form of PDAs, laptop computers, or desktop machines with wireless network adaptors. The task of deploying a reliable and scalable ad-hoc network of wireless devices is intricate because a number of factors must be taken into account, such as radio connectivity, routing protocols, and security issues.

An unique problem with wireless networks is radio range. To increase the range of wireless communications the simplest solution is to increase the transmission power of the radio transmitter. However, the maximum power of radio transmitters is limited by various factors including the power resources of the transmitting node and international law. Thus the range of wireless devices can only be increased to a certain point, beyond which repeater stations are needed to boost the signal. Ad-hoc networks should require no existing infrastructure and a repeater station would violate this condition. Thus, it is up to the individual nodes of the network to forward traffic to its destination.

If a message in an ad-hoc network must travel through other nodes before it reaches its destination, the network is said to be a multi-hop wireless network. This is demonstrated in Figure 1.1, where the radio range of the source node does not reach the destination node. In this work, the source and destination nodes are referred to as the *corresponding nodes*. The corresponding nodes do not have a physical connection, however, a logical network connection exists between them. A number of routing protocols have been proposed in the literature in order to facilitate the forwarding of traffic in ad-hoc networks.

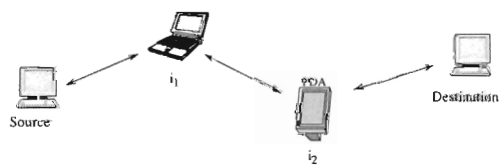

Figure 1.1: Packets must pass through intermediate nodes i_1 and i_2 to reach the corresponding nodes.

Forwarding other node's traffic comes at a cost. Especially if devices have limited processing power or battery life. This leads to a very interesting issue of ad-hoc networks that is rarely

seen in wireline network architectures: the tendency for the nodes in the network to become selfish users of the network [4]. By this we mean that the node will join the network and use the resources of other nodes to forward their traffic, but will not participate in forwarding either routing protocol information or other nodes' traffic.

Nodes may have various reasons as to why they act in a selfish manner. For example, they may have no choice but to cheat in order to conserve their battery life. They are not willfully trying to disrupt the network. However, this work focusses on nodes that may be acting in a selfish fashion because it suits them better to conserve all their resources for their own tasks. By the very nature of ad-hoc networks the role-players are often untrustworthy. Because ad-hoc networks are distributed, no single node can serve to verify the trustworthiness of a new node in the network. Some form of distributed mechanism must be established in order to verify that nodes will act in a cooperative fashion. Without this mechanism, the network could quite easily become chaotic, with all nodes choosing to act selfishly, and this will cause the network to fail.

While this problem has not been thoroughly explored by network engineers, it is common in the field of economics. Economic models that describe the payoffs for various parties in a negotiation have been used for many years to describe a range of scenarios, from the outcome of wars to different market systems [5]. In this research these models will be applied to ad-hoc networks in order to analyse mechanisms that can counteract the effect of selfish nodes. Both analytical and experimental evaluations are performed to address the problem of enforcing cooperation of nodes in a test-bed framework. Game Theory is a tool for modeling the choice of strategy between self-interested users. It is used in this work to predict the behaviour of nodes that are faced with the option to cheat and the repercussions if they do so.

1.1 Thesis Objectives

The primary objective of this research is to implement a trust and reputation model that effectively enforces node cooperation on an evaluation framework. In order to do this successfully, it is necessary to develop a suitable test-bed architecture capable of fulfilling the needs of a cooperation enforcement mechanism. This generic experimental test-bed should be capable of accommodating experiments in the field of node cooperation, but also in the fields of developing ad-hoc routing algorithms, testing throughput performance of

ad-hoc networks, and addressing security issues.

An analytical evaluation is to be performed on the cooperation enforcement scheme using economic models. In addition, experiment observations should determine the detrimental effects of uncooperative nodes on ad-hoc networks. Further evaluations must rate the effectiveness of the cooperation enforcement scheme, and to what extent this scheme can improve network reliability and efficiency. These results are to be analysed to determine whether or not a cooperation enforcement scheme is necessary and how effectively the scheme achieves its goals.

This research strives to deliver meaningful data and analysis of the proposed economic and trust models. The specifics regarding the implementation of the evaluation framework architecture will be discussed to assist further work in this field. Future researchers will be able to emulate and extend the evaluation framework and the experiments conducted.

1.2 Scope and Limitations

The field of ad-hoc wireless networks is very broad and there are a number of issues surrounding the implementation of a suitable test-bed framework. This research will attempt to discuss all of the framework design considerations. However, a full discussion into all design issues including physical layer protocols, adaptive routing algorithms and quality of service factors, is beyond the scope of this thesis.

Moreover, this work attempts to accurately model the behaviour of the nodes in an ad-hoc wireless network. At this time not enough is known about the tendency of nodes in such networks to become selfish. It is not known if the models that are used to describe the behaviour of nodes in an ad-hoc network are a true representation of their behaviour. This work will not attempt to analyse or improve these models.

As with all security mechanisms, there is no doubt that once measures are put into place to enforce cooperation, a method will be devised to counteract these measures. As such this is a continually evolving field of research, and at this infant stage it is impossible to predict what future threats ad-hoc networks will face.

Because this research involves the design and implementation of an evaluation framework, there are resource constraints that limit the number of active nodes in the network. While this is not a problem in the current field of research, the limited size of the test-bed may

restrict further investigation and the test-bed will need to grow accordingly. Simulations are a very effective tool in predicting network performance but they do not take into account real-world factors, and often do not reflect the true behaviour of networks. The choice to create a practical test-bed framework, rather than rely on simulations, limits this research in that it is not as easy to adapt a physical test-bed to new experiments and topology changes.

To model the interaction of nodes in ad-hoc networks, non-zero sum games are used. While these games are a useful tool they do not fully represent the interactions of nodes as they do not take into account varying traffic loads, changing network topologies and the quality of service needs of the applications that may be running on the nodes.

There will be no specific discussion into sensor networks. Sensor networks consist of inexpensive battery-powered nodes consisting of radio transceivers and small microcontrollers. The nodes of a sensor network are spatially distributed, and often monitor temperature changes or other environmental factors. Typically the nodes have very limited battery life and processing power. While the concepts presented in this paper may well be useful in sensor networks, they are outside the immediate scope of this work, as these networks are typically maintained by a single authority. Thus they are not prone to the uncooperative behaviour addressed in this research.

1.3 Thesis Outline

The format of the rest of this thesis is as follows:

A thorough review of research in the field of ad-hoc wireless networks is presented in Chapter 2. In this chapter existing test-beds and experiments are analysed. Useful work and concepts are extracted and applied to this work. A discussion is held of applicable studies in the field of game theory and microeconomics.

In Chapter 3, the need for cooperation enforcement models in ad-hoc wireless networks is reviewed. A novel trust-based cooperation enforcement model is formulated and is analysed using a game theoretic approach, and the Prisoner's Dilemma is presented.

In Chapter 4, an evaluation framework architecture is discussed, including a suggested modification to the Dynamic Source Routing protocol. The implementation of this architecture including the various choices of hardware, software and protocols are reviewed,

and the completed test-bed implementation is presented. The chapter suggests ideas for other researchers to install similar test-beds and working installations of multi-hop ad-hoc networks. The Click Modular Router project is discussed and a flooding mechanism configuration is presented. Extensions to the Click router specific to the experimental test-bed are also described.

In Chapter 5, throughput and reliability tests are performed on the evaluation test-bed. The tests, performed in the presence of selfish nodes, highlight the need for cooperation enforcement in ad-hoc wireless networks. In addition, an Iterated Prisoner's Dilemma tournament is held. The tournament consists of all the nodes in the test-bed playing a version of the Iterated Prisoner's Dilemma game against each other. Certain nodes in the game exhibit various degrees of un-cooperativeness. The tournament illustrates the cooperation enforcement model's ability to identify selfish nodes. In a separate evaluation, the cooperation enforcement model is used to find trusted routes through ad-hoc networks. It is shown that by using trusted routes, over minimum hop-count routes, network throughput can be increased dramatically in certain situations.

In Chapter 6, conclusions, future work and recommendations are presented.

Chapter 2

Related Work

2.1 Introduction

Existing literature in the fields of multi-hop ad-hoc networks, node cooperation enforcement and economic preferences are reviewed in this chapter. Current experimental test-bed architectures are presented, including the successful *Grid* and *Roofnet* projects.

In addition, a review is performed on existing cooperation enforcement models. In the field of economics, work that describes the interaction of nodes as non-zero-games is discussed.

2.2 Experimental Ad-hoc Networks

In recent years there has been an increasing interest in the field of ad-hoc networks. This is no doubt due to the increase in popularity of wireless hand-held devices. As such, a number of research groups now focus entirely on ad-hoc networks and the issues surrounding such networks. There is a focus on the IEEE 802.11[1] family of protocols for physical layer transmissions, as the majority of work in the field of ad-hoc networks utilises 802.11b due to its relatively high data rates, long range and widespread popularity. Below is a review of research carried out on networks that are of a similar nature to the test-bed developed as part of this research.

The MIT LCS's Parallel and Distributed Operating Systems Group are currently involved with the *Grid Ad-hoc Networking Project* [6]. The Grid project focuses on routing issues

[1]The IEEE 802.11 family of protocols are often referred to as WiFi

in ad-hoc networks. Currently they have two active test-beds. Their first test-bed is a 29-node indoor network hosted in their building, and the second test-bed is a 50-node outdoor network connecting graduate students apartments in the Cambridge area. Of particular interest is their outdoor network which they have called *Roofnet*. As part of the Roofnet project, the group has studied the link-level characteristics of the IEEE 802.11 protocol, and they have also investigated finding high-throughput routes in the event of lossy links. Through Roofnet, users are offered high-speed Internet access via the MIT backbone. The capability exists for users of the network to share their Internet access with the rest of the Roofnet, although this has not been implemented yet. The Grid and Roofnet projects have spawned various other research projects in the field of ad-hoc networks and several papers have been published regarding this work. One such paper describes the suitability of the IEEE 802.11 protocol for a 38-node urban network, in terms of packet loss [7]. Interestingly, they find no clear distinction between working and non-working links. Also, while they find that distance between nodes and signal-to-noise ratio does influence packet losses, the correlation is weak. Through the use of hardware channel emulators they find that the most likely cause of an intermediate level of packet loss is due to multipath fading - a typical phenomenon seen in wireless radio networks.

The Roofnet utilizes the *Click Modular Router* [8]. The Click router is capable of routing 333,000 64-byte packets per second [9]. Click allows system designers to build flexible and configurable high-performance routers, which run either at the user-level or as a kernel module in the Linux operating system. Click router configurations are directed graphs with Click elements at the vertices. Packets flow along the edges of the graph. All the software for this project is open source and custom elements can be written for the router. Various ad-hoc routing protocols have been successfully implemented using Click during the Grid project. Additional information about Click is presented later in this research.

Long-range 802.11 networks, such as the Roofnet, have proved to be fairly successful. Research has shown the viability of such long-range networks and the theoretical distance between nodes can be as much as over 20 km [10]. This research also identifies node collaboration as an integral factor in ensuring the success of ad-hoc networks, as well as justifying the need for ad-hoc network test-beds over simulation-based experimentation. It is shown that while long-range ad-hoc networks are feasible, a number of issues surrounding routing, power management and security in particular must be addressed for such networks to be suitably reliable while provide quality of service guarantees.

A similar project to the Roofnet has been set up in New Zealand by the WAND Research

Group at the University of Waikato. The group has set up a rural community network composed of 9 remote schools and 12 houses [11]. The network, which is set out in a tree structure, is called the CRCNet (Connecting Remote Communities). Like Roofnet, CRCNet, utilizes WiFi technology for its links. The group has been successful in providing high-bandwidth Internet access and video-conferencing facilities to locations that are highly inaccessible. The group is currently researching different link-layer and routing protocols for their network. Their research also demonstrates the viability of long range WiFi networks for providing Internet access. However, this network can not be considered ad-hoc due to the permanent nature of their installations.

2.3 Node Cooperation Enforcement

While research into node cooperation enforcement in ad-hoc networks is still very much in its infancy, there exists some literature that is worthwhile reviewing. Lack of cooperation amongst the entities that comprise multi-hop ad-hoc networks (otherwise referred to as network nodes or agents), may have detrimental effects on network performance [12]. In this section, schemes are reviewed that attempt to promote or reward cooperation, and ensure ad-hoc data networks are feasible for civilian use.

Researchers at the University of Southampton have identified that trust and reputation are central to effective interactions in, what they call, open multi-agent systems [13]. In this case an open multi-agent system is a network composed of a variety of stake-holders that are free to enter or leave the system at any time, or simply choose not to participate in the system. Their studies indicate that the individual components of the system will tend to act in an autonomous, flexible and unpredictable manner. Because the stake-holders generally have different aims and objectives, the following three outcomes emerge:

1. The agents will most probably be unreliable and self-interested.
2. The agents do not know everything about their environment.
3. The agents can not be controlled by a central authority.

For these reasons the researchers propose a trust model that incorporates a variety of sources of trust information, and each node must be able to evaluate the trustworthiness

of other agents for themselves. They also suggest that the agents should be robust against possible lying from other agents due to the fact that they are all inherently self-interested. The authors identify that there are two distinct sources of trust information: from the individual agents and from the whole society of agents. The trust information gained from the society is referred to as a particular node's reputation, and this is based on the past experiences of all agents in the system. In order to evaluate trust in multi-agent systems the authors propose a model that integrates four different types of trust and reputation information. The variety of information sources ensures that if there are uncertainties with regards to a particular trust metric, the agent can rely on the other metrics to validate the trustworthiness of an interacting agent. A novel type of reputation, certified reputation, is proposed for situations where no other trust information is available other than from the interacting agent itself. This certified trust information is based on references from other sources about the agents past performances. While this certified information does allow an agent to gain trust information from an interacting agent without referring to the society, it is highly vulnerable to lying agents, and should probably be avoided unless absolutely necessary.

Researchers from Institut Eurecom in France have developed a collaborative reputation mechanism to enforce node cooperation in mobile ad-hoc networks, known as CORE [14]. This model stimulates node cooperation by means of a collaborative monitoring technique and reputation mechanism. Every agent in the network bases their trust of their individual neighbours with respect to a requested function, and collects information based on observations of the execution of the function. In order to avoid false detection information, such as when there are broken links, the CORE model introduces an aging factor that gives greater relevance to past observations. The model also specifies that no negative ratings can be issued about agents in the network. The result of this is that malicious agents cannot decrease other agent's reputation. In order to punish misbehaving agents, the network gradually isolates them when their reputation falls beneath a predefined threshold value. At this stage, services to these isolated agents is suspended until such time as the agent has increased its reputation sufficiently by participating in network operations.

The effectiveness of the CORE model has been studied using both cooperative and non-cooperative game approaches [15]. Using the cooperative game approach the researchers were able to establish a lower bound on the number of legitimate nodes in a network adopting the CORE model. The non-cooperative game approach describes the behaviour of agents adopting the CORE mechanism and shows that where half an agent's energy is used

to cooperate with other agents, then the strategy selection phase stabilizes asymptotically to a fair position.

Buchegger and Le Boudec [16] propose the CONFIDANT protocol based on theories of selective altruism and utilitarianism[2]. In a similar manner to the cooperation enforcement schemes outlined above, the CONFIDANT protocol aims to isolate misbehaving nodes and thus make misbehaviour unattractive to cheating agents. The protocol is designed to operate with the network layer Dynamic Source Routing protocol which will be described in detail later in this work. A number of components make up the CONFIDANT protocol including the *monitor*, the *reputation system*, the *path manager* and the *trust manager*. These systems allow the agents to monitor the behaviour of their next hop neighbours and detect malicious behaviour. Thresholds can be set differently to satisfy the security requirements of the particular network. If the reputation system establishes that a malicious agent is present then it informs the path manager to scrub all routes containing the misbehaving node from its path cache. To inform others of the misbehaving agent, a message is transmitted by the trust manager that includes relevant information about the violating agent's transgressions. Agent's monitor components receive such messages and pass them to their trust manager component which will then establish how to react to this information based on whether the sender of this information is itself trustworthy. The authors show that the CONFIDANT scheme is able to thwart forwarding and routing attacks in multi-hop ad-hoc networks. It is also shown that the scheme is scalable in terms of the number of agents comprising the network and performs suitably well with as many as 60% of the agents in the network acting maliciously.

Interestingly, the question as whether there is a need at all to stimulate cooperation in multi-hop ad-hoc networks has been researched. German researchers have studied the effects of different approaches that intensify participation in ad-hoc networks [17]. They focus on sparse networks using on-demand routing protocols. Their work shows through simulation, that medium and large scale ad-hoc networks with long average routes are not greatly affected by cooperation enforcement mechanisms. This result is intuitive as dense highly-connected ad-hoc networks will not suffer as much from misbehaving agents because idealistic cooperation schemes isolate such agents. As a result it would be as if there was no node there at all. However, it is shown that for smaller ad-hoc networks with short average path lengths, that cooperation models do succeed in increasing the network's reachability

[2]Utilitarianism is an ethical framework which posits that all action should be directed toward achieving the greatest utility for the greatest number of people

both in terms of providing a tolerable number of intermediate hops, and in providing an acceptable probability of data reaching its destination.

Punishing uncooperative agents in a multi-agent system should result in the agents becoming cooperative or being isolated from the network. In some situations, however, it is better to coax uncooperative nodes through other means. The detection-based approach to agent cooperation relies on the awareness of the agents to the penalty of misbehaviour. Examples of such schemes have been presented above and the common feature of these mechanisms is that knowledgeable agents deny services to misbehaving agents. There are certain unresolved issues surrounding reputation-based systems. One such issue is that there is no formal specification as to what incentives are offered to agents that choose to act in an altruistic manner. It has also not been considered that the selfish nodes might choose to cooperate amongst themselves, hence maximizing their own welfare at the expense of the rest of the network. In addition, detection-based systems often rely on the broadcast nature of wireless networks to detect selfish nodes. This may be a problem if there are asymmetric links (due to nodes using power control), and because some networks may utilise directional antennas making the interception of traffic physically impossible. Therefore, other methods of ensuring node cooperation have been proposed that rely on a motivation-based approach, where agents receive rewards for their relative unselfishness.

Buttyan and Hubaux [18] show that a large percentage of power in ad-hoc networks is dedicated to forwarding the traffic of others. As a result the battery lifetime of a particular node could be dramatically extended by a refusal to execute their forwarding duty. They have found that when nodes are under the control of end-users, there is a strong temptation to alter the nodes, most probably to gain maximum utility from the network. Ensuring that all nodes in an ad-hoc network are completely tamper-resistant would be near impossible, so they propose that a single hardware module on the node be tamper resistant. This tamper-proof hardware is known as the *security module* and is similar to SIM cards in GSM phones which perform similar functions. In their design, every packet that passes through a particular node, either generated by the node or received for forwarding, is passed to the security module. This module maintains a counter which they term a *nuglet counter*. The counter is decreased when the node originates packets and is increased when the node forwards packets for others. In order for a node to send packets, it must have a positive nuglet value. Otherwise it must forward packets for the benefit of others to increase its nuglet value sufficiently to send packets. The aim of this scheme is to promote users of the ad-hoc network to keep their nodes switched on (to forward other's traffic)

and to refrain from sending large volumes of traffic to distant destinations. The scheme is based on the following two simple rules:

1. If a node wishes to send a packet to a particular destination it first must estimate the number of hops required to get there. If its nuglet counter is greater or equal to this value the packet is sent and the nuglet counter reduced by this estimated figure. If the node does not have sufficient nuglets the packet is not sent and the counter remains unchanged.

2. Every packet that is successfully forwarded by the node increases the nodes nuglet value by one.

The authors of the nuglet scheme discuss the relative utility that nodes will achieve if they adopt different rules for dropping packets. Clearly nodes should always send packets if they have sufficient nuglets, but at some point it would increase the nodes individual benefit to drop packets when it feels that it has enough nuglets for its own purposes. This is especially true if the node only has sufficient energy left to send its remaining packets. An interesting result observed from their experiments is that a node that acts in a highly cooperative manner, i.e. forwards all packets, can be more energy efficient in the long term. This is because of the random manner in which packets arrive. A node that only maintains a store of just enough nuglets, will find that a situation occurs when it does not receive enough forwarding packets to cover the cost of sending its own packets.

Moreover, the authors have found that with a large percentage of uncooperative nodes, the network's average throughput is reduced, but not by a significant amount. It appears that the nuglet scheme outperforms their two previous payment methods which were the *Packet Purse Model* and the *Packet Trade Model* [19]. The Packet Trade Model allows intermediate nodes to buy a packet from a previous hop and sell it off for more nuglets to the next hop. Unfortunately this leads to overloading of the network since the packet sources never have to pay for the services of other nodes. In the Packet Purse Model the source loads a number of nuglets into the packet before sending it. Intermediate nodes will then acquire nuglets from the packet as payment if they forward it, but if the packet runs out of nuglets it is dropped. The problem with this model is that if a source underestimates the number of nuglets required for the packet to reach its destination, it will be dropped and the source will subsequently lose its investment. In addition, if the source over-provisions for the number of nuglets required for the packet to reach its destination, it might end

up paying too much for the service it requires. Furthermore, if a packet is lost due to a network fault, the nuglets contained in the packet will be lost. This results in the total number of nuglets in the system decreasing rapidly, to a point where there are not enough nuglets for any transactions.

The virtual currency system appears to meet its goals in stimulating the forwarding of packets, encouraging nodes to remain powered up even when they have no traffic to send and discourage sending large amounts of traffic over long distances. However, there are a number of issues that have not been resolved. The concept of a security module is problematic in an open multi-agent system. These modules will add cost and complexity to network nodes, thereby discouraging the use of the network. Another issue with virtual currencies is that one cannot be sure of what network topology will be present. In highly mobile systems that form a tight mesh structure there should be no problems acquiring enough currency to forward one's own packets. However, in a system where nodes are not highly mobile, and certain nodes find themselves on the border of the network, there will be very little opportunity to forward other node's traffic and hence acquire sufficient nuglets to survive. Moreover, the cost of sending a small packet and a very large packet are the same and the reward for forwarding a small packet is the same as forwarding a large packet. As a result, it may be found that nodes tend to be selective as to which packets they decide to forward based on the length of the packet. Eventually network nodes would only be able to transmit very small amounts of data or risk having their packets dropped.

Researchers from Yale introduce a novel method of stimulating cooperation [20] similar to that of the Buttyan and Hubaux method, except without the need for a tamper-proof hardware module. They refer to the model as: A Simple Cheat-Proof, Credit-Based System for Mobile Ad-Hoc Networks (Sprite), and it works in a fairly simple manner. When a message is received at a node, the node keeps a receipt of that message. Later when the node obtains a connection to a Credit Clearance Service (CCS) it reports all the messages that it received and forwarded by means of the receipts. The CCS will then determine how much to charge and reward the individual nodes involved with the transmission. What is interesting about this scheme is that it relies on human being's minor motive of earning money or credits by promising personal monetary gain if the node exhibits correct routing behaviour.

In the Sprite model there are two ways of obtaining more credit in the system - either by purchasing the credit at a variable rate based on the current performance of the system or, preferably, by forwarding other's traffic. It is envisioned that each node will require at

least two network interfaces. One network interface connects to the ad-hoc network using either a WiFi or Bluetooth connection and the other interface will connect to the CCS via GPRS or some other long-range communication protocol. Receipts are saved on the node using local storage and forwarded to the CCS when it has enough power. Alternatively the node can hand over its receipts when it is in range of a proxy desktop computer that can forward the receipt. The receipts themselves are derived from the content of the messages but do not contain the exact content of the messages. This maintains privacy and means that the CCS only needs to be trusted to maintain the correct credit balances of the nodes.

The Sprite system suggests that the best charging scheme is to charge only the sender of the message. This solves the issue of denial of service attacks, whereby nodes may send many messages to a recipient who would then be expected to pay for them and possibly run out of credit. They propose that if the recipient of a message gains utility from the message then they should reimburse the sender later through some form of application-layer payment protocol. To ensure that only nodes that have legitimately forwarded traffic are rewarded, the CCS only provides payment if there is a successor of that node on the path reporting a valid receipt of the message. The total payment of forwarding traffic is always less than the cost of sending the traffic. This eliminates collusion between cheating nodes. Periodically the CCS distributes a fixed amount of credit to every node in the network in order to avoid the net outflow of credit in the system. This periodic reward system does not benefit selfish nodes and does not reduce the incentive for nodes to act in a non-selfish manner. The authors of Sprite propose a number of solutions to issues regarding selfish nodes colluding to maximise their utility of the network [20].

2.4 Competition and Collusion

Bolton and Ockenfels propose a model to describe an individual's desire to not only gain maximum utility from a game but also to have a high relative payoff to the other players [21]. The *Equity, Reciprocity and Competition model* (ERC) describes how the motive to be relatively superior to the other players can organise a large and seemingly disparate group of experiments into a consistent pattern of behaviour. This model dictates that a motivational function, i.e. a function that might cause an individual to alter their strategy in a game, is made up of a payoff function and a concern for relative standing in the game. When analysing a cooperation enforcement scheme it is useful to introduce the ERC model to describe the tendency of nodes to cooperate when typical models might suggest that

they would not. Because ERC explains stable patterns for relatively simple games played over a short time span [21] it is especially useful for analysing the prisoner's dilemma game introduced in the next chapter. ERC can be applied to the nodes of an ad-hoc network, in order to model nodes' preference structures and predict their behaviour.

Nguyen and Jennings have formulated a heuristic model for managing concurrent negotiations in time-constrained settings where agents have no prior knowledge of their opponents [22]. This situation is found in ad-hoc networks, and can be studied using such models. The authors identify drawbacks to current bi-lateral negotiation models where an agent must identify a trade partner without *a priori* knowledge. However, if there are multiple agents that can provide similar services, such as the forwarding of a nodes traffic or routing information, then the agent has the choice to either negotiate sequentially or concurrently with each service provider. Sequential negotiations simplify the negotiation process, as the outcome of one negotiation can be used to predict the future behaviour of the agents involved. However, it results in lengthy negotiation encounters. The authors develop a bidding model for concurrent negotiations in which the threads of the various negotiations mutually influence each other. The relative progress of one negotiation may affect an agent's stance in another negotiation. Thus a node of an ad-hoc network may find that because a negotiation is going well with one neighbour it may adopt a tougher stance with another neighbour, and try to extract an even better deal. We do not review the specifics of the model but the authors have found, through an empirical evaluation, that their model leads to better deals more quickly than a similar sequential model. This is useful in networks where delay bounds exist, for example if there are quality of service constraints. This model can be used in ad-hoc networks to describe bargaining with neighbouring nodes for services required. Bargaining is not discussed in detail in this research, but is a field of interest for future work in cooperation enforcement.

2.5 Discussion

In this chapter, work has been identified that is closely related to the topic of this research. Several ad-hoc network infrastructures have been examined, and this has shown the popularity of the IEEE 802.11 protocols in this type of research.

A number of cooperation enforcement models have been reviewed. In general, the authors of papers regarding such models agree that without some form of security mechanism to

combat node selfishness the ad-hoc networks will likely fail. A common feature of all approaches is to discourage selfishness by offering incentives to cooperative nodes, or by punishing selfish nodes.

In the section on competition and collusion two models were reviewed. Firstly the ERC model that, although not intended for this particular purpose, describes the behaviour of ad-hoc network nodes very well. A model for bi-lateral trade negotiations has also been reviewed, that could enhance cooperation enforcement mechanisms by facilitating fast negotiations with network nodes for service requirements.

Chapter 3

Trust Based Cooperation

3.1 Introduction

In the previous chapter, several state of the art cooperation enforcement techniques applicable to ad-hoc wireless networks were reviewed. While a number of cooperation enforcement schemes have been proposed in the literature, almost none have been implemented in practical test-beds. Furthermore, there are currently few results indicating the degree to which these schemes achieve their objectives. In this chapter a novel cooperation enforcement model is proposed, building on work published by other authors. This model is reviewed analytically, illustrating that such a model can entice cooperation in mobile ad-hoc networks.

3.2 The Need for Cooperation Enforcement

Open multi-agent systems are likely to be abused by misbehaving nodes due to the scarceness of resources available. It has been shown that the only logical behaviour of nodes is to maximise their own utility [23]. Below it is shown that without some form of cooperation enforcement scheme, civilian ad-hoc networks will ultimately fail, and are thus unlikely to gain widespread favour.

3.2.1 Routing

The goal of routing protocols is to establish an efficient route between source and destination nodes so that they may communicate in a timely and reliable manner. Any routing protocol used in an ad-hoc network must be distributed amongst the nodes of the network. This is because, by definition, an ad-hoc network is one with no existing infrastructure and no node assumes control over any other node.

Due to unpredictable topology changes, routing protocols designed for wired networks are not suitable for mobile ad-hoc networks. Thus a number of routing protocols have been proposed for use within wireless environments. These can be divided into two sets, namely *table-driven* and *on-demand*. A common characteristic of ad-hoc routing protocols from either set is that each node acts as a router and takes part in both the discovery and maintenance of the routes [24].

In table-driven routing protocols, every node manages tables containing routing information to every other node in the network. When the network topology changes for any reason, it is the responsibility of the nodes to propagate this information throughout the network, so that all nodes can update their tables accordingly. Thus all nodes always have a consistent and up-to-date view of the network.

However, if the network were to contain uncooperative nodes that do not propagate this information, the routing tables of downstream nodes could contain stale information. This may cause other nodes to waste their own resources by sending information down routes that no longer exist, only to find that the information is lost. Another unfortunate result is that the lost packets will cause the sender to issue updated routing information (informing other nodes of the missing links), and may add to network congestion.

On-demand routing protocols do not rely on tables to maintain up-to-date routing information. Rather on-demand routing protocols wait until there is information to send, at which point a sending node will try to establish a route to the destination. This is performed by some form of route discovery, usually achieved by flooding the network, and determining a shortest path to the destination according to the replies received. In order to avoid having to flood the network unneccesarily, each node must maintain a route cache that is used in future communications to various destination nodes. If a message is sent, and is found to have been lost in transit, the sending node may perform some form of route maintenance. This leads to the discovery of a new route to the destination.

The effects of uncooperative nodes can be highly detrimental to on-demand networks. It may be in a selfish node's favour to discard route request messages, because this will ensure that it is not part of the final route to the destination node. The selfish node will conserve its own resources, but create additional work for its surrounding nodes. This will also most likely result in a sub-optimum route to the destination being found, hence reducing the overall efficiency of the network. If a particular node is selectively selfish, this could also result in additional work for the network as a whole. If the selfish node cooperates with the initial route discovery only to drop packets later then the sending node will again have to flood the network to find a new route. For this reason the increase in traffic load on the network due to a single uncooperative node can be substantial. The effects of selfish nodes in an ad-hoc environment have been simulated in the literature and the results of this simulation show that a small percentage of selfish nodes can lead to severe degradation of the overall network performance [25].

3.2.2 Traffic Forwarding

In this work it is assumed that all nodes in an ad-hoc network, whether selfish or unselfish, cooperate with all routing related requests and forward any topology changes to their neighbours. This is a logical assumption because it is also assumed that the reason the node is being selfish is not because it is maliciously trying to sabotage the network but rather because the node is simply trying to conserve its own resources. It would be unwise for a node to adversely affect the routing process in the network because, as it has already shown, the entire network becomes unreliable and there is additional load on all nodes. This contradicts the goals of a node that is trying to save its own resources. Essentially, in this case the selfish node will have to perform more work to send its own traffic. If the marginal benefits of reduced network load and increased reliability for the selfish node exceed the marginal benefits of dropping routing related traffic, it is in the nodes best interests to be cooperative in this function.

On the other hand, logical nodes will still try to maintain their own resources by dropping other traffic. If there is no incentive to forward the traffic of others then it makes sense for a forwarding node to discard traffic immediately and not to waste its processing resources or battery power. If the node chooses to be randomly selfish then the result is that two corresponding nodes with a selfish intermediate node will experience a highly lossy link. These corresponding nodes will waste their own resources significantly, and the link will

almost certainly experience reduced throughput.

3.3 A Novel Cooperation Enforcement Model

Based on the findings of the literature review, it was decided that a simple distributed trust mechanism should be developed. Similar existing models are very difficult to implement on an evaluation framework. Thus a simple trust and reputation model for cooperation enforcement is proposed as part of this research. The model prevents nodes from manipulating each other either by lying or through collusion, which would satisfy their own selfish interests [26]. The proposed model is similar to the Collaborative Reputation Mechanism by Michiardi and Molva [14], in that it also relies on various forms of trust and reputation information.

Due to the fact that nodes in an ad-hoc network have incomplete knowledge of other nodes, they must rely on trust when interacting with each other. Trust has been defined as a measurable level of the subjective probability with which a node assesses that another node will perform a particular action. This is both before the node can monitor such action and in a context in which it affects its own action [13]. In an ad-hoc network the nodes can not rely on a centralised authority to gain trust information. It can only ascertain the trustworthiness of other nodes either by interactions with the individual node under consideration or at a society level. If it has never had an interaction with a particular node, or if the number of interactions with the node do not represent a significant enough sample size, then the corresponding node must look elsewhere for trust information. From this it is clear that various forms of trust information are required to get an accurate measure of the cooperativeness of a node in the network. Thus four types of trust are defined: *Interactive Trust*, *Non-Interactive Trust*, *Rumoured Trust* and *Bragging*.

The goal of this scheme is to establish how much an interested node can trust a suspect node to act in a cooperative manner. If a suspect node is found to be cooperative, and forwards all packets on behalf of corresponding nodes, then it is given a good trust rating and is included in future routing decisions. On the other hand, if it is found that a suspect node is not cooperative it will be given a poor trust rating. The rest of the network punishes the suspect node by refusing to provide services to it, and gradually excludes the suspect node from the network all together.

The model, implemented on every network node, performs the following tasks:

- Identifies network nodes that are not acting in a cooperative manner;
- Punishes the uncooperative nodes by denying them network services (fail to forward traffic for them and exclude them from any routing decisions);
- Informs selfish nodes that they are being punished, thus enticing them to become cooperative again or leave the network;
- Scrubs routes containing selfish nodes from the route cache, in order to increase the link throughput and reliability between corresponding nodes.

In this work the model is to be implemented on a small evaluation framework, and therefore the model is designed to be as adaptable as possible, but its implementation is not very scalable. The model facilitates a large number of nodes but the implementation relies on a flooding mechanism. Hence the implementation of the model in this particular work is only applicable to relatively small networks.

3.3.1 Gathering Trust Information

In order to gather information about a suspect node a request is sent to all other nodes in the network. This request and the corresponding replies are flooded throughout the network. Because the requests and replies are flooded throughout the network, all the nodes can offer their information about a suspect node, and can also view information about the suspect node sent by other nodes. In this way every node establishes a trust rating for every other node in the network very quickly.

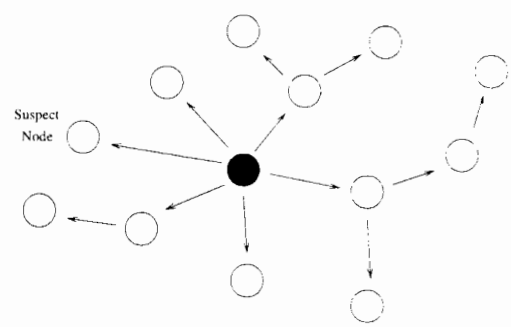

Figure 3.1: Flooding an information request packet throughout the network.

In Figure 3.1 the source node (black) floods an information request packet throughout the network. The nodes then reply (shown in Figure 3.2) with their respective trust information detailed in the sections below.

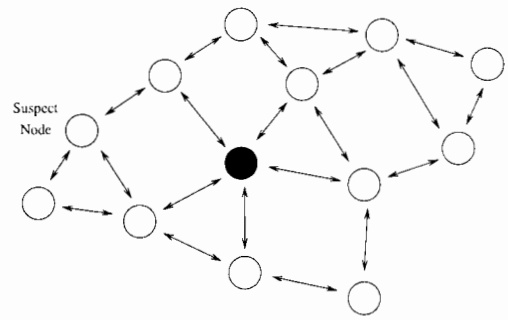

Figure 3.2: The nodes reply with their trust information which is also flooded throughout the network.

Flooding information throughout the network does increase the traffic burden on the network. Thus it also diminishes the resources of the nodes that we are trying to protect. Unfortunately, it will be shown later in this chapter that without such a mechanism ad-hoc networks may fail and thus this increased traffic burden is necessary. However, the trust information packets are insignificantly small compared with data and routing requests, and may be piggy-backed on these packets for further savings. Therefore this cooperation enforcement scheme will not place undue load on the network.

3.3.2 Interactive Trust

The most reliable evidence of a particular node's cooperativeness is gathered by the last k direct observations of that node. If a node has had previous experiences with a suspect node (e.g. the suspect node forms part of one of its routes), this is called a direct observation. Interactive trust is only gathered by direct observations, and is not based on other's opinions. Thus no information request packets are required. If a node has direct correspondence with another node and finds it to be uncooperative it must rate this information with the highest priority. Consider a situation where a node interacts directly with a suspect node n. In a particular interaction i the node assigns a rating x based on personal observations which results in the three-tuple $r_i = (n, i, x)$.

For the sake of simplicity the values of x are normalised to the range $[-1, 1]$, where -1

indicates the suspect node is completely uncooperative, and 1 indicates that the suspect node is fully cooperative. A rating of 0 indicates that no rating has been given to the suspect node yet. The weights are adjusted according to the relevance to be placed on past interactive trust information. More weight w is given to more recent ratings. The interactive trust T_I is calculated as:

$$T_I(n) = \sum_{i=0}^{k} w(r_i).x_i \tag{3.1}$$

However there are often situations, particularly when devices are frequently entering or leaving a network, where a node will not have had many direct interactions with a suspect node and as such would have to rely on other sources of information.

3.3.3 Non-interactive Trust

When there is insufficient previous direct correspondence with a suspect node, then the community's experience must be relied upon. At the request of a node the community shares their respective interactive trust information. This is achieved by flooding the network with an information request packet (illustrated in Figure 3.1 and Figure 3.2), and waiting for the replies. This form of trust can be valuable if a sufficient number of nodes have reliable information as to whether a suspect node is cooperative or not. However, if a number of selfish nodes collude then this form of trust could be subject to cheating and a distorted view of the suspect node could be presented. A sending node requests information about a suspect intermediate node. Each node, p, in the N-node network that responds to the request, replies with their interactive trust of the suspect node n, in the form of the three-tuple $r_p = (p, n, x)$. Once received, this new information is added to the existing non-interactive trust information T_N*. Again recent information is weighted more than older information. The non-interactive trust is calculated as follows:

$$T_N(n) = \sum_{p=0}^{N} w(r_p).x_p + w(T_N*).T_N* \tag{3.2}$$

3.3.4 Rumoured Trust

Periodically a node may find that a neighbour has become particularly uncooperative, or has begun acting cooperatively again. In this case it is unwise not to inform the other

nodes about this new behaviour. The neighbours are informed via a rumour and this rumour is propagated throughout the network, via the flooding mechanism. It would not be beneficial for cheating nodes to spread incorrect rumours about others as the nodes are assumed to be purely self-interested. Of course this assumes that selfish nodes are not colluding. Nodes may not start rumours about themselves. Received rumours take the form $r_n = (p, n, x)$. The rumours are added to the existing rumour information T_R*. The rumoured trust calculation is similar to the non-interactive trust calculation,

$$T_R(n) = \sum_{p=0}^{N} w(r_p).x_p + w(T_R*).T_R* \tag{3.3}$$

Essentially rumoured trust is the same as non-interactive trust, except that rumoured trust is not solicited. Any node may offer a rumour at any stage in order to either

- Allow the rest of the network to punish an uncooperative node by decreasing its trust rating and denying it network services, or
- Reward a newly cooperative node by increasing its trust rating, and restoring its access to network services.

3.3.5 Bragging

If it is found that the amount of trust information from the above sources is insufficient to make a judgement about a particular node's cooperativeness, then the node resorts to one final form of trust known as bragging. Bragging refers to a suspect node, n, claiming a certain amount of cooperativeness and takes the form $r_b = (n, x)$. Only the most recent bragging information is used.

$$T_B(n) = x_b \tag{3.4}$$

Clearly there is an incentive for cheating nodes to lie about themselves. To prevent such a situation occurring, the model dictates that if any node overhears bragging that it determines to be a blatant lie, then it will inform the source node that the information is not to be trusted, via a rumour. Thus bragging packets are also flooded throughout the network. If the source node receives enough reports that the suspect node has lied then

it can conclude that the suspect node is not trustworthy, and will punish it severely. This scheme provides enough incentive for nodes to be at least partially honest about their trustworthiness, however, this trust information should only be used as a last resort.

3.3.6 Composite Trust Measure

Every node contains a weighting factors set for each of the various forms of trust $W_j = \{W_{1..4}\}$ (the interactive trust, non-interactive trust, rumoured trust and bragging respectively). These weighting factors are dynamically adjusted over time according to how much information is available from the various forms of trust. This yields an overall composite *trust rating*

$$T(n) = \frac{W_1.T_I + W_2.T_N + W_3.T_R + W_4.T_B}{\sum_{j=1}^{4} W_j} \tag{3.5}$$

3.3.7 Cooperation Enforcement Model Discussion

The proposed model is simple, but highly flexible as it allows for nodes in different environments to adapt to changing conditions. For the model to be successful all the weighting factors must be adjusted to account for differing amounts of information and the doubtful reliability of some information. Ideally some form of learning algorithm should be adopted for this purpose. Once trained the algorithm would be able to assign appropriate weights based on previous experience. If it is found for example that agents in the system change their cooperation strategies rapidly, then substantially more weight should be placed on recent information. After a suitable number of interactions with a suspect node $W_1 \gg W_4$, i.e. interactive trust should be valued much higher than bragging. Whereas if an agent joins a system of which is has absolutely no knowledge, then it should place very little weight on its interactive trust rating of a suspect node.

The model aims to provide payoffs to nodes in the form of trust ratings. A new node in the network will strive to achieve a good trust rating from the community so that it has full access to network services. This good rating can be considered a positive payoff for the node. The node rates this payoff similarly to having ample resources of its own. This is because without a good trust rating a node can not enjoy the services of the network, regardless of how much battery power it has remaining. If nodes see that a previously

uncooperative node has become cooperative again (forwarding flooded packets, etc.), then its trust rating will be increased, and it will once again gain access to network services. In the next sections an analysis is performed on the ambition for nodes to achieve a balance between a good trust rating, and conserving their own resources.

3.4 Game Theoretic Analysis of Stimulated Cooperation

The effects that can be expected from the introduction of the above model into an open multi-agent system are now reviewed. Rational nodes of an ad-hoc network, where no cooperation incentives exist, will eventually choose to be selfish to the detriment of the entire network. For a cooperation enforcement scheme to be successful, it must ensure that the strategy of the individual nodes is to cooperate even though there are implications in the form of wasted resources. A node that does not cooperate faces being excluded from the network altogether. An effective method of describing this situation was introduced by Albert Tucker in 1968, which he called the *Prisoner's Dilemma* [27].

3.4.1 The Prisoner's Dilemma

The Prisoner's Dilemma (PD) shows the difficulty of analysing non-zero-sum games - games in which one competitor's victory is not necessarily the other competitor's defeat. The dilemma depicts two partners involved in a crime who have been arrested under the strong suspicion they are guilty. Both criminals, who have been kept separate, have a choice to make: they can either confess (defect), thereby implicating the other criminal or keep quiet and cooperate with each other. If both confess, then they will serve an intermediate jail sentence. If both keep quiet then they will each serve a short jail sentence, because there is insufficient evidence. If the one partner confesses while the other keeps quiet then the criminal who confessed goes free while the criminal who keeps quiet serves a very long jail sentence.

The PD is a strategy game in which each player must try to anticipate the behaviour of its opponents and adjust their strategy accordingly. In terms of an ad-hoc network the prisoner that confesses is analogous to a node that defects from its network duties, and a prisoner who keeps quiet is analogous to a node that cooperates and forwards the

traffic of the other nodes. The utility gained from the different games is represented by a payoff matrix, shown in Table 3.1. δ_j is always non-negative, and can vary depending on the current resources available to the node. The payoff matrix shows that a node gets maximum payoff when it defects and the other nodes continue to cooperate. Thus the node gets a free-ride, and the result is that the other nodes must bear the burden of this selfishness.

Table 3.1: The PD payoff matrix.

	Cooperate	Defect
Cooperate	(u, u)	$(u - \delta_1 - \delta_3, u + \delta_2)$
Defect	$(u + \delta_2, u - \delta_1 - \delta_3)$	$(u - \delta_1, u - \delta_1)$

The PD is also found in economics literature as *Bertrand and Cournot games* which refer to oligopolistic[1] markets [28]. If the game is repeated many times it is known as an *Iterated Prisoner's Dilemma.*

3.4.2 Applying the Prisoner's Dilemma to Ad-hoc Networks

The analogy of the PD can be applied to ad-hoc networks. In the PD it is in both criminals' best interest to cooperate and thus both get relatively short jail sentences. In ad-hoc networks it is also in all nodes' best interest to cooperate, and in doing so, offer a percentage of resources for forwarding others' traffic. But in both the PD and in ad-hoc networks there is always the incentive to cheat (by being selfish) and gain greater payoff. Thus the PD offers a very simple model of the behaviour of nodes in an ad-hoc network.

The cooperation enforcement model attempts to preempt another node's selfishness, and thus make selfishness less attractive. Say, for example, a suspect node, X, is surrounded by a number of other nodes, depicted in Figure 3.3. Every node is separately interested in how selfish X is. If the suspect node becomes very selfish, and defects most of the time, then every other node must have a mechanism to discover this and to punish it. The nodes do this by sharing information amongst themselves about the behaviour of the suspect node, as described by the cooperation enforcement model. In terms of the PD it would

[1] A market condition in which sellers are so few that the actions of any one of them will have measurable impact on competitors.

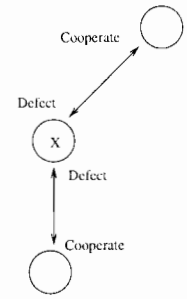

Figure 3.3: A selfish node interacting with two cooperative nodes.

be as if there were a third criminal in one of the cells, who could tell one of the first two criminals how the other usually acts. Thus one of the criminals could have a good idea of what the other's decision will be (to cooperate of defect) before that decision is made.

In Chapter 5, a PD game is set up in order to examine how well a cooperation enforcement scheme can successfully identify selfish nodes, and preempt their selfishness.

3.4.3 Nash Equilibrium

If both criminals in the Prisoner's Dilemma act in a rational manner, then they will both confess. This is because each criminal reasons that if he confesses, and the other criminal confesses, he will receive an intermediate sentence, and if he confesses and the other keeps quiet, he will walk free. Whereas if he keeps quiet and the other confesses he will serve a long sentence. If they both confess he will receive a short sentence. If both players trusted each other enough they each could have had an individual payoff of u, and a overall payoff of $2u$. However, because they both confess their individual payoff is only $u - \delta_1$ and the overall payoff is only $2u - 2\delta_1$.

In terms of an ad-hoc network this means that the nodes, by attempting to gain greater payoff, end up sabotaging the network. In addition, if all nodes decide to defect (be uncooperative), then the whole network will fail. This is because all nodes will be generating traffic but none will be forwarding it on.

John Nash, a Noble-prize winning economist and mathematician, called the tendency for both prisoners in the PD to confess the *Nash Equilibrium* point. By definition a Nash Equilibrium is a list of strategies, one for each player, where that no player can unilaterally change his strategy and get a better payoff [29]. What the Nash Equilibrium makes clear

is that in an iterated prisoners dilemma game, the cooperative outcome (both criminals keep quiet) is unstable in ways that make cooperation difficult to maintain as it is not in equilibrium [30]. The goal of a cooperation enforcement scheme, such as the one proposed in this study, is to shift the equilibrium point so that the outcome is *socially desirable*. A socially desirable outcome is presented later in this chapter.

Typically, implementation theory dictates that a game should have a *Pareto efficient*[2] outcome. The outcome of a game is Pareto efficient if there is no other outcome that makes every other player at least as well off and at least one player strictly better off [31]. In the case of the Prisoner's Dilemma, every outcome is Pareto efficient except at the Nash equilibrium where both prisoners confess. Thus the proposed cooperation enforcement model attempts to avoid Pareto inefficient outcomes either by excluding uncooperative nodes (thus increasing the weighted sum of every node's utility from the network), or by inciting uncooperative nodes to change their strategy.

3.4.4 An *N*-Node Prisoner's Dilemma

An ad-hoc network can be analysed as a prisoner's dilemma game, using a revised version of the game theoretical approach by Michiardi and Molva [15]. This approach provides an analytical evaluation of the effects of cooperation enforcement, while also highlighting the payoffs that nodes perceive.

If an ad-hoc network of N nodes exists, and all the nodes implement a cooperation enforcement scheme, it must be determined how many nodes, if any, choose to play cooperatively. Such networks exist in the form of community ad-hoc networks.

If the total number of cooperating nodes in the network is k then the payoff (the benefits of using the network) for a defecting node is given by $D(k)$, shown in Figure 3.4. If instead the defecting node chooses to play cooperatively then clearly the payoff is less, as it must sacrifice its own resources, and the cost of this is denoted by $C(k)$. As such the total payoff p for a cooperating node n is given by

$$p_n = D(k) - C(k)$$

[2]Also known as Pareto Optimal. Vilfredo Pareto argued that an individual's preferences where the beginning point of economic analysis and that ordinal payoffs rate more importantly than cardinal payoffs [31].

If $p_n < D(k+1) - C(k+1)$, then cooperation becomes more attractive, as the payoff increases according to the increase in k.

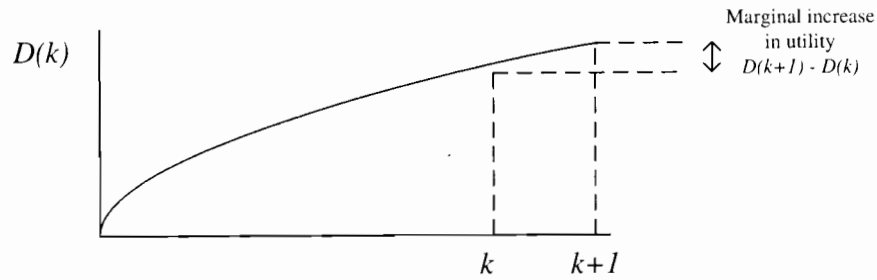

Figure 3.4: The curve of *D(k)* is strictly increasing and concave down.

It is known that cooperating is not as lucrative as cheating and thus,

$$D(k+1) - C(k+1) < D(k)$$

The total payoff for all the nodes of the network can be represented by

$$p_T = N.D(k) - k.C(k)$$

If a particular node chooses to change its strategy to cooperate, then our total payoff would be

$$p_N = N.D(k+1) - (k+1).C(k+1)$$

It is assumed that a *socially desirable* situation occurs when $p_N > p_T$ [15]. At this point all cooperating nodes benefit from a node choosing to play cooperatively.

3.4.5 Applying the ERC Model to Motivational Payoffs

A short summary of the ERC model [21], describing the tendency of nodes to be motivated to cooperate by both absolute and relative payoffs, was presented in the literature review. This model is applied to the nodes in the ad-hoc network. Every node is motivated to

cooperate in the network by gaining utility from the network, obtaining a good trust rating from other nodes, and from gaining similar utility from the network relative to the other nodes. The nodes act to maximise their motivational function v_i, which is given by

$$v_i = v_i(p_i, \sigma_i)$$

where σ_i is i's share of the total network payoff c.

$$\sigma_i = \sigma_i(c, p_i) = \begin{cases} p_i/c, if & c > 0 \\ 1/N, if & c = 0 \end{cases}$$

and

$$c = \sum_{j=1}^{N} p_j$$

Note that if node i has the choice $v_i(p_1, \sigma_1) = v_i(p_2, \sigma_2)$, and $p_1 > p_2$, then node i chooses $v_i(p_1, \sigma_1)$, choosing maximum payoff over relative gains.

Using the ERC model a node's motivational function to cooperate can be calculated,

$$v_i = \alpha_i u(p_i) - \frac{\beta_i}{2}(\sigma_i - \frac{1}{N})^2 \tag{3.6}$$

Where $u(p_i)$ is the nodes utility function of the payoff received, and is strictly increasing and concave down. α_i and β_i are non-negative weights. A reasonable utility function might look like $u(p_i) = \sqrt{p_i}$. The marginal rate of substitution between payoff and relative benefits is given by the ratio $\frac{\alpha}{\beta}$. If the node is primarily interested in its payoffs relative to the others then $\alpha \to 0$, whereas if the node is highly self-interested, or selfish, then $\beta \to 0$.

It is necessary to analyse what would cause a node to play cooperatively rather than selfishly. A node's objective is to maximise its motivational function. The motivational function when a node cooperates is

$$m_c = \alpha_i u[D(k+1) - C(k+1)] - \frac{\beta_i}{2}\left[\frac{D(k+1) - C(k+1)}{N.D(k+1) - (k+1)C(k+1)} - \frac{1}{N}\right]^2$$

However, when the node defects, its motivational function is

$$m_d = \alpha_i u[D(k)] - \frac{\beta_i}{2}\left[\frac{D(k)}{N.D(k) - k.C(k)} - \frac{1}{N}\right]^2$$

The nodes act rationally, therefore they will choose to cooperate if the motivational function has a greater or equal value when they cooperate, i.e. when $m_c \geq m_d$. Which can be written as

$$\alpha_i(u[D(k+1)-C(k+1)]-u[D(k)]) \geq \frac{\beta_i}{2}\left[\left(\frac{D(k+1-C(k+1)}{N.D(k+1)-(k+1)C(k+1)} - \frac{1}{N}\right)^2 - \left(\frac{D(k)}{N.D(k)-k.C(k)} - \frac{1}{N}\right)^2\right]$$

Because $D(k+1) - C(k+1) < D(k)$, and $u()$ is a strictly increasing function, $u[D(k+1) - C(k+1)] - u[D(k)] < 0$, therefore

$$\frac{\alpha_i}{\beta_i} \geq \frac{1}{2}\frac{\left(\frac{D(k)}{N.D(k)-k.C(k)} - \frac{1}{N}\right)^2 - \left(\frac{D(k+1)-C(k+1)}{N.D(k+1)-(k+1)C(k+1)} - \frac{1}{N}\right)^2}{u[D(k+1) - C(k+1)] - u[D(K)]}$$

It is known that $\frac{\alpha_i}{\beta_i} \geq 0$. Therefore if the right-hand side of the inequality is less than or equal to zero the inequality always holds. For the right-hand side to be less than zero, either the numerator or denominator (but not both) must be negative. From above it can be seen the that denominator must be negative. Now all that remains is to find when the numerator is positive or zero. This occurs when

$$\left(\frac{D(k+1) - C(k+1)}{N.D(k+1) - (k+1)C(k+1)} - \frac{1}{N}\right)^2 \leq \left(\frac{D(k)}{N.D(k) - k.C(k)} - \frac{1}{N}\right)^2 \qquad (3.7)$$

Thus if there are any cooperating nodes when the network is in equilibrium and they are over-compensated for the loss in absolute gain by approaching the average gain, $\sigma_i \to \frac{1}{N}$, then there will be cooperating nodes. Furthermore, it can be shown that at equilibrium the number of coalition nodes is at least $\frac{N}{2}$ [15].

3.5 Chapter Discussion

In this chapter the need for some form of cooperation enforcement scheme in ad-hoc networks has been demonstrated. In order for such networks to support efficient and reliable routing and data transfer then certain assurances must be made that the nodes in the network will cooperate to the best of their resources.

A novel cooperation scheme that relies on various forms of trust information has been proposed. If a node gains a bad reputation for being untrustworthy then it will be excluded from the network, and so it is in the best interests of all rational nodes (even malicious ones) to maintain their reputation. This can only be done by participating in the network functions.

The interaction between the nodes in the network has been modelled as a non-zero sum game, and the prisoner's dilemma has been used as an example. The various forms of payoff and equilibria have been discussed, and it is clear that if the nodes were left to their own devices ad-hoc networks would fail.

To determine if there is an equilibrium point where nodes do cooperate a mathematical revue of the network was performed. The ERC model was used to predict the behaviour of the nodes. According to the model, nodes are not only interested purely in absolute gains but also in gains relative to the other nodes in the network. Hence the need for nodes to be equivalent in terms of the payoffs they receive.

In the model review, the conclusion was reached that if a cooperation enforcement model can increase a node's payoff relative to the entire network's payoff, then the node will be inclined to cooperate, and the model will achieve its goal.

Chapter 4

Architecture and Implementation of an Evaluation Framework

4.1 Introduction

It is an aim of this research to investigate the effects of selfish nodes in practical ad-hoc networks. In addition, the effectiveness of a cooperation enforcement scheme, for identifying selfish nodes and establishing high-throughput routes, will also be evaluated. Thus it was necessary to design a test-bed architecture that was able to perform these evaluations. This chapter deals with the requirements of the test-bed architecture, and presents detailed specifications of the system implementation.

Because this research focusses on the issues surrounding selfish nodes in ad-hoc wireless networks, the evaluation framework was to be able to facilitate experiments addressing node selfishness, and be as similar as possible to practical community networks. In this way the results are more applicable to practical network installations. The proposed cooperation enforcement model specifies that trust information be distributed via a flooding mechanism. The framework was to facilitate such a mechanism, and also be capable of implementing other ad-hoc routing protocols for future research.

In this chapter, existing ad-hoc routing protocols will be discussed. The problem of selfish nodes was not addressed when these protocols were designed, and thus these protocols do not take into account the detrimental effects of selfish nodes in a network path. However, the cooperation enforcement model assigns a trust rating to every node in the network

and it is possible to use this information when making routing decisions. Therefore, selfish nodes can be excluded from routing decisions and the reliability of network paths can be increased greatly. A novel path metric called the *trusted route* is presented below. The trusted route determines the best network path to a particular destination node taking into account the selfishness of intermediate nodes, and the number of hops the route traverses. The network framework provides a platform for evaluating the effectiveness of this metric in a practical environment.

4.2 Trusted Route

The routing protocols that are used for wired networks cannot be used in ad-hoc wireless networks. This is because nodes may join or leave the network at any time, causing a constantly changing network topology. Thus several routing protocols have been proposed for use in ad-hoc wireless networks.

The trusted route is a metric that is designed to be used within certain existing ad-hoc routing protocols. A brief synopsis of one table-driven protocol and one on-demand protocol is now included to show how the trusted route can enhance the routing decision process of these protocols.

4.2.1 Destination-Sequenced Distance Vector

The *Destination-Sequenced Distance-Vector* (DSDV) [32] is a table-driven routing protocol. It is an improved variation of the classic Bellman-Ford routing algorithm. Every node maintains a routing table that includes all available destination nodes, the number of hops to reach each destination and a sequence number. The sequence number is issued by the destination node and helps to distinguish between new and stale routes. A node transmits its routing table to its immediate neighbours periodically or when significant changes in the table have been made. Route updates can either be sent as a *full dump* (whole routing table), or as an *incremental update* (only those entries whose metrics have changed). In a stable network incremental updates are issued to avoid extra traffic, but in a fast-changing network incremental updates are not sufficient and full dumps will occur more frequently.

4.2.2 Dynamic Source Routing

Dynamic Source Routing (DSR) [33] is an on-demand routing protocol. DSR nodes maintain the entire route to each destination in their own routing tables. DSR has two main phases: *route discovery* and *route maintenance*. If a node does not have a route to a corresponding node then it initiates a route discovery. This involves flooding the network, depicted in Figure 4.1. The route request contains the address of the source and destination nodes, as well as a sequence number. The destination node chooses the route with the least hops and replies along that route with the route information. If two or more routes have an equal number of hops the route is chosen randomly between them. Each node maintains a route cache to avoid causing the network unnecessary congestion. If a node has a route to a particular destination it will not propagate the route request but rather reply with the information from its route cache. If the source node finds there has been a routing error then it will initiate route maintenance which involves flooding the network again to find a route to the corresponding node.

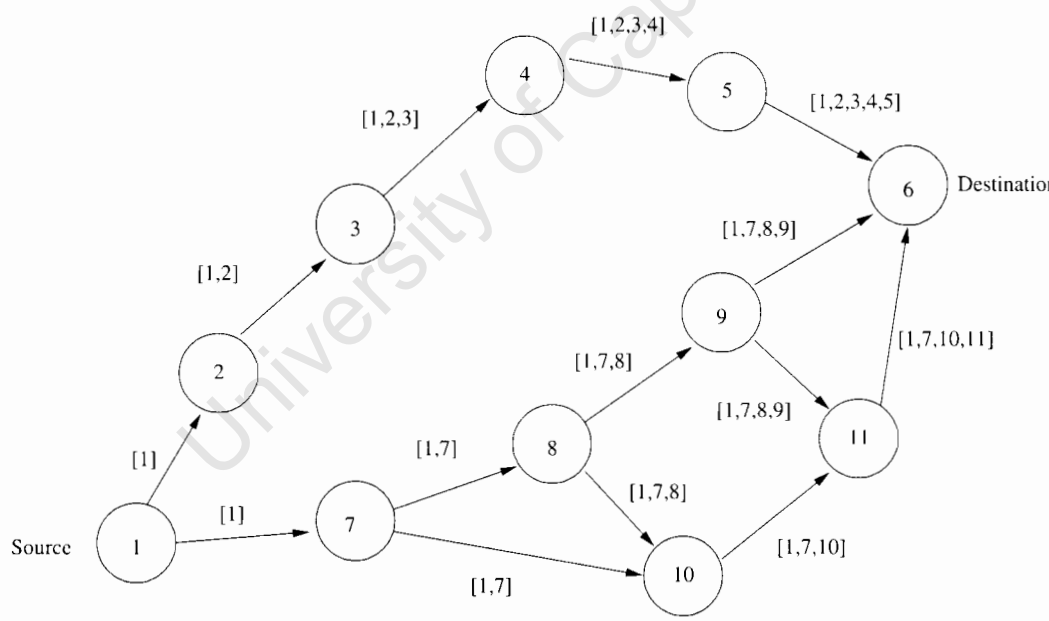

Figure 4.1: The *Route Discovery* phase of the Dynamic Source Routing protocol.

DSR is a robust routing protocol particularly when nodes in the network are mobile. However, in very small networks it is advantageous to use *flood routing* all the time. Every packet is propagated throughout the network. This is very inefficient in large networks,

adding unnecessary traffic. However, it suits low-complexity nodes and networks where bandwidth is not scarce. In addition, in networks in which strict quality of service constraints are imposed, flood routing always transmits packets from source to destination in the quickest time, because it will always find the optimum route.

4.2.3 Expected Transmission Count Metric

The metric used most often by ad-hoc routing protocols to determine the best path between source and destination nodes is the *minimum hop count metric*. This metric assumes that links either work well or not at all. This is a reasonable assumption when dealing with wired networks. However, in wireless networks the links may have variable packet loss ratios, particularly if selfish nodes are present.

A problem with a protocols such as DSR or DSDV is that if two routes are the same length the protocols choose arbitrarily between them, ignoring the fact that one path may offer substantially higher throughput. However, the DSR and DSDV protocols can be adapted so that another metric is used, instead of using a minimum hop-count metric to determine the optimal route. An example of this is the *expected transmission count* (ETX) metric [34] which is used for finding high-throughput paths in ad-hoc wireless networks.

The ETX metric determines the expected number of physical layer transmissions that will be required in order to transmit a packet from source to destination. This is achieved by examining link layer information returned from the network adaptor. If there are very lossy links in the path then the ETX will be much greater than the hop-count. If the channel is loss free then the ETX will be exactly the same as the hop-count for a particular route. The ETX metric takes into account the effects of link loss ratios, the asymmetry in loss ratios between the two directions of the link, and the interference between two successive links in a path. The ETX metric can improve traffic throughput greatly compared to traditional DSR [34].

4.2.4 Definition of the Trusted Route

It is proposed in our research, that instead of using the ETX metric to determine the optimum route in a DSR network, a metric based on the trustworthiness of the nodes along the path is used. Uncooperative nodes can be avoided during the routing process. Avoiding uncooperative nodes at this point means that when traffic is to be forwarded, only

cooperative nodes will take part in the routing, ensuring a robust link. Thus a cooperation enforcement scheme that rates the trustworthiness of nodes in the network, can greatly improve the performance of the Dynamic Source Routing protocol.

The nodes are each assigned a trust rating T_i (normalised to the range [0, 1]), according to the Cooperation Enforcement Model presented in the Chapter 3. When it comes to choosing between paths, the corresponding nodes must choose the path with the shortest route and the most trustworthy intermediate nodes. This is calculated by finding the average trust rating of all the nodes in a particular route. If two paths have an equal hop-count then the path with the highest average trust rating is chosen. If a particular route has more hops, denoted by h, it is still possible that it is better than a shorter route containing selfish nodes. (The number of intermediate nodes is one less than the number of hops.) The average trust rating of the nodes in the route is

$$\frac{\sum_{i=1}^{h-1} T_i}{h-1}$$

However, this does not take into account the number of hops in the path. A multiplier $(\frac{1}{\sqrt{h-1}})$ is used to reduce the metric of longer routes. The square root ensures that even very long routes containing cooperative nodes might still be chosen over routes with very selfish nodes. Thus the trusted route metric R is calculated as

$$R = (\sum_{i=1}^{h-1} T_i)(h-1)^{-\frac{3}{2}} \tag{4.1}$$

This metric provides good compromise between number of hops and the trustworthiness of the intermediate nodes. If a short route contains particularly selfish nodes then a longer route will be preferred.

4.2.5 Trusted Route Discussion

The trusted route equation is very general in its native form. A different exponent for the denominator can be substituted to give more priority to either short-but-unreliable routes or long-but-trustworthy routes. Also in some networks, a weighted sum of the trustworthiness of the nodes and the number of hops may provide better results. Depending on the nature of the traffic, it may be necessary to ensure that every packet is delivered,

in which case a more robust route with higher latency can be used. Traffic with strict end-to-end delay limits may need to traverse shorter paths regardless of the overall trust rating of the path. However, these issues are not investigated further in this work.

4.3 Test-bed Implementation Constraints

The test-bed implementation takes into account the requirements of the network architecture discussed above. However, there were certain implementation constraints when implementing the network architecture and these are presented below.

- The network nodes were to be able to implement several ad-hoc routing protocols. Specifically the nodes had to flood packets throughout the network to meet the requirements of the cooperation enforcement model. The software on the nodes was to be very flexible, and thus open source software was used throughout.
- The network had to include a gateway to a wired network. This ensured simple software updates from the Internet, and other wired machines. However, unsolicited traffic was not to be introduced to the ad-hoc network, causing congestion and possibly distorting evaluation results. Thus the gateway was to only allow connections originating from within the network.
- The hardware had to be readily available and widely used by existing mobile computers. The limited time constraints of this project restricted the choice of hardware to off-the-shelf solutions. This also ensured that the hardware used in the evaluation framework was similar to that used by actual community ad-hoc networks. In addition, the radio range and data transmission rates between the network nodes had be reasonable for this purpose.
- The network was to operate wirelessly in an unregulated (ISM) frequency band. This ensured that a license was not required as these bands are free to use, provided the maximum transmission power of the wireless devices falls within the specified range. Wireless ISM network adaptors typically operate in either the 2.4 - 2.4835 GHz or the 5.725 - 5.875 GHz frequency ranges.

4.4 Routing Software Options

A flexible and scalable manner of implementing packet forwarding was essential in order for the test-bed to be successful. While the primary focus of the experimental test-bed was to support the implementation of the proposed cooperation enforcement model, this framework was also to be used in future research within the broad field of ad-hoc networking. As such the choice of routing software was very important so that the test-bed framework could be adapted to meet the needs of research into routing, security, quality of service and other related issues. For research purposes the choice was limited to open source software, which allowed for finer control and adaptation to the test-bed needs. Two choices of routing software were identified from similar ad-hoc networking projects:

- The *Zebra* routing software [35] is modular routing software distributed under the GNU Public License. This flexible and reliable software package manages TCP/IP routing protocols. Zebra assigns a separate process for each protocol, using multi-thread technology under Unix kernels supporting multi-threading. Zebra is flexible as each module can be upgraded independently of the others, allowing fine tuning of the protocol stack. This architecture also provides reliability, as the failure of one process does not affect the entire system.

- A router similar to Zebra is the *Click Modular Router* [9]. Click was developed by a number of parties including the Parallel and Distributed Operating Systems Group at MIT, and is well documented in Eddie Kohler's PhD thesis [8]. The Click architecture allows one to build flexible and configurable routers out of packet processing modules called *elements*. Click routers can route up to 435,000 64-byte packets per second on a modest 700 MHz Pentium III machine. The Click Modular Router has been used in various ad-hoc networking test-beds including the Grid [6] and Roofnet [36] projects, and was thus preferred over Zebra. Click is open-source, and with its simplicity and flexibility, it was a perfect candidate for the evaluation framework. More information on the Click router and its integration into the framework is given later in this chapter.

4.5 Wireless Transmission Protocols

Several physical layer protocols were available for the test-bed implementation. A selection of these protocols are reviewed in this section, in order to establish their suitability for the architecture required in this work.

Currently, the most globally widespread wireless transmission system is based on the GSM family of protocols [37]. These are utilised by over one billion cellular telephone users. The family includes GSM, GPRS, EDGE and 3GSM. These protocols offer both cellular voice connectivity, and provide high data bandwidth (especially in the case of EDGE and 3GSM) over long range. However, these protocols are not suited to ad-hoc networks due to their inherent need for infrastructure. It is this infrastructure that allows the network operators to keep track of their network subscribers and bill them appropriately. Infrastructure such as base stations and radio network controllers can not exist in true ad-hoc networks, because it is a cooperative distributed environment that forms the infrastructure of ad-hoc networks. Therefore the possibility of using such protocols in this research was not considered.

A popular wireless protocol that has become the de facto standard for low-cost short-range between mobile PCs, mobile telephones and other portable devices is *Bluetooth* technology. Bluetooth is often a feature of small, low-power personal devices, and thus can be used in ad-hoc networks. Industry leaders in the fields of telecommunications, computing and networking form the Bluetooth Special Interest Group (SIG), and release the Bluetooth specifications. The IEEE 802.15 Working Group has approved the Wireless Personal Area Networks (WPAN) standard, which is derived from the Bluetooth specification [38]. A Bluetooth network consists of a master station and up to seven active slaves, and is known as a *piconet*. If two such piconets overlap then they form a *scatternet*, depicted in Figure 4.2. A slave node may communicate with all the piconets it belongs to but only in a time-multiplexing fashion whereby it communicates to a single piconet exclusively during a particular time-slice. While the Bluetooth specifications do define the notion of a scatternet they fail to provide the mechanisms to construct one.

The Bluetooth protocol stack covers most of the OSI layers, and includes the following protocols: Bluetooth radio, Baseband, Link Manager Protocol (LMP), Logical Link Control and Adaptation Protocol (L2CAP), and the Service Discovery Protocol (SDP). The Bluetooth radio operates in the ISM 2.4 GHz band, and every piconet has a gross bit rate of 1 Mb/s, before taking taking into account the protocol overheads and the polling scheme.

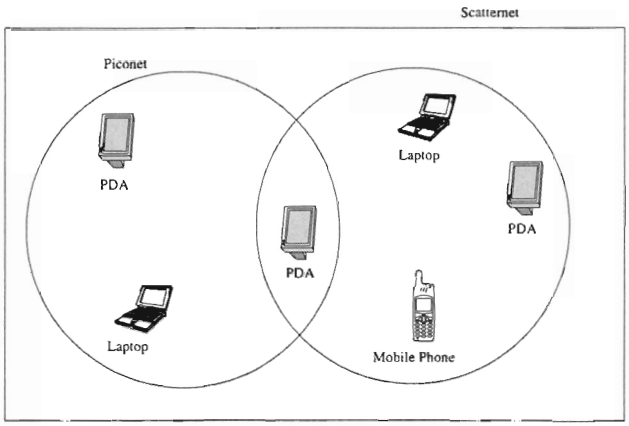

Figure 4.2: Overlapping Bluetooth piconets made up of PDAs, Mobile Phones and Laptop Computers form a scatternet.

The radio utilizes a frequency hopping technique called Frequency Hopping Spread Spectrum (FHSS), based on a pseudo-random hopping sequence, that reduces interference with other devices operating in the same range of frequencies.

Bluetooth is a suitable candidate for ad-hoc networks. It uses very little power and achieves reasonable bit rates. However for this particular research it was not ideal in the following respects:

- The master slave relationship, while only at the lower protocol layers, suggests that the network is not truly ad-hoc, as one node has control over another's behaviour. Since nodes may only transmit after they have received a packet from the master node, the master could be considered the controller of the network. Since this research focusses on nodes not performing their network functions correctly, no one node should control the behaviour of the other nodes. A contention-based scheme was more appropriate for the test-bed framework.

- The range of Bluetooth is very short, sometimes as little as 10 metres. If the ad-hoc network is to be used effectively for data transfer over fairly long distances (e.g. 1 km) traffic would travel through an excessive number of hops.

- The Bluetooth protocol stack is very well defined and needs to be thoroughly modified to incorporate scatternet functionality at the network layer.

The above factors made Bluetooth inappropriate for the required architecture. However, another candidate wireless protocol exists in the form of the IEEE 802.11 family of protocols [38]. The first IEEE 802.11 protocol was adopted in 1997. Since then a number of task groups have extended the standard to include 802.11a, 802.11b, 802.11e, 802.11g and 8011n. The 802.11b standard allows for data rates of up to 11 Mbps in the 2.4 GHz frequency band. The 802.11a standard operates in the 5 GHz band and specifies data rates of up to 54 Mbps. The 802.11g standard also operates in the 2.4 GHz band, and through Orthogonal Frequency Division Multiplexing (OFDM) modulation has also realised data rates of up to 54 Mbps. The 802.11e task group has attempted to enhance the medium access control with quality of service features to support voice and video applications over 802.11 networks.

A new standard that is still to be ratified by the IEEE will be known as 802.11n. 802.11n may see transmission rates of up to 500 Mbps. However, it is more likely that the data rates will be in the range of 250 Mbps, brought about by the use of multiple input multiple output (MIMO) antenna technology. This high data rate will allow bandwidth-hungry applications to operate well over wireless links, and may further increase the popularity of civilian ad-hoc networks. The IEEE Wireless LAN standards are summarised in Table 4.1.

Table 4.1: Wireless LAN frequencies and maximum data rates.

IEEE Standard	Frequency Band	Data Rates
802.11b	2.4 GHz	11 Mbps
802.11g	2.4 GHz	54 Mbps
802.11a	5 GHz	54 Mbps
802.11n	2.4 or 5 GHz (still to be decided)	200+ Mbps (estimated)

Many mobile computing devices, such as laptops, now incorporate 802.11 technology for wireless network access. Furthermore, because of the high data rates it is possible to stream high quality video over these links. A benefit of the 802.11 protocols is that they can be used in infrastructure mode or in ad-hoc mode[1]. When used in infrastructure mode, nodes must register with an access point unit. When a station goes out of range of an access point it must connect with a new, fixed-position access point, and this is called a *handoff*. Conversely, in ad-hoc mode stations communicate directly. An ad-hoc 802.11 network is known as an *Independent Basic Service Set* (IBSS).

[1] Ad-hoc 802.11 networks are also known as infrastructureless networks

The Intel Corporation has recently proposed the 802.11s standard. This standard will be reserved for ad-hoc mesh networks. Although this standard has not been defined yet, it will be built on top of, and be backwards compatible with, the current 802.11a/b/g standards. 802.11s will include *mesh portals* that will be able to connect to regular 802.11 networks. It is also expected that quality of service provisions will be built into the standard to enable different traffic types to be prioritised.

Unlike Bluetooth, 802.11 only specifies the physical layer and MAC layer of the OSI protocol stack. This means that there is a lot of flexibility when it comes to specifying higher layers. In particular it is easy to integrate the lower 802.11 layers with an ad-hoc network layer. A number of different technologies are supported by the 802.11 physical layer such as infrared, frequency hopping spread spectrum, direct sequence spread spectrum and orthogonal frequency division multiplexing. The 802.11 MAC layer provides both contention-based and contention-free access control. The MAC protocol's basic access method is the Distributed Coordination Function (DCF), which is a carrier sense multiple access with collision avoidance (CSMA/CA) protocol. However, the MAC layer also specifies the point coordination function (PCF) which operates similarly to a polling system. Because no master / slave relationship is desired in the test-bed framework, the PCF function is not used. In addition to this, the PCF is usually controlled by an access point, and this is also not appropriate. The brief description of the 802.11 DCF function that follows will justify the suitability of a contention-based medium access scheme in the test-bed implementation.

Stations that use DCF must first sense the channel to determine whether it is available. If the station finds the channel to be idle for a period of time exceeding the Distributed InterFrame Space (DIFS) then it concludes that the channel is available for its own transmission. Unfortunately, this is susceptible to the hidden terminal problem, which is not discussed here but is well documented in the literature [39]. If a station has a message to transmit, it includes the projected length of its transmission, which every other station then stores in its *Network Allocation Vector* (NAV). Thus the other stations will know the length of time that the channel will remain busy and do not waste power listening to the channel during this period. Since no collision detection is implemented in the DCF, the stations must wait for a positive acknowledgement from the receiver to know that the frame arrived successfully. The receiver waits for a period known as the *Short InterFrame Space* (SIFS) before transmitting an acknowledgement frame. Note that the SIFS is shorter than the DIFS so that the channel does not appear idle to other stations. If a station has

just transmitted a packet and has another ready for transmission, it must first perform a backoff procedure before transmitting the second packet. This method guarantees fair access to the wireless medium [38].

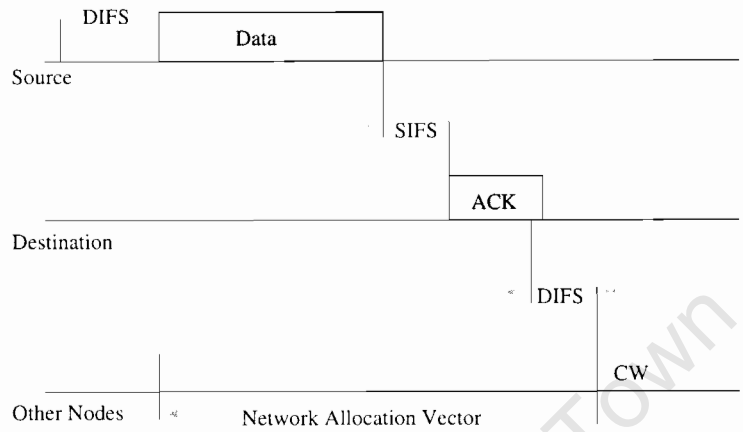

Figure 4.3: A successful DCF transmission.

If it is determined that there is a transmission error or frame collision then the channel remains idle for at least one *Extended InterFrame Space* (EIFS). After this the stations initiate the backoff procedure to schedule their transmissions. The backoff process is a slotted binary exponential backoff technique and it ensures that all stations have reasonable access to the channel.

In ad-hoc mode there is no master station to control the medium. Therefore, 802.11 employs two functions, *synchronisation acquisition* and *synchronisation maintenance*, to synchronise the stations in the IBSS. When a node joins an ad-hoc network, it scans the specified channels searching for control frames. If the node does not locate its IBSS then it initialises a new IBSS. This is known as synchronisation acquisition. Synchronisation maintenance maintains a common clock amongst the nodes. It relies on a distributed algorithm that involves transmitting beacon frames at a known nominal rate. The beacon interval timing is determined by the station that initialised the IBSS. Because of the suitability of 802.11 DCF and the DSSS protocol in ad-hoc networks, they are utilised in the test-bed framework.

4.6 Directional vs. Omni-Directional Antennas

A number of antennas are available for transmitting ISM-band signals. For increased radio range it is possible to use directional antennas to link two nodes over large distances. However, due to the nature of ad hoc networks, it is beneficial to have radio connectivity to all neighbours at the same time. Assuming the network nodes are distributed fairly uniformly in a particular area, a directional antenna will not provide radio communication to all neighbouring nodes, and is thus not appropriate.

Omni-directional antennas that provide a 360^{o} radio coverage area, allow the network to maintain a fully connected mesh structure. This is very important if the nodes are mobile, which is often the case in ad-hoc networks. The drawback of omni-directional antennas is that they have less gain and the radio coverage of the individual nodes is reduced. In the next section the link budgets of the wireless adaptors using omni-directional antennas are analysed to determine the maximum distance between nodes of the test-bed.

4.7 Link Planning

Three factors must be taken into account to determine the maximum distance between nodes: the *effective transmitting power* (ETP), the *free space loss* (FSL) and the *effective receiving sensitivity* (ERS) [10]. For effective 802.11 transmission, $ETP+FSL+ERS > 0$, and what remains above zero is the margin of the link. For very good performance a link should have a margin of between 6 and 10 dB.

In an ordinary link budget calculation the sending and receiving properties are generally different, but if identical transceivers are used in a peer-to-peer fashion in a symmetric environment then we can assume that these budgets are the same. The ETP (in dbm) can be calculated as follows

$$ETP = TransmitterPower[dBm] - CableLoss[dB] + AntennaGain[dBi]$$

Most 802.11 desktop hardware transmits at approximately 30 mW (15 dBm). However, it was found that the Cisco Air-PCI 352 network adaptors transmit up to a maximum of 100 mW (20 dBm). These adaptors are used in the test-bed for increased signal quality.

Cisco adaptors have cable and connector losses of approximately -6dB, and the antenna gain is 2.2 dBi [40]. This results in an ETP of 16.2 dBm.

On the receiving side the ERS is calculated as

$$ERS = AntennaGain[dBi] - CableLoss[dB] + ReceiverSensitivity[dBm]$$

Once again the cable and connector loss is -6dB and the antenna gain 2.2 dBi. The receiver sensitivity of the Cisco Aironet card running at 11 Mbps is -85 dBm. Giving a total ERS of 81.2 dBm.

To calculate the FSL the following equation is used:

$$FSL = 92.45 + 20\log_{10} f + 20\log_{10} d$$

where f is the frequency in GHz and d is the distance in kilometres. If a 10 dB margin for the link is allowed then the maximum free space loss can be -87.4 dB. At a frequency of 2.45 GHz this allows a maximum separation of approximately 224 metres between nodes. These calculations are for outdoor environments only. Indoors there are a number of obstacles which can attenuate 2.45 GHz signals [41], demonstrated in Table 4.2. These factors significantly affect the maximum range of communication. It is known that water attenuates the 2.4 GHz frequency. Thus rain and trees with high moisture content can also reduce the maximum range of communicating nodes. For experimental purposes the transmission power of the wireless adaptors can be reduced in order to reduce the communication range of the nodes. This allows experimental multi-hop networks to be set up in a smaller area.

4.8 The Click Modular Router

The Click software is installed on the nodes of the test-bed to route modified IPv4 packets throughout the network. The elements that make up a functional router are software components that represent a unit of router processing. Elements perform relatively simple tasks, but are linked together using *connections* to make up more complex router configurations. Click router configurations can be thought of as directed graphs of elements with connections making up the edges.

Table 4.2: Typical attenuations of indoor obstacles.

Obstacle	Attenuation (dB)
Floor	30
Brick wall with window	2
Office wall	6
Metal door in office wall	6
Cinder block wall	4
Metal door in brick wall	12.4
Brick wall next to metal door	3

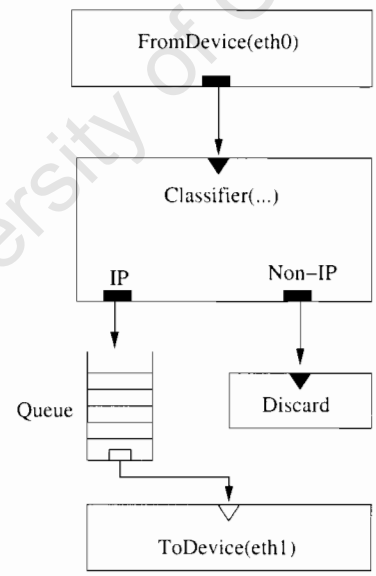

Figure 4.4: A Click Router Configuration to filter non-IP packets from the *eth0* interface and forward IP packets to the *eth1* interface.

A simple Click router configuration is shown in Figure 4.4. The elements may have any number of input and output ports depending on their type. The ports may be of type *push* (black), *pull* (white) or *agnostic*. A push port will pass packets to downstream elements, whereas a pull port allows a destination element to initiate a packet transfer. Agnostic ports are either push or pull depending on the port they are connected to. It is illegal to connect a push port to a pull port. If no packet is available a pull upstream element returns a null pointer to avoid blocking the router. In the example in Figure 4.4 the *Classifier* element pushes packets to either the Discard element if it is a non-IP packet, or pushes it onto a queue if it is an IP packet. The *ToDevice* element pulls packets from the queue when it is ready to transmit.

Click runs on the Linux operating system, either at *userlevel* or as a *kernel module*. The difference is shown in Figure 4.5. At userlevel, Click sends messages to and from the operating systems protocol stack using the *tunnel* device. The tun device looks like a file to the Click process and like a network interface to Linux. Running Click at userlevel results in a large amount of overhead, including buffers in the tunnel device and the delay in crossing the userlevel / kernel boundary. However, if the router has a fatal error at userlevel then at least the kernel will not crash. For this reason it is very useful to debug Click configuration files at userlevel before porting them to kernel. When running Click at kernel level, the Click module runs on a separate kernel thread. The Click router lies between the kernel's protocol stack and the network device drivers. In this way Click can intercept packets from network device drivers without the kernel seeing them, unless the Click configuration specifically passes theses packets to the host. In addition, packets from the kernel can be manipulated before passing them through to the network device drivers. This feature is exploited in the experimental test-bed.

4.9 Adapting IPv4 for Ad-hoc

Ad-hoc networks are often found as stand-alone entities, not connected to any further networks. However, although true ad-hoc networks require no existing infrastructure, it is often useful to link an ad-hoc network to some other wireless or wireline network. This can be accomplished by assigning a static gateway node that links the ad-hoc network to the Internet. Ad-hoc routing protocols are fundamentally different to wireline routing protocols and this presents a problem in designing the architecture of the gateway node. In this research it was useful to make the flooding mechanism compatible with an ordinary

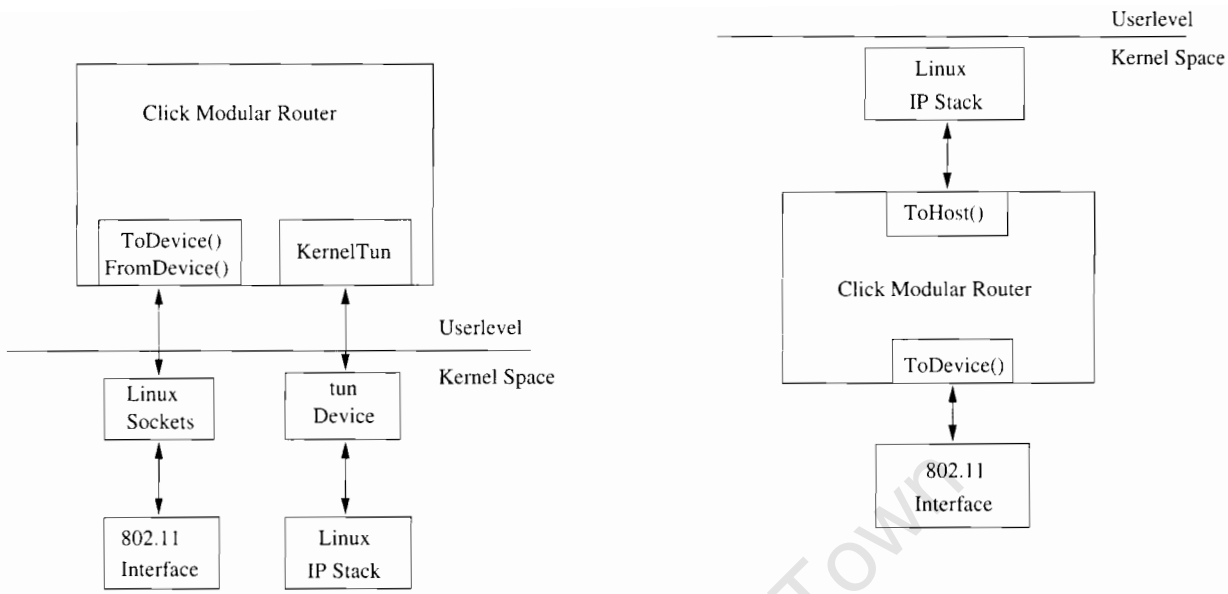

Figure 4.5: Click running at userlevel (left) and in kernel space (right).

IPv4 protocol network without the need for tunneling. The Linux protocol stack provides support for IPv4 already, and thus it was decided to use the IPv4 packet architecture for the flooding mechanism. Note that this is purely for the purposes of simplicity and the test-bed framework is not limited to be used in this way. However, if the network is going to have Internet access then the nodes must use IP.

It is also possible to encapsulate ordinary IPv4 packets in another network-layer header. This header can dictate the routing path for the packet to take. DSR stipulates that the entire packet path be included in the packet header, as it is only the source node that contains the routing information. This adds overhead but greatly simplifies the routing. In a flood-routing scenario it is not necessary to have routing information and thus no additional headers are required.

Cooperation enforcement models implement flood routing to probe all the other nodes of the ad-hoc network for trust information. For effective flood routing, packets must be differentiable so that when a node receives a packet it can know whether it has seen it already. Then it chooses whether to propagate the packet or not.

The structure of the IPv4 packet is shown in Figure 4.6. The *Identification* field is typically used to identify the current datagram allowing datagram fragments to be pieced together. However, measurements have shown that less than 0.25% of Internet packets are fragmented

[42], because it is known that network-layer fragmentation is detrimental to end-to-end network performance [43]. Thus modern network stacks prevent fragmentation, regardless of the underlying media, by implementing automatic Maximum Transfer Unit (MTU) discovery [44]. Therefore, the Identification field is used in the implementation of the flooding mechanism to uniquely identify packets in the network.

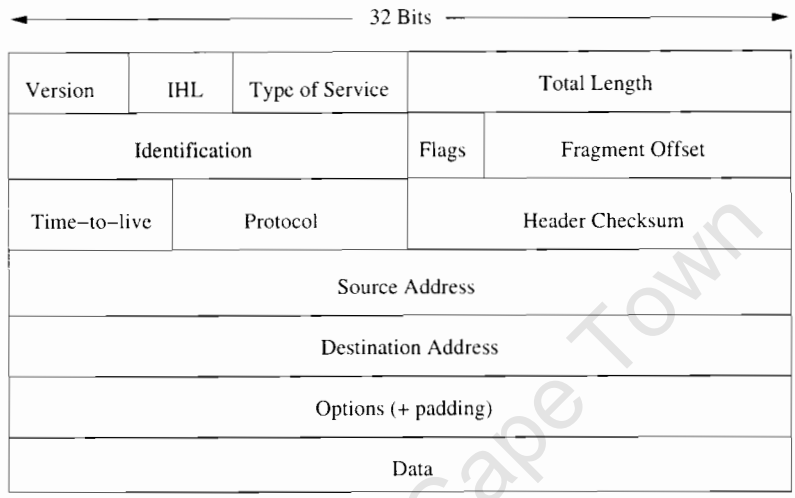

Figure 4.6: The structure of the IPv4 packet - the identification field is used for uniquely identifying packets.

For the purposes of identifying individual packets in the network the identification field is set to a random 16-bit number. This allows nodes in the network to identify a particular packet and decide whether or not it has been seen recently. This is achieved by including a new Click element in the router called *RandomID(...)*. The element takes the IP packet and alters only the identification field, assigning it a random number in the range [0 .. 65535]. Another new element is also included in the router configuration called *SeenBefore(...)*. This element keeps a table of the last N packets, and outputs the packet on *port zero* if it is not in the table, and on *port one* if it has been seen recently. The recently seen packets are discarded, and new packets are forwarded. The inter-connection of these elements is detailed in the Click Router Configuration later in this chapter.

IP routers used in their traditional context will always decrement the IP packets *Time-to-live* (TTL) field, when hopping between subnets. This 8-bit field allows a packet to transit up to 255 hops before being discarded. The TTL prevents a packet from bouncing around indefinitely, and should be used in ad-hoc networks to avoid loops and unnecessary

network congestion in case of a routing error. In Click the *DecIPTTL(...)* decreases the TTL field by one. The TTL field in the test-bed is set to a reasonable number to ensure that if by some chance a loop does occur, that the packet will eventually be destroyed.

4.10 Handling Broadcasting and ARP Requests

The *Address Resolution Protocol* (ARP) maps network layer IP addresses to data link layer hardware addresses. This is required because stations communicate at the link layer using hardware addresses. In order to transmit messages the hardware address of the destination node must be known. A wired Ethernet station performs an ARP lookup by broadcasting an ARP request packet. A host that has the corresponding IP address will respond with its unique 48-bit Ethernet address and this will be added to the requesting host's ARP table. This protocol works well in an Ethernet segment and other ad-hoc routing protocols, but will not work well with the flooding mechanism. In order to flood the network it is essential that each station be able to broadcast every packet to all its neighbours at the same time, regardless of the destination IP address. The broadcast Ethernet address is accepted by all hosts regardless of their unique Ethernet addresses, and can be used for this purpose.

Since ARP has been included in the Linux protocol stack with the IEEE 802.3 MAC sublayer protocol, and it is not desirable to make major modifications to this stack, all ARP queries from Linux are intercepted. It was found that to do this effectively Click should operate at kernel level. At userlevel it is possible to sniff the packets that pass out of the Linux stack and modify them. However, the original packets are still transmitted, together with the modified packets, causing multiple instances of the packet in the network. When Click runs as a kernel module, the packets can be intercepted without being transmitted.

When a node wants to send a packet to a particular IP address, and this address is not in its ARP table, it sends out an ARP request. These ARP requests are intercepted as they leave the Linux protocol stack, and an ARP reply is sent back to the host, without the packet having been transmitted. Thus the Linux ARP table can be made to reflect any IP-MAC mapping that is chosen. When Linux sends out an ARP Request it is given an arbitrary MAC address in return. Thus all IP addresses in every host's ARP table are mapped to bogus MAC addresses, in order to satisfy Linux's need to have a corresponding MAC address for every IP address.

It is necessary that all neighbouring nodes hear every transmission for the flooding mecha-

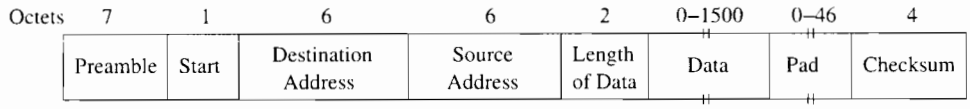

Figure 4.7: The 802.3 frame format.

nism. When outgoing packets pass through Click they have their 802.3 headers stripped and replaced with the 802.3 headers containing the broadcast MAC address (FF:FF:FF:FF:FF:FF) as the destination address. The result is that the neighbouring nodes hear every transmission, because their network adaptors accept all broadcast communications. The neighbours are then able to propagate these transmissions throughout the network.

4.11 Interfacing Click with the Linux Protocol Stack

For the router to work in the desired fashion, Click needs to operate between the 802.11 network device and the Linux protocol stack (Figure 4.5). Thus click can intercept packets from the network device, or the protocol stack, and modify them before passing them on. In order to do this a fake network interface is created, shown in Figure 4.8. Linux treats this network interface as if it were a regular network interface, and hence Click is transparent to the kernel. The fake network interface has its own IP address and subnet mask, and it appears in the Linux routing table (Figure 4.9). The interface is given an arbitrary MAC address. The IP address of the fake interface, *fake0*, is used for all communication with the node. The IP address of the actual wireless adaptor interface, *eth0*, is irrelevant as it not used by the kernel. All packets destined for the ad-hoc wireless network are sent to the fake interface by the Linux protocol stack. They are passed through the Click router, and then to the wireless network adaptor.

When incoming packets are received at the network adaptor they are handed to Click. These packets are then passed to the kernel, appearing to come from the fake network interface. This allows the presence of the Click router to be hidden from the Linux protocol stack. The underlying protocols and network are treated by Linux as a regular wireline packet-switched IPv4 network.

```
fake0     Link encap:Ethernet  HWaddr 00:01:02:03:04:05
          inet addr:10.192.0.88  Bcast:10.192.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:306194 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 b)  TX bytes:96916656 (92.4 MiB)

eth0      Link encap:Ethernet  HWaddr 00:11:20:48:02:46
          inet addr:10.128.0.100  Bcast:10.255.255.255  Mask:255.0.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:893416 errors:0 dropped:0 overruns:0 frame:0
          TX packets:344679 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:331840862 (316.4 MiB)  TX bytes:100038188 (95.4 MiB)
          Interrupt:11 Base address:0xe000
```

Figure 4.8: *ifconfig* shows the fake0 interface and the Cisco wireless network adaptor.

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
10.192.0.0	*	255.255.0.0	U	0	0	0	fake0
default	10.192.0.1	0.0.0.0	UG	0	0	0	fake0

Figure 4.9: The Linux routing table uses only the fake network interface.

4.12 Click Router Configuration

The flooding mechanism configurations used in the test-bed are shown in Figure 4.10 and Figure 4.11. The various Click elements used in the router configurations are detailed in Appendix A. The router floods packets throughout the network, eliminating loops and duplicate packets at every node.

Frames originating from the host machine, and those arriving at the 802.11 network adaptor, are handled separately. Frames that are received by the network adaptor (Figure 4.10) are classified according to their type. ARP requests are handled by the *ARPResponder*. The hosts should not receive any ARP requests as these should be intercepted at the source machine. However, ARP functionality is included for robustness. None of the nodes use individual MAC addresses because they want their communications to be heard by all neighbouring nodes, thus they discard any ARP replies. Non-IP packets are also discarded. IP packets are stripped of the 802.3 header information and passed through to a classifier. At this point the host determines whether to keep, discard or forward the incoming packets. Packets destined for the host machine are passed to Linux via the *ToHost* element. Packets for other hosts that have not been seen before, have their Time-to-Live field reduced by one, and are sent back out the network adaptor.

Frames originating from the host itself (Figure 4.11) are classified as ARP requests or IP packets. ARP requests are handled by the Click router. The router replies with a bogus MAC address, thus satisfying Linux's requirement to have a IP / MAC Address mapping. Outgoing IP packets are stripped of Linux's MAC header, which includes the bogus MAC address. They then have a new 802.3 header appended. This new header has the broadcast MAC address in its destination field and the network adapter's MAC address in the source field. These frames are then sent to the 802.11 device.

Applications running on the nodes of the ad-hoc network will see the underlying network as a traditional IPv4 network stack. The wireless topology and routing is hidden from the application layers which makes this network very flexible in term of the range of applications it can support. While it will be shown that the Transmission Control Protocol (TCP) does not perform well in this type of environment, TCP is still functional and no additional configuration is required to allow the nodes to use TCP for their communications.

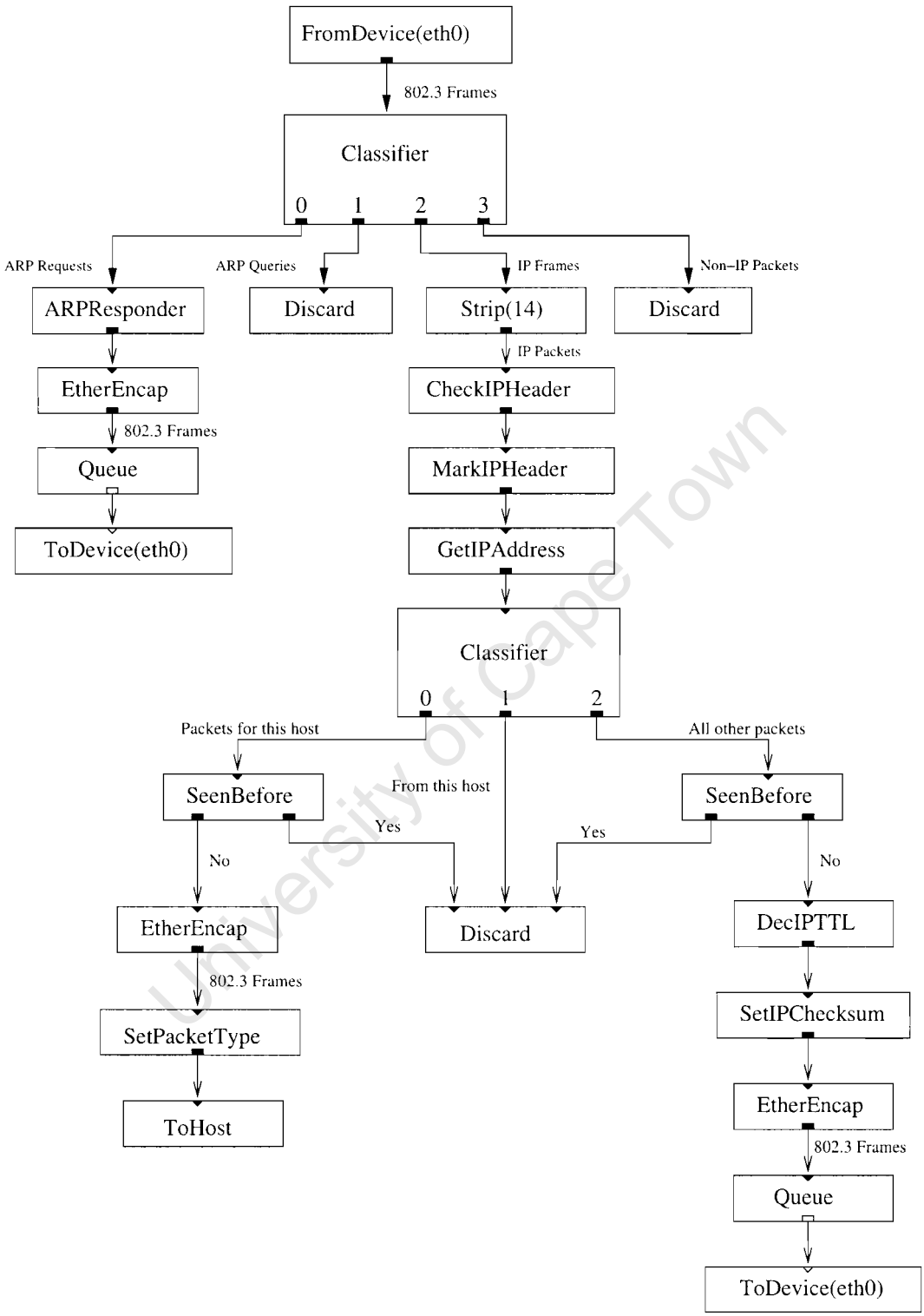

Figure 4.10: Click router configuration for frames originating from the 802.11 device.

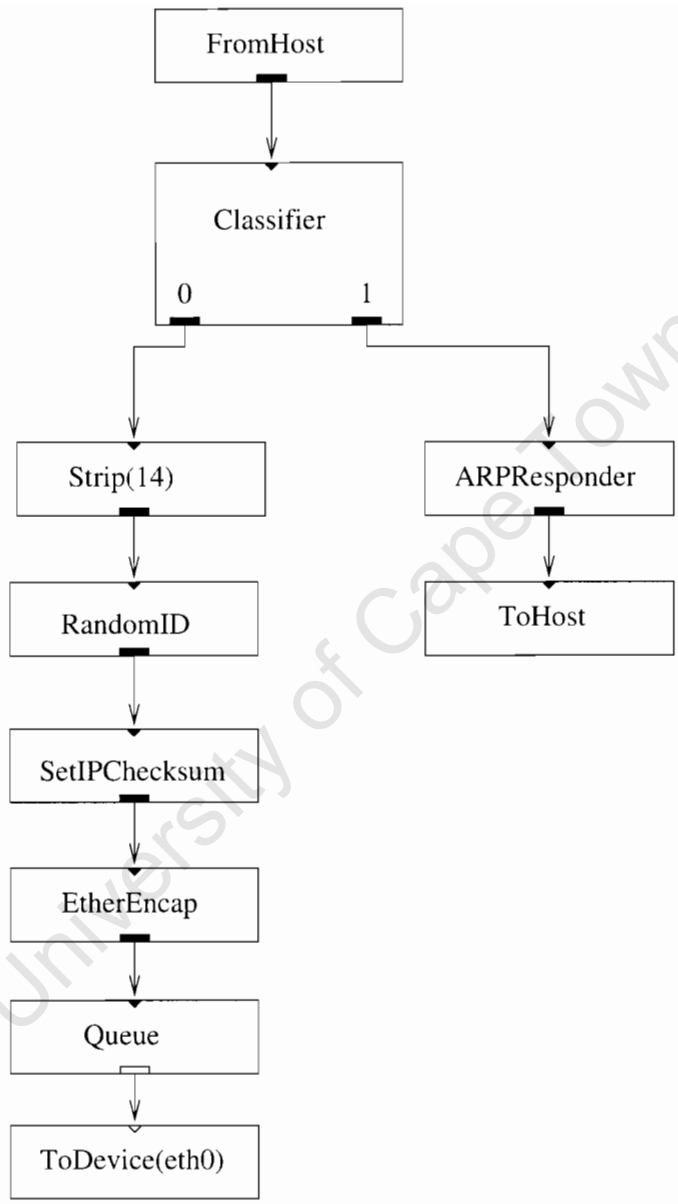

Figure 4.11: Click router configuration for frames originating from the host machine.

4.13 Gateway Node

One node in the network acts as a gateway to a wired Ethernet network. This does not strictly conform to the definition of an ad-hoc network as the gateway can be considered to be fixed infrastructure. However, for the purposes of research it is useful that the nodes have Internet connectivity and access to existing wireline machines. Thus the nodes include the gateway node in their Linux routing table as the default entry (Figure 4.9). If they have any traffic which is not destined for a node in the ad-hoc network it is routed through the wireless network, and enters the wired network through the gateway node.

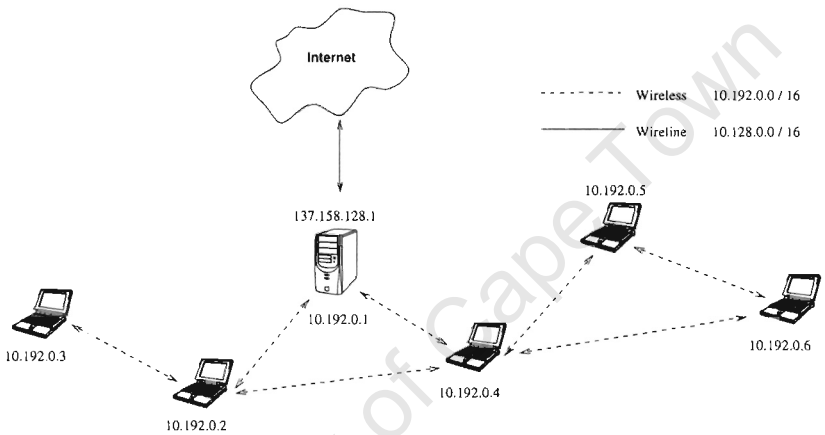

Figure 4.12: The gateway node forwards and NATs traffic to the wireline network.

The gateway node has both a wired and a wireless interface. If the gateway node sees traffic on its wireless interface not destined for a wireless node, it passes this traffic to its Linux stack. The test-bed nodes do not use globally routeable IP addresses. Thus the gateway node must perform Network Address Translation (NAT) on the nodes' addresses. The Linux utility *iptables* is used to perform NAT on the traffic and sends this traffic out the wired interface. The ad-hoc network has its own IP subnet and is hidden from the wired network. All traffic from the nodes of the ad-hoc network appears to come from the gateway node.

4.14 Evaluation Framework Summary

In the preceding sections the issues surrounding the design and implementation of the network test-bed were discussed. A summary of the node specifications is given in Table 4.3. The final node specifications were determined through research of similar test-bed architectures and a fair amount of trial-and-error. It is important to note that the architecture described in this chapter is, at this stage, specific to the Linux operating system, and therefore the hardware used must have suitable Linux drivers. Initial attempts with Atheros chipset based 802.11 hardware were unsuccessful as the drivers did not function properly in ad-hoc mode. However, the Cisco Aironet-352 PCI and PCMCIA cards are released with comprehensive Linux drivers. These cards also have better antenna gain, and higher maximum transmission power than most of the 802.11 hardware reviewed.

Table 4.3: Node specifications.

Processor	Pentium MMX 166MHz (333.41 BogoMips)
Operating System	Debian Linux 2.4.26 (Patched for Click)
802.11 Hardware	Cisco Aironet 352
WiFi Protocol	IEEE 802.11b
Transmission Speed	11 Mbps (max)
Transmission Power	100 mW (max)
Antenna	2.2 dBi Omni
Distance between nodes	20 metres (average)
Routing Software	The Click Modular Router 1.4.3
Routing Protocols	Flood Routing / DSR / Modified DSR (Using Trusted Routes)

4.15 Chapter Discussion

In this chapter the requirements of the ad-hoc wireless test-bed architecture were outlined. The benefits have been discussed of using the proposed cooperation enforcement model in conjunction with the Dynamic Source Routing protocol to determine trusted routes. A detailed description of the implementation of the test-bed has been included, and the issues surrounding this implementation of the framework have been addressed.

The nodes of the test-bed are ordinary desktop or notebook machines running the Click Modular Router with the Linux operating system. The machines use 802.11b network adaptors, that operate in ad-hoc mode. The current network architecture supports the flooding mechanism required by the cooperation enforcement scheme, and the Click configuration to achieve this has been presented. The DSR and DSDV routing protocols, developed by other researchers, have also been implemented on the current network test-bed. The Linux IP protocol stack is used as far as possible to make the architecture flexible, and support all existing network-dependant applications. The topology of the network can be changed as required by the particular investigation being carried out, and is only limited in that every node must be in radio range of at least one other node at all times.

In the next chapter the evaluations carried out on the network test-bed are described. Experiments are performed to analyse the detrimental effects of uncooperative nodes on traffic throughput and reliability, and also to determine to what extent a cooperation enforcement scheme can mitigate these effects. Experiments are also performed to quantify the benefits of improving DSR in an uncooperative environment.

Chapter 5

Evaluations Performed Using the Experimental Framework

Several experiments are described in this chapter that will confirm the need for cooperation enforcement in ad-hoc wireless networks. The experimental test-bed architecture, presented in the previous chapter, is used to implement these evaluations. This study is limited to the three specific experiments, that are designed to show the need for cooperation enforcement and the effectiveness of distributed trust mechanisms. The rest of this chapter is laid out as follows:

1. In the first set of experiments, the negative effects of selfish nodes in an ad-hoc network are demonstrated. Practical evaluations are performed to determine the effects that selfish intermediate nodes have on the reliability and throughput of end-to-end connections. It is shown that selfish nodes can dramatically decrease the quality of ad-hoc network links.

2. In the second set of experiments the cooperation enforcement model - a distributed trust mechanism proposed in Chapter 3 - is evaluated to determine how well it identifies and rates selfish nodes. The experiment involves the nodes of the test-bed playing in an iterated Prisoner's Dilemma tournament against a node that exhibits varying amounts of un-cooperativeness. Descriptions of the origin and rules of the iterated Prisoner's Dilemma tournament are included. Various strategies are presented that aim to out-perform a selfish node in the tournament, but it is shown that only the

distributed trust mechanism performs adequately. These concepts are explained later in this chapter.

3. The novel concept of the trusted route was presented in the previous chapter. The trusted route is based on trust ratings provided by the cooperation enforcement model. The third and final evaluation of this research aims to show how the trusted route can be used as a metric to enhance the Dynamic Source Routing protocol, whereby routes that contain cooperative intermediate nodes are selected. The results indicate that by using trusted routes, it is possible to increase the average throughput in a network that contains uncooperative nodes.

Five nodes were available for the evaluations. Where more than five nodes were required for the trusted route experiments, the intermediate node adjusted their behaviour dynamically thereby simulating the presence of extra nodes in the network. This did not affect the results of the experiments, as comparisons were only made between results obtained on the same network architecture.

5.1 Throughput and Reliability Evaluations in an Uncooperative Environment

An evaluation is described in this section that assesses the effects of uncooperative nodes in the experimental test-bed environment. These experiments show that selfish nodes in an ad-hoc wireless network can severely reduce traffic throughput and network reliability. The experiments also reveal the throughput limits of the test-bed architecture in a practical setting. The results of the experiments performed in this section emphasize the need for cooperation enforcement in ad-hoc wireless networks.

It is advantageous to use the test-bed for these experiments, instead of simulating the results, so that all the factors of practical networks are taken into account, e.g. wireless interference, processing overheads, and hardware limitations. The throughput and reliability tests are performed using several different network topologies that replicate practical scenarios.

5.1.1 Throughput Tests

In order to evaluate the maximum traffic throughput supported by the evaluation framework, a fixed amount of traffic is generated at a source node, and the time taken for it to be successfully delivered to a destination node is measured.

The framework presented in the previous chapter only describes the characteristics of the test-bed from the physical layer in the OSI model up until the network layer. However, a transport layer protocol must be used in order to provide reliable data transport from the source to the destination node. Two popular transport layer protocols used extensively in the Internet, are the Transmission Control Protocol (TCP) [45], and the User Datagram Protocol (UDP) [46]. TCP provides a reliable end-to-end byte stream over an unreliable network, whereas UDP is a connectionless, unreliable protocol. UDP has essentially the same functionality of raw IP, except that it makes use of ports.

Ad-hoc networks require some form of reliable transport mechanism, especially if there are selfish nodes present. Because TCP ensures reliable data transfer, and its use is already widespread, it is likely that TCP will be used extensively in community ad-hoc networks. Thus it is also used in these experiments. However, a drawback to using TCP in this environment is that TCP implements congestion control mechanisms. These mechanisms attempt to ensure that traffic load is appropriate for the network conditions. In a very unreliable network environment TCP will confuse the lossy end-to-end link with excessive network congestion. Moreover, TCP uses an exponential backoff mechanism to lower the frequency at which unacknowledged packets are retransmitted [47]. Thus a lossy link will cause additional delays at the transport layer, and overall traffic throughput is further reduced. This feature of TCP is illustrated in the results of the traffic throughput experiments.

Several improvements for TCP performance over wireless links have been proposed in the literature [48]. A transport protocol better suited to lossy wireless links (e.g. a modified version of TCP) should preferably be used in community ad-hoc networks, however, this work does not attempt to address this particular issue.

Throughput in a Purely Cooperative Environment

If the network does not contain any selfish nodes, it is a purely cooperative environment. Traffic throughput in a purely cooperative environment is affected by a number of factors,

including the number of hops between source and destination. For interest, the data rates achieved in the test-bed are measured using an increasing number of hops. The figures, shown in Table 5.1, show the average throughput rates of a suitable sample size of generated TCP traffic. Packets containing 1500 Bytes of arbitrary data are used.

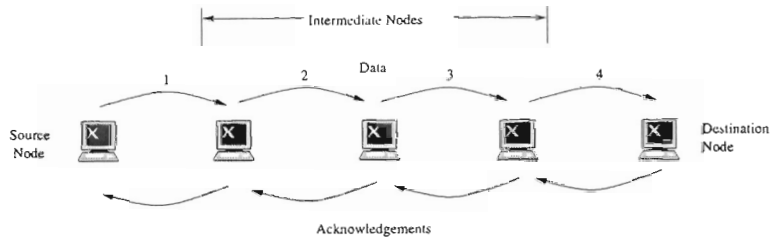

Figure 5.1: TCP traffic and the corresponding acknowledgement packets flowing over four hops to reach a node.

Table 5.1: An increasing number of hops reduces realisable data rates and increases latency.

Number of Hops	Average Measured Throughput (kbps)	Average Measured Latency (ms)
1	572.2	1.02
2	567.3	1.85
3	515.8	2.60
4	454.9	4.75

The results indicate that the rate of traffic throughput decreases when traffic must flow over many hops to reach its destination, but this decrease is not substantial. However, it is clear that if the network was much larger and traffic was routed over a large number of hops then the network performance would decrease to an unacceptable level. In this case the radio range of the nodes would have to be increased, to reduce the total number of hops between source and destination nodes. However, it is unlikely that community networks using a similar architecture would require significantly more hops to reach a destination.

The latency of the traffic increases fairly linearly with the number of hops, but this is to be expected with store-and-forward[1] routing. Large latencies can be problematic when using real-time applications, such as voice or video [49].

[1]Store-and-forward refers to communications systems in which messages are received at intermediate routing points and recorded before being transmitted (forwarded).

Throughput Evaluation 1

The topology of the network for the first selfish-node experiment (Topology A) is shown in Figure 5.2. The nodes implement the flood routing configuration presented in the previous chapter. Two corresponding nodes, that are not in radio range of each other, are connected via an intermediate node with varying selfishness. Ten-thousand 100-Byte datagrams, containing arbitrary data, are sent from the source node to the destination node using TCP (these datagrams are multiplexed into significantly fewer packets by TCP). TCP requires that all messages sent receive an acknowledgment to indicate whether packets arrived correctly. Therefore it is possible that either a packet itself or its corresponding acknowledgement packet could be lost in transit.

In the experiments the selfish intermediate nodes (N_i) are instructed to drop a certain percentage (P_i) of packets. In the first experiment, node N_1 drops packets with $P_1 = \{0, 1, .., 10\}$. This is achieved by inserting the *RandomSample(x)* element into the intermediate node's Click configuration file, and results in $\frac{1}{x}$ of all packets destined for other nodes being discarded.

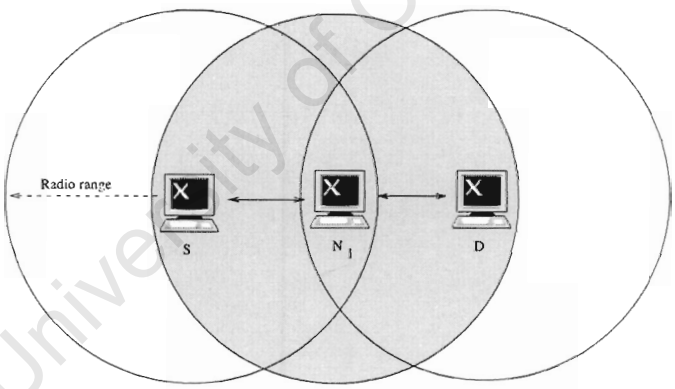

Figure 5.2: Topology A.

The time taken between transmitting the first packet and receiving an acknowledgement for the last packet is shown in Table 5.2. The average throughput results are also graphed in Figure 5.3.

Table 5.2: Average times for Throughput Evaluation 1.

Selfishness P_1 (%)	Average Measured Time (s)	Average Measured Throughput (kbps)
0	14.102	567.3
1	14.215	562.8
2	14.499	551.8
3	15.768	507.4
4	16.191	494.1
5	16.587	482.3
6	20.228	395.5
7	20.417	391.8
8	22.494	355.6
9	26.993	296.4
10	28.006	285.7

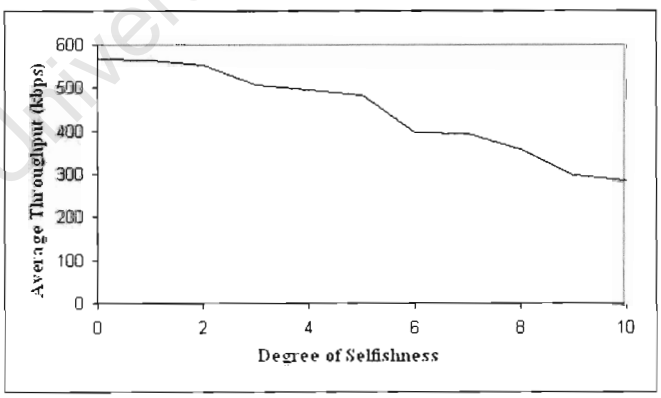

Figure 5.3: Curve showing reducing data rates in Throughput Evaluation 1.

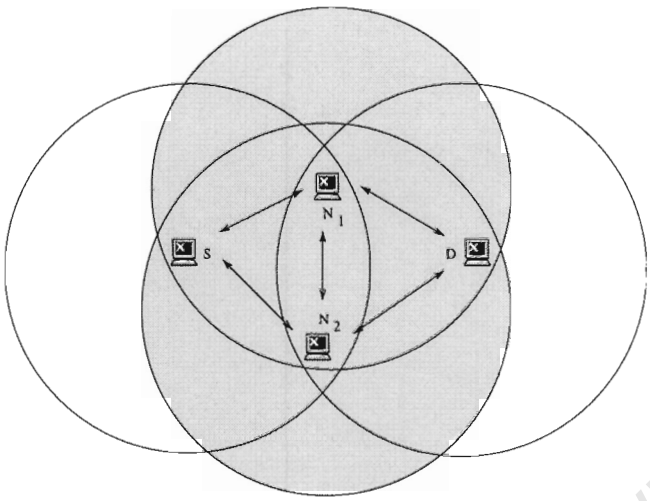

Figure 5.4: Topology B.

Throughput Evaluation 2

The network topology (Topology B) for Throughput Evaluation 2 is shown in Figure 5.4. The radio ranges of the intermediate selfish nodes are highlighted by the grey areas. Note that although the source and destination nodes do not have radio connectivity, the intermediate selfish nodes do. It is possible that in a practical scenario these selfish nodes are acting in collusion to provide poor service to other nodes.

The packets have four possible distinct paths from source S to destination D and vice-versa, shown in Table 5.3, but most routing schemes will always choose the shortest length path [33]. In the case of flood-routing, such as used in this experiment, the packets have a greater chance of reaching the destination because if the one selfish node drops the packet there is still a chance that the other node may forward it. Thus compared to the previous experiment, it is expected that nodes in this topology will provide a better link quality even if they are both selfish.

Table 5.3: Paths between source and destination nodes.

$S \rightarrow N_1 \rightarrow D$
$S \rightarrow N_2 \rightarrow D$
$S \rightarrow N_1 \rightarrow N_2 \rightarrow D$
$S \rightarrow N_2 \rightarrow N_1 \rightarrow D$

Table 5.4: Average Times for Throughput Evaluation 2.

P_1 (%)	P_2 (%)	Average Measured Time (s)	Average Measured Throughput (kbps)
0	100	14.785	541.1
10	90	21.921	365.0
20	80	103.657	77.2
30	70	2384.73	3.4

From the figures in Table 5.4, it can be seen that even adjoining selfish nodes can degrade the network performance to the point where the network's bandwidth tends to zero. As expected however, a 10% selfish node provides a better link quality if it has another node sharing the traffic burden, even though the adjoining node is very uncooperative itself. While flood routing generally degrades traffic throughput, adding unnecessary burden on the network, in this situation it can clearly increase the performance of the end-to-end-link.

5.1.2 Reliability Tests

The reliability of the links in the test-bed framework are evaluated in the following experiments. The experiments are conducted using the same topologies (A and B), depicted in Figure 5.2 and Figure 5.4. These are *Reliability Evaluation 1* and *Reliability Evaluation 2* respectively. The aim of these experiments is to test only the reliability of the paths and not put strain on the links. Thus the Linux *ping* utility is used that sends out one ICMP echo request every second and waits for the ICMP echo response packet. This system is often used to test the connectivity of networks. If the echo request or echo response packets are lost then, unlike in the throughput tests, the nodes do not immediately attempt to retransmit the original packet. This is because it does not have any reliable transport mechanisms. Thus there is no guarantee that the sending nodes will receive a reply from a particular echo request.

Reliability Evaluation 1

In this evaluation ten-thousand echo request packets are sent to a destination node through the intermediate node N_i, using the topology shown in Figure 5.2. The intermediate node exhibits selfishness $P_i = \{0, 1, .., 9\}$ and $P_i = \{10, 20, .., 100\}$. Theoretically the forward-and-return packet loss ratio (plr) for this topology can be calculated as follows:

$$plr = 1 - (1 - P_i)^2$$

The measured average forward-and-return path loss ratio for the run of ten-thousand packets is shown in Table 5.5. The results of the tests are in line with the predicted packet loss. When an intermediate node becomes more selfish over time, the packet loss ratio of the link increases substantially. These values are also graphed in Figure 5.5.

Table 5.5: Average packet loss ratio for Reliability Evaluation 1.

Degree of Selfishness P_1 (%)	Average Measured Packet Loss Ratio (%)
0	0
1	3.12
2	4.91
3	5.75
4	8.18
5	15.61
6	15.97
7	16.02
8	17.01
9	17.69
10	20.53
20	36.45
30	52.90
40	66.61
50	74.90
60	85.64
70	94.89
80	97.88
90	98.60
100	100

Reliability Evaluation 2

In this evaluation the network topology shown in Figure 5.4 is used. Again ten-thousand echo requests are sent to a destination host, and the average packet loss ratio is measured. The results are shown in Table 5.6.

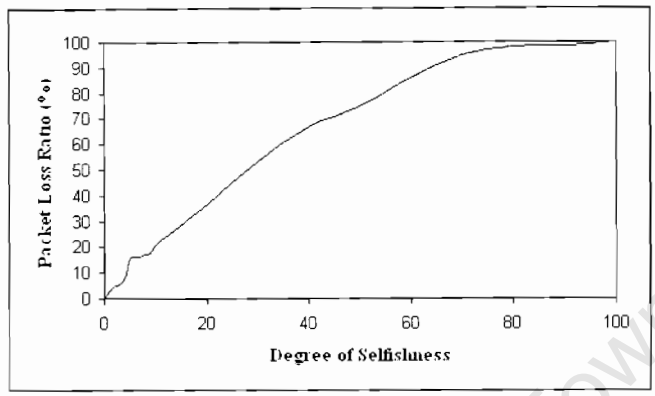

Figure 5.5: Curve showing increasing packet loss ratio in Reliability Evaluation 1.

Table 5.6: Average measured packet loss ratio for Reliability Evaluation 2.

Selfishness P_1 (%)	Selfishness P_2 (%)	Average Measured Packet Loss Ratio (%)
0	100	0
10	90	22.95
20	80	31.01
30	70	40.39
40	60	44.99
50	50	48.51

The results of this experiment show clearly that a large packet loss ratio in a link with a selfish node is mitigated by the presence of another intermediate node (even though this node may also be selfish to some degree). This result indicates that in a tightly connected mesh network of many nodes, the negative effects of selfish nodes on network reliability are reduced.

5.1.3 Throughput and Reliability Discussion

In this section a number of experiments were performed on the evaluation framework to determine the network throughput and reliability, when nodes displaying varying degrees of selfishness were present. Firstly a purely cooperative environment was examined to determine the effects of multiple hops on the traffic throughput. It was found that a reasonable number of hops between corresponding nodes did not severely affect the network performance.

In the next experiment two nodes were joined by a selfish node. It was found that the selfish node reduced the link quality for the corresponding nodes dramatically. When two selfish nodes colluded to provide a lossy link between corresponding nodes, the effects of their selfishness was still very detrimental to the network performance. This effect was worsened further by the unsuitability of TCP for this environment. TCP assumes that packet losses occur due to congestion but in an open multi-agent system packets may be dropped indiscriminately even if there is no congestion. In a wireline network packets are usually only dropped because of congestion. The idea of nodes dropping packets to conserve resources was never conceived when designing the TCP protocol. Thus a transport layer that anticipates such behaviour would be more suitable for these environments, but falls beyond the scope of this work.

When the reliability of the links was measured it was determined that an increasingly selfish node exponentially increases the packet loss ratio of the link. If two selfish nodes were present then their selfishness is mitigated to a certain extent, by each other's presence.

5.2 Cooperation Enforcement Model Evaluation

A novel cooperation enforcement model was proposed in Chapter 3. In order to evaluate the performance of this model over other non-zero sum game strategies, five nodes of

the experimental test-bed are pitted against each other in an *Iterated Prisoner's Dilemma Tournament*. The goal of this tournament is to determine whether non-selfish nodes can out-perform selfish nodes, thus altering the selfish nodes' motivational functions, and enticing them to become cooperative.

In an iterated Prisoner's Dilemma (PD) game, players have only two options: to *cooperate* or *defect*. In terms of an ad-hoc network this would be equivalent to forwarding traffic for the neighbouring nodes, or simply discarding it. The games are played simultaneously such that both players must choose their strategy at the same time. Thus the opponent's strategy is only known to each player once they have already chosen whether to cooperate or defect.

The Prisoner's Dilemma scenario accurately models the interaction of the nodes in an ad-hoc network environment. This is because the interaction between nodes in an ad-hoc network can be thought of as a non-zero sum game. One node's loss is not necessarily another node's gain. If all nodes played cooperatively then the overall payoff for the network would be higher. But there is always the temptation to act selfishly, and conserve one's own resources.

Political scientist Robert Axelrod held the first Iterated Prisoner's Dilemma Tournament in the late 1970's [27]. Contestants in the tournament submitted computer programs that competed in an iterated prisoner's dilemma game of approximately two hundred rounds. The contestants played five games against every other opponent, and after all the games the scores were tallied. In this research the nodes of the evaluation test-bed play a similar game of two hundred rounds. The nodes in the tournament have the potential to adopt different strategies. However, one node in the network always plays the selfish strategy defined below. For the rest of the players it is in their best interests to identify the selfish node and adapt their strategies when playing against it. The tournament objectives are slightly different to those of Axelrod's game. The objective for each node is not only to score highly in each game, but also to have a higher score relative to the selfish node. According to the Equity, Reciprocity and Competition (ERC) model, this helps to eliminate the temptation to be uncooperative.

It is highlighted in previous chapters that nodes might have many reasons for acting in a selfish fashion (defect), but for the sake of these experiments it is assumed that the nodes are acting to preserve their resources, and are not acting maliciously. It is also assumed that nodes participate in routing requests, and limit their selfishness to traffic forwarding.

However, this is not a requirement of the cooperation enforcement model, because the model dictates that information requests are flooded throughout the network.

The tournament is implemented using a software program running on each of the nodes. Details of this program are provided in the appendices. Further information about the tournament is presented below and a selection of playing strategies are highlighted.

5.2.1 Rules and Strategies

The Prisoner's Dilemma payoff matrix was presented in Chapter 3, Table 3.1. For the sake of this tournament, the values of the variables in the payoff matrix are: $u = 3$, $\delta_1 = 2$, $\delta_2 = 2$, and $\delta_3 = 1$. This yields a payoff matrix shown in Table 5.7. The score of each node is simply the overall payoff it receives for the games.

Note that these values satisfy the condition $2u > (2u - \delta_1 + \delta_2 - \delta_3) > (2u - 2\delta_1)$, i.e. the overall payoff (sum of both player's scores) for nodes that cooperate with each other is higher than if one, or both, nodes defect. This payoff is equivalent to the node's utility gained from use of the network $D(k)$, less the nodes dis-utility from wasting resources for other nodes $C(k)$.

Table 5.7: The iterated Prisoner's Dilemma tournament payoff matrix.

	Cooperate	Defect
Cooperate	(3, 3)	(0, 5)
Defect	(5, 0)	(1, 1)

If all nodes play cooperatively all the time then they will each obtain three points for every game. This is the ideal scenario in an ad-hoc network, and indicates a purely cooperative and trustworthy network environment. The points of the games are equivalent to the benefits the nodes receive from the network. A higher score indicates that a node has used very little of its own resources, yet still enjoyed use of the network. A very low score indicates that a node has used the network very little for its own purposes but has had to forward a great deal of traffic for others, and in doing so, has wasted a fair proportion of its resources. Every player maintains a table of its scores against every other player.

The score against each of the opposition nodes are stored separately. After two hundred games are played between the selfish node and the four other nodes in the network, the

scores are divided by the number of games played, giving an average score against a particular opposition node. These scores are listed in the results tables. The final column of the results tables indicates how well the players faired against the selfish node on average.

Each of the nodes in the tournament only starts playing after a random amount of time. This is more realistic because in practical ad-hoc networks nodes may join and leave at any time. As such the players will most likely score differently in their games.

Throughout the different experiments the nodes are assigned various strategies. The strategies to be used in the tournament are described below. The list contains common strategies used in prisoner's dilemma tournaments, as well as the strategies unique to this research.

Selfish Node Strategy

The node using the selfish strategy defects with probability P. However, this defection is not completely random. Nodes will tend to have periods of low resources and thus choose to defect for several games in a row. Thus the defections throughout the whole game are not evenly distributed. The selfish node plays cooperatively for a period and then defects for $100P$ games in a row, but always ensures that the total ratio of defections to games does not exceed P.

Poor Trusting Fool Strategy

A node using the poor trusting fool strategy cooperates in every round of the game, regardless of how its opposition is playing. Nodes that are not selfish, and that do not implement any form of cooperation enforcement scheme, always play the poor trusting fool. It is very easy for selfish players to take advantage of opposition that use this strategy.

Go-by-Majority Strategy

The go-by-majority strategy examines the history of the competitors actions, and tallies the total number of defections and cooperations. If the cooperations outnumber the defections the strategy will cooperate, otherwise it will defect.

Tit-for-Tat Strategy

A node that uses the tit-for-tat strategy cooperates in the first round of the game, and then in every subsequent round the node mimics its opponents previous move. If a competing node defects in round n the tit-for-tat strategy will defect in round $n + 1$, and vice versa. A similar strategy is the tit-for-two-tats strategy. In this strategy a player waits for its opponent to defect twice before defecting, but cooperates immediately after the other node cooperates. The tit-for-two-tats strategy is not implemented in these experiments, but is included for interest.

Since Axelrod held the first PD competition many similar competitions have been held worldwide. Although very simple, it has often been found that the tit-for-tat strategy is the dominant strategy in these tournaments [27].

Trust Rating Strategy

Using the cooperation enforcement model, described in Chapter 3, a node will use trust information gathered from all the other nodes in the network, and from its own experiences. This includes interactive trust, non-interactive trust, rumoured trust and bragging. The node then calculates a composite trust rating for its opponent and chooses its strategy accordingly. With this model a node with very little information about an unknown opposition (a *suspect node*) can still choose an appropriate strategy.

This strategy has an advantage over the other strategies in that it may use information gathered from the other players in the tournament to determine how trustworthy a suspect node is at a particular time. Thus a given node can predict what behaviour to expect from the selfish node and play an appropriate strategy. This full information framework, where nodes have *a priori* knowledge of the other node's behaviour, is the basis of distributed cooperation enforcement.

5.2.2 Poor Trusting Fool Tournament

In this game four nodes play the poor trusting fool strategy, while the selfish node defects with probability $P = \{0.1, 0.2, 0.3\}$. Table 5.8 shows each node's average result from the the three games, where the selfish node is 10%, 20% and 30% selfish respectively, and the other four nodes cooperate 100% of the time.

Table 5.8: Poor Trusting Fool tournament results.

Game 1	1	2	3	4	Average
Selfish Node - 10%	3.21	3.20	3.19	3.18	3.20
Poor Trusting Fool Nodes	2.69	2.68	2.71	2.74	2.71

Game 2	1	2	3	4	Average
Selfish Node - 20%	3.40	3.40	3.40	3.36	3.39
Poor Trusting Fool Nodes	2.43	2.41	2.38	2.48	2.43

Game 3	1	2	3	4	Average
Selfish Node - 30%	3.58	3.58	3.55	3.59	3.58
Poor Trusting Fool Nodes	2.13	2.13	2.12	2.10	2.12

The results show that the poor trusting fool strategy does not stand up well against the selfish node. The selfish node out-performs the other nodes dramatically in all instances, but particularly when the selfish node defects with probability 0.3.

The poor trusting fool tournament demonstrates the danger of not enforcing cooperation in the ad-hoc network environment. The payoffs of the non-selfish nodes tend towards zero, and the total average payoff also drops significantly reducing the overall efficiency of the network.

The poor trusting fool strategy is the default strategy for devices in an ad-hoc network that simply assume their neighbours are cooperative. In this work it has been demonstrated that this assumption may well be wrong. This experiment shows that selfish nodes surrounded by purely cooperative nodes save their own resources, and severely reduce the resources of their neighbouring nodes.

5.2.3 Tit-for-Tat Tournament

In the following series of games the four nodes play the tit-for-tat strategy and the selfish node defects with probability $P = \{0.1, 0.2, 0.3\}$. The results from the tournament are given in Table 5.9. It can be seen from these results that the non-selfish nodes do much better than when they played the poor trusting fool strategy. In addition, while the selfish node generally still scores higher than the non-selfish nodes, this difference is not as great.

Table 5.9: Tit-for-Tat tournament results.

Game 1	1	2	3	4	Average
Selfish Node - 10%	2.86	2.86	2.82	2.86	2.85
Tit-for-Tat Nodes	2.85	2.85	2.81	2.83	2.84

Game 2	1	2	3	4	Average
Selfish Node - 20%	2.63	2.65	2.63	2.66	2.64
Tit-for-Tat Nodes	2.65	2.64	2.63	2.63	2.64

Game 3	1	2	3	4	Average
Selfish Node - 30%	2.46	2.46	2.44	2.45	2.45
Tit-for-Tat Nodes	2.44	2.46	2.44	2.46	2.45

Once again the selfish node outperforms the nodes playing the tit-for-tat strategy. This is most likely because the tit-for-tat players play completely reactively and are often unaware that the selfish node is about to defect. The tit-for-tat strategy ensures that the scores between the players stay fairly equal. This is not necessarily good as the selfish node still has no motivation to cooperate if its payoff is equal to that of a cooperative node.

However, all the scores are lower than in the previous game. The tit-for-tat games show that this strategy does make the network more equitable, i.e. the cooperative and uncooperative are mostly prejudiced equally by uncooperative behaviour. It can be seen, however, the total efficiency (the sum of all players scores) of the network decreases rapidly, as a suspect node becomes increasingly selfish.

5.2.4 Go-by-Majority Tournament

The go-by-majority strategy is fairly simple. It dictates that a suspect node be judged purely on its past performances. Unfortunately, sometimes there is very little information about the past performances of a suspect node. Since the go-by-majority strategy uses only the individual player's experience then it may have incomplete knowledge of the actual trustworthiness of the suspect node. This is made clear by this tournament. In Table 5.10, it can be seen that the selfish node easily out-performs other nodes adopting the go-by-majority strategy.

The reason the go-by-majority strategy doesn't work well is that it takes too long to realise

Table 5.10: Go-by-Majority tournament results.

Game 1	1	2	3	4	Average
Selfish Node - 10%	3.21	3.19	3.17	3.19	3.19
Go-by-Majority Nodes	2.67	2.71	2.74	2.70	2.71

Game 2	1	2	3	4	Average
Selfish Node - 20%	3.41	3.41	3.39	3.40	3.40
Go-by-Majority Nodes	2.40	2.40	2.42	2.41	2.41

Game 3	1	2	3	4	Average
Selfish Node - 30%	3.57	3.57	3.56	3.57	3.57
Go-by-Majority Nodes	2.13	2.15	2.15	2.15	2.15

that a competing node has become selfish, especially if this node has been cooperative for a long time in the past. The assessment of whether or not to trust the suspect node is based purely on long-term statistical evidence and not on the suspect node's current behaviour. An improvement on this strategy would be to place more weight on more recent observations, thus identifying changing behaviour quickly.

5.2.5 Trust Rating Tournament

In these games the nodes choose their strategy according to the trust rating that the cooperation enforcement model assigns their opposition nodes. The ratings are in the range [-1,1]. Therefore if a suspect node has a rating of less than 0, the best strategy is to defect. In the first few rounds of the game the trust rating of a suspect node is calculated from information received from other nodes regarding the suspect node (non-interactive trust and rumours), and what level of trust the suspect node claims (bragging). Once twenty or more games have been played against the suspect node then the trust rating starts to include the scores from these games in its trust rating of the suspect node.

The results from these games are shown in Table 5.11. The four nodes using the cooperation enforcement scheme to determine their strategy score significantly higher than the selfish node. Some scores are much better than others. This is due to the high-scoring players joining the tournament later and having more evidence from their fellow players about the trust rating of the suspect node.

Table 5.11: Trust Rating Strategy tournament results.

Game 1	1	2	3	4	Average
Selfish Node - 10%	2.59	2.66	2.73	2.76	2.69
Trust Rating Nodes	3.01	2.98	3.00	3.00	3.00

Game 2	1	2	3	4	Average
Selfish Node - 20%	2.33	2.20	2.47	2.05	2.26
Trust Rating Nodes	2.93	2.97	2.76	2.88	2.89

Game 3	1	2	3	4	Average
Selfish Node - 30%	2.22	2.13	2.22	2.17	2.19
Trust Rating Nodes	2.68	2.78	2.74	2.80	2.75

The overall efficiency of the network drops as the suspect node becomes more selfish but to a lesser degree than using the other strategies.

5.2.6 Analysis of the Cooperation Enforcement Model Evaluation

Several traditional PD strategies were examined to highlight their weaknesses and strengths against selfish nodes. It was found that none of the traditional PD strategies, for example the tit-for-tat strategy, could score significantly better than the selfish node.

On the other hand, it has been shown that in a practical network environment the cooperation enforcement scheme can successfully identify a selfish node, by assigning it an appropriate trust rating. An accurate trust rating was given to the suspect node by the trust rating strategy by using evidence from their own experiences, and information gathered from other nodes also playing against the opponent. A suitable strategy was adopted against this node and it was defeated in a PD game. Figure 5.6 summarises the results from all of the tournaments. The bars on the left represent the average score of the selfish node, and the adjacent bars on the right shows the average scores of the competing four nodes. These graphs indicate that the cooperation enforcement model, shown in the last frame of the figure, is the most effective strategy tested in this research.

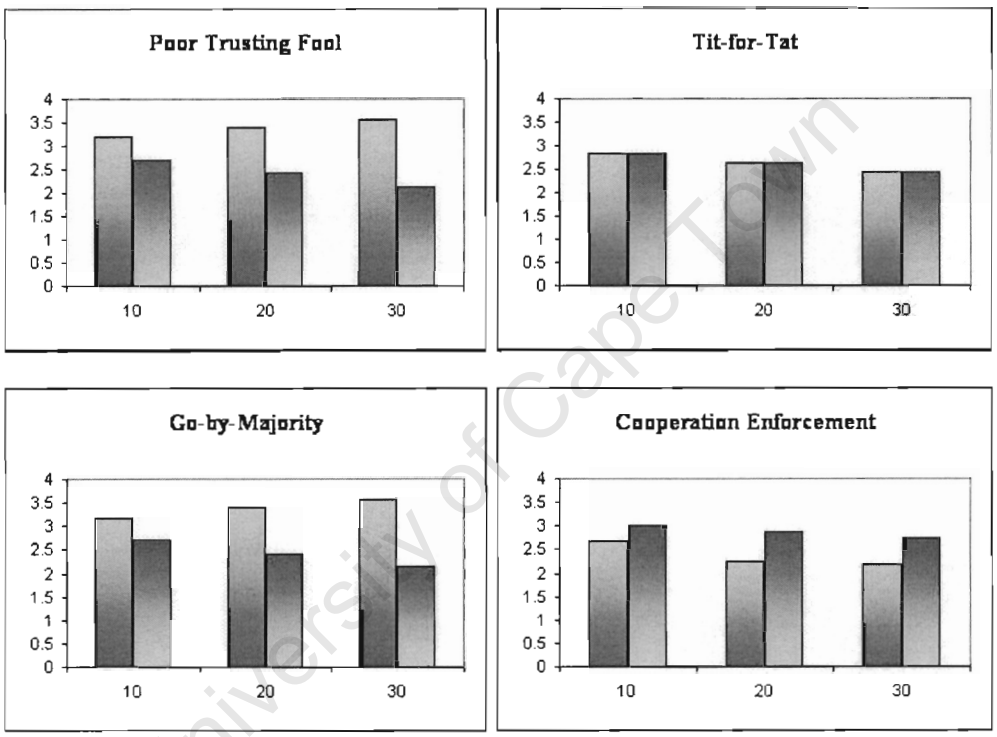

Figure 5.6: Bar graphs showing the average scores from each of the tournaments. The selfish node's scores shown on the left bars and the cooperative nodes' scores adjacent.

5.3 Evaluation of the Trusted Route

The concept of the Trusted Route involves finding a route from the source to destination that incorporates the most trustworthy nodes. The method for calculating the Trusted Route is described in Chapter 4. In this chapter experiments are performed to evaluate the performance of the Trusted Route in an environment that includes selfish nodes. The trusted route serves two purposes:

1. The trusted route will only include the most trustworthy nodes in the network, thereby increasing network throughput and reliability.
2. Selfish nodes are gradually completely excluded from the network. This entices the nodes to become more cooperative as shown in Chapter 3.

The evaluation involves finding the proportion of high-throughput routes found using the trusted route metric. The results of the experiments are compared with the Dynamic Source Routing (DSR) path selection process. Routes are selected in DSR based on minimum hop-count. If two or more routes have the same number of hops then the path is selected randomly between them. The trusted route selects routes based on the average trust rating of the intermediate nodes in the route. Thus routes that incorporate untrustworthy nodes are only used as a last resort.

5.3.1 DSR Routes

Traditional Dynamic Source Routing is evaluated first in the experiments. The sending node initiates a DSR route request for the destination node. The available routes are made known to the destination node via the DSR route discovery mechanisms described in Chapter 4. The destination node then chooses arbitrarily between the minimum-hop routes, and informs the sending node of the route choice.

For the experiments in this chapter the User Datagram Protocol (UDP) is used. TCP is not used because of the congestion control issues highlighted previously in this chapter. UDP is a connectionless transport layer protocol. Once a route has been selected between the corresponding nodes, the source node begins sending a thousand 172-octet UDP packets - one every twenty milliseconds. Once these packets have been sent the destination node arbitrarily chooses another route from its cache of minimum-hop routes, and another

thousand packets are sent. The experiment is run two-hundred times. The destination node notes the rate of received packets for every route chosen.

The full DSR protocol is not implemented for these experiments. A stripped down version including the route discovery mechanisms is utilised. The experiment is designed to show the contrast between traditional DSR, and DSR using trusted routes (described below), and not the maximum performance of the link.

5.3.2 Trusted Routes

The same experiment is repeated as above, except in this second experiment the source node implements the cooperation enforcement model, presented in Chapter 3. Each node in the mesh is given a trust rating T_i by the destination node, normalised to the range [0,1]. A T_i value of 0 represents a node that is guaranteed to drop packets, and 1 represents a node that is guaranteed to cooperate. A DSR route discovery is initiated, but the destination node does not choose arbitrarily between the minimum hop-count routes. Rather, it determines the overall trust rating of the route using the trusted route metric:

$$R = (\sum_{i=1}^{h-1} T_i)(h-1)^{-\frac{3}{2}}$$

The route with the highest overall trust rating R is selected. Again UDP packets are sent from source to destination and the rate of received packets is noted for every run. The experiment is repeated two hundred times.

5.3.3 Trusted Route Topologies

The experimental test-bed is set up so that two corresponding nodes are linked via a static mesh of intermediate nodes, illustrated in Figure 5.7. Lines between the nodes indicate radio connectivity. The intermediate nodes exhibit selfishness S as indicated by the values in the figure. The corresponding nodes have no knowledge of the intermediate nodes, except from what they have observed from their own interactions, and information they have gathered from other agents in the mesh. The intermediate nodes drop packets with probability $S\%$, except *route request* and *route reply* packets which are always forwarded. The minimum-length path between the corresponding nodes in Topology A is four hops, whereas in Topology B it is three hops.

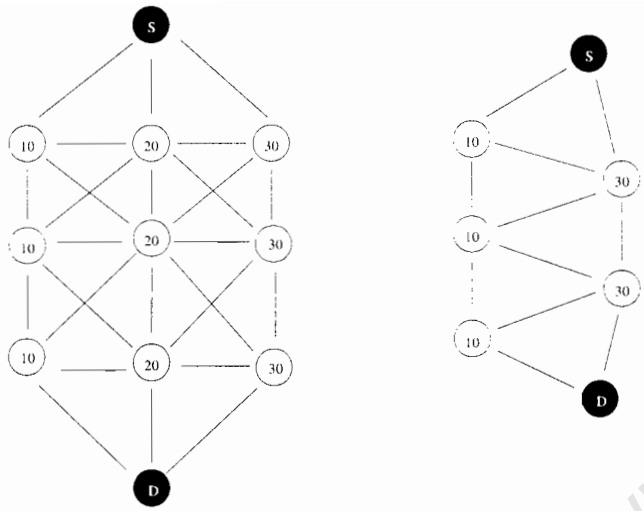

Figure 5.7: Topologies A (left), and B (right), of the test-bed for the trusted route experiments.

5.3.4 Implementation

The DSR protocol was chosen for these experiments as it has been widely used in multi-hop ad-hoc network research. Two experiments are performed using both topologies to compare the effects of using the DSR route discovery algorithm - one with and one without using the trusted route metric. It is expected that the results of these experiments would be similar if the DSDV protocol (discussed in the previous chapter) was used, as it also chooses randomly between minimum length routes.

5.3.5 Trusted Route Results (Topology A)

Using the traditional DSR route selection algorithm it was found that path throughput varied greatly over the different runs. Figure 5.8 shows a histogram plot of the relative frequencies that the different routes achieved a certain throughput. It was found that there was no way to accurately predict what path throughput would be achieved due to the random nature of the DSR route selection process.

When using the cooperation enforcement model, the route selection process was controlled by the trust information gathered by the model. The model correctly identifies the best path through the intermediate nodes and consistently selects this path. The throughput achieved is always reasonable and it is mostly consistent.

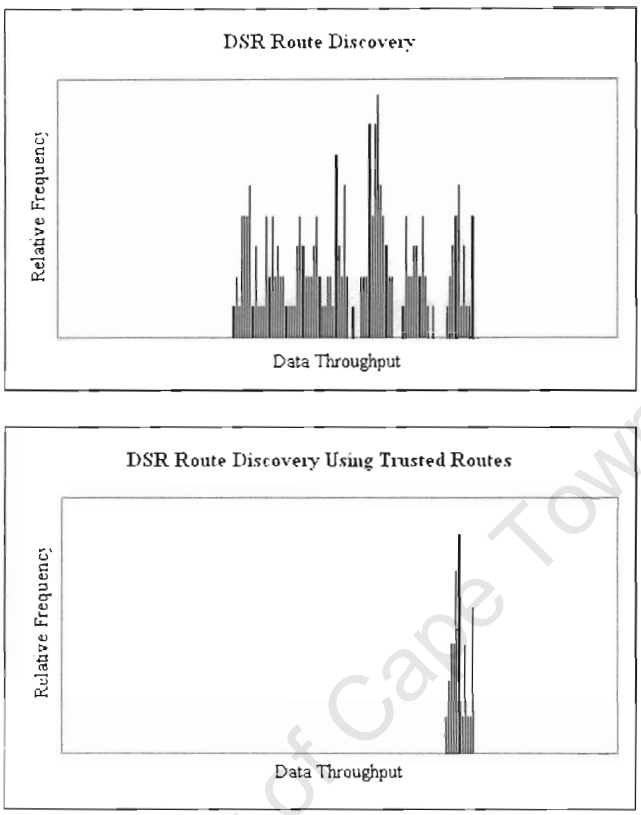

Figure 5.8: Topology A: Histograms showing the relative frequency that different throughput routes are chosen with DSR route discovery (top) with and trusted routes (bottom).

Table 5.12: Topology A: Comparison of DSR Routes vs. Trusted Routes (in kbps).

	DSR Routes	Trusted Routes
Mean	297.67	410.51
Median	296.30	410.12
Minimum	178.46	395.25
Maximum	425.78	424.50
Standard Deviation	69.89	8.12

The data from the experiments is summarised in Table 5.12. From this summary it can be seen that the mean path throughput when using trusted routes is substantially higher than when using DSR's arbitrary minimum length routes. It can also be seen that the standard deviation of the trusted route throughput is much lower. This is because the source route will generally choose the same path if there is no change to the intermediate node's trust ratings. This ensures that the link characteristics do not vary greatly over time.

5.3.6 Trusted Route Results (Topology *B*)

Topology *B* is used for this experiment to present a situation where several *viable* different length paths exist through the mesh. This experiment was performed to show that the trusted route metric can identify routes that provide significantly higher throughput than routes relying on the minimum hop-count metric.

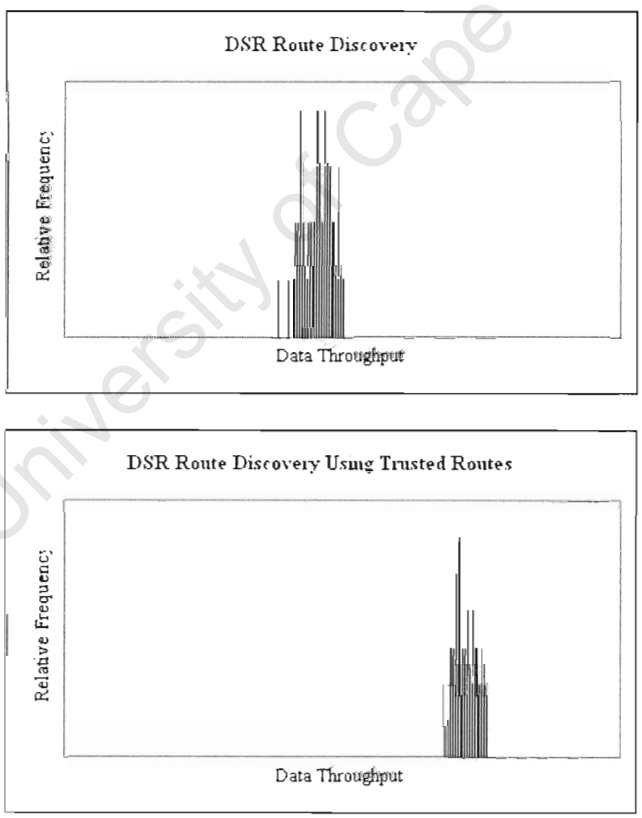

Figure 5.9: Topology B: Histograms showing the relative frequency that different throughput routes are chosen with DSR route discovery (top) with and trusted routes (bottom).

Figure 5.9 shows the large proportion of high-throughput routes chosen when using the trusted route metric, and a summary of the results from the experiments is given in Table 5.13. The results indicate that the average throughput of the routes discovered using the trusted route metric is significantly higher than those found using traditional DSR route discovery.

Table 5.13: Topology B: Comparison of DSR Routes vs. Trusted Routes (in kbps).

	DSR Routes	Trusted Routes
Mean	260.36	412.51
Median	262.55	409.23
Minimum	222.51	390.98
Maximum	287.14	420.03
Standard Deviation	15.87	8.95

To determine why this happened the trusted route metric can be calculated for each path. Looking at Topology B (Figure 5.7) it is clear that two viable path choices exist. The path down the left of the diagram is longer but more trustworthy. The path down the right of the diagram is shorter but less trustworthy. All other choices of paths will have either more hops, or be less trustworthy. Traditional DSR chooses the minimum hop path on the right. Clearly the trusted route metric has chosen a different path as the throughput results differ greatly. If the trusted route equation is examined, it can be seen that the shorter route through the two nodes on the right of the topology diagram, yields the following metric:

$$T_R = \frac{2(0.7)}{\sqrt{2^3}} = 0.495$$

Whereas, the longer route through the three more cooperative nodes on the left of the diagram results in the metric:

$$T_R = \frac{3(0.9)}{\sqrt{3^3}} = 0.520$$

Thus the route discovery using the trusted route metric chooses the longer, but more cooperative route. This results in a higher throughput path between the source and destination nodes, and accounts for the better results achieved when using the trusted route.

5.3.7 Trusted Route Discussion

The benefits for using a cooperation enforcement scheme were highlighted in previous chapters with regards to enticing the cooperation of selfish nodes. In this chapter it has been shown that such a scheme also provides a mechanism for finding trusted routes in an ad-hoc network.

Ad-hoc routing protocols, such as DSR and DSDV, select the route between corresponding nodes purely on the distance or hops between them. If two routes of equal distance exist then the choice of route is random. It is proposed in this research that alternative metrics be used to calculate the best path. Specifically this work has shown that by using trust information, gained by measuring the selfishness of intermediate nodes, it is possible to determine a trusted route.

It has been shown that routes selected based on the trust rating of intermediate nodes can offer significantly higher data throughput as opposed to routes which do not use this information.

5.4 Summary of all Evaluation Results

In this chapter several evaluations are performed on the experimental framework that demonstrate the detrimental effects of uncooperative nodes in an ad-hoc network environment. However, it has also been shown that these effects can be mitigated by employing a distributed cooperation enforcement scheme that assigns trust ratings to the nodes in the network. If it is known that there is a high probability that a node will act selfishly, then the node can be punished, excluded from the network, and omitted from any routing decisions.

In the first set of experiments it was shown that selfish nodes severely reduce both the reliability and the throughput of network links. It was determined that much of the decrease in throughput can be attributed to TCP being unsuitable for ad-hoc wireless environments.

In the second set of experiments it was shown that the trust rating assigned by the distributed cooperation enforcement model, accurately identifies selfish nodes in a network. Using this information the nodes played a Prisoner's Dilemma Tournament, and it was

found that nodes basing their strategies on trust information performed best against selfish nodes.

In the final experiment it was shown that the trust rating can also be used to calculate a trusted route metric for use with the Dynamic Source Routing Protocol. DSR selects the best path through an ad-hoc network based on minimum hop count, whereas the trusted route is a path with the most trustworthy nodes (but still gives preference to shorter routes). It was shown that over a period of time, the trusted route could regularly provide high-throughput routes, even if very selfish nodes were present. It was also shown that the trusted route metric could successfully identify longer routes that had higher overall throughput, than the minimum hop-count routes.

Chapter 6

Conclusions and Recommendations

6.1 Conclusions

This study has investigated the issues surrounding selfish nodes in community ad-hoc wireless networks. Specifically, this work has addressed cooperation enforcement based on a distributed trust mechanism that punishes uncooperative nodes.

It was found that several ad-hoc network cooperation enforcement techniques have been proposed in the literature. However, none of these techniques have yet been applied to practical networks. Thus a goal of this work was to implement a cooperation enforcement scheme on an evaluation framework, and present experimental results. In order to achieve this a novel trust-based cooperation model was introduced. The model utilised four types of trust information to assign an accurate trust rating of individual nodes in the network. A brief analytical overview of node preference structures was presented, to show that nodes are inclined to cooperate with each other in an ad-hoc network based on both absolute and relative payoffs.

An evaluation framework architecture was developed to implement the cooperation enforcement scheme. The architecture was implemented on machines using 802.11 hardware devices. The test-bed was able to successfully perform the evaluations required in this work. Useful results were obtained that confirmed the detrimental effects of selfish nodes in ad-hoc networks. Based on the findings in the preceding chapters, the following conclusions are drawn:

- Existing community ad-hoc networks are prone to cheating by selfish nodes. Nodes

can conserve their own resources by refusing to forward traffic in the network. However, this has detrimental effects on the reliability and throughput of the network.

- Cooperation enforcement techniques can be applied to combat node selfishness in ad-hoc networks. Several of these techniques rely on existing infrastructure (e.g. authentication servers), or tamper-proof hardware modules. The requirement of existing infrastructure is not suitable for ad-hoc networks, as these networks are distributed and are neither owned nor controlled by a single stakeholder. Tamper-proof hardware modules are not feasible as they add cost and added complexity to the host machines. Therefore, a distributed trust-based cooperation enforcement mechanism is the best solution to ensure node cooperativeness.

- In order to predict the behaviour of selfish nodes economic principles must be applied. The Equity, Reciprocity and Competition model can be used to predict the behaviour of network nodes, and the Prisoner's Dilemma scenario is useful to model the payoff structures observed in ad-hoc networks. Game theory must be utilised to predict the equilibria points, where a certain number of nodes in the network cooperate and the others defect.

- The choice of transport layer protocol in a community ad-hoc network is very important. The Transmission Control Protocol (TCP) utilises various mechanisms to control congestion, including exponentially increasing the retransmission timer. These mechanisms do not take into account that packets could be dropped for reasons other than excessive congestion. Thus when selfish nodes drop packets the TCP mechanism slows the data rate unnecessarily, resulting in poor network throughput performance. Thus TCP is not suitable for these environments.

- The novel trust-based cooperation enforcement technique proposed in this work successfully identified selfish nodes. If selfish nodes are punished, by denying them network services and excluding them from the network, then they will be inclined to become cooperative. In order for trust ratings to be fair and accurate, several types of trust information must be used. Trust ratings should also give more relevance to recent observations, to detect any sudden changes in a node's cooperativeness.

- Ad-hoc routing protocols, such as Dynamic Source Routing (DSR) and Destination Sequenced Distance Vector (DSDV), only take into account the number of hops between the source and destination nodes when determining the best path. In networks

where selfish nodes are present DSR and DSDV perform poorly because it is preferable to omit uncooperative nodes from the routing decision. This can be achieved by employing a path metric such as the Trusted Route to find the path with the most trustworthy nodes and consequently the best throughput.

The proposed cooperation enforcement scheme was successfully implemented on an evaluation framework. Various evaluations showed that the cooperation enforcement scheme could effectively identify selfish nodes in an Iterated Prisoner's Dilemma tournament. In addition, the trust ratings applied by the cooperation enforcement could be used by the trusted route metric. DSR implementing the trusted route metric was more effective in finding high-throughput routes in the test-bed environment than traditional DSR.

6.2 Recommendations and Future Work

While conducting this work a number of issues surrounding ad-hoc network and this project itself were raised. These issues are presented here for the consideration of future research in this field.

- This work has focussed primarily on the problem of node selfishness in ad-hoc networks. This a relatively new field of research, and not much is known about the tendency for this to happen. It is envisaged that community ad-hoc networks will become widespread in the near future, and these networks will be used for Internet connectivity, voice applications and file transfers. Further research must be carried out to find out what need there is for cooperation enforcement in community networks.

- The evaluation of the behaviour of selfish nodes in this work, has been primarily based on economic models. These models may or may not provide a true representation of node behaviour. The assumption throughout this paper is that nodes will try to act in a rational manner, and hence will try to conserve their resources as much as possible. Unfortunately it is the end-users that decide how their devices will operate, and end-users are not known for acting in a rational fashion. Thus the rational node assumption may be flawed and further work is required to see exactly how end-users will attempt to modify devices in order to protect the device's resources.

- The proposed cooperation enforcement model was use for simplicity, and was primarily based on research by other authors. Further research is required to determine what form of cooperation enforcement is optimal. While a trust-based scheme is desirable in terms of it not requiring any additional hardware modules, or relying on some form of infrastructure, it is also the most likely to be thwarted by selfish users. Network security is an ongoing process and must constantly adapt to new threats.

- The analysis of the cooperation enforcement model relied heavily on the Equity, Reciprocity and Competition (ERC) model to analyse a node's preference structure. This model is generally reserved for human preferences but was used because it was assumed that end-users would control their devices. If it transpires that the devices themselves are the ones that control their own preference structure then research must be done into the algorithms that dictate these preferences.

- The experimental framework design can be improved. At the present the host machines in the test-bed are very slow. The 802.11b protocol used only transmits at a maximum of 11 Mbps and will soon become obsolete, especially with the introduction of the 802.11n protocol. Thus currently the test-bed does not properly illustrate the true performance that can be achieved in ad-hoc networks. Future work includes upgrading the test-bed framework, and implementing more complex routing algorithms in order to examine the network's potential. Recently newer versions of the Click Modular Router have been released that support the Linux 2.6 kernel. Once sufficient testing has been performed on this new Click version the test-bed can be upgraded to take advantage of the 2.6 kernel features.

- It was shown that TCP is a poor choice of protocol for ad-hoc wireless networks, thus a more appropriate transport-layer protocol should be developed for this environment. However, the nodes in ad-hoc networks would then be unable to connect to traditional wireline hosts using TCP. A suitable architecture of the gateway node that links the two types of networks must be investigated.

- The proposed 802.11s mesh networking standard will no doubt extract ideas from existing ad-hoc 802.11 test-bed architectures. The test-bed framework should be used in the future to address issues surrounding the design and roll-out of the 802.11s standard due to be ratified in 2008. Mechanisms to combat node selfishness, such as the ones discussed in this work, should form part of the 802.11s architecture.

- The concept of the trusted route needs to be further researched. It is primarily based on the Expected Transmission Count Metric (ETX), but the ETX metric simply uses the sum of all link-level transmissions to determine the best route. The trusted route uses a slightly more complex equation, which appears to fit nicely. The aim of the trusted route was to prove that DSR could be improved by taking into account the trustworthiness of nodes in the network. However, more research and thorough testing are required in order to determine the optimum trusted route equation. A weighted sum (of the trustworthiness of the route and the number of hops), may prove to be a better metric.

- This work briefly examined the link-layer characteristics of multi-hop ad-hoc networks. It was decided to use a contention-based scheme. Further research should be conducted to determine what link-layer protocols most successfully facilitate fair access to the wireless medium, particularly in dense mesh networks.

- Apart from ensuring node cooperativeness, this research has not examined the security issues of having nodes forward each other's traffic. It would be fairly trivial for an intermediate node to corrupt data travelling between communicating nodes, unless effective encryption and authentication mechanisms are employed.

- Ad-hoc networks that aim to support voice and video applications will need to offer quality of service guarantees. Further work must be carried out to determine whether such applications can be supported in ad-hoc networks, particularly if selfish nodes are present.

Bibliography

[1] P. Chaudhury, W. Mohr, and S. Onoe, "The 3GPP Proposal for IMT-2000," *IEEE Communications Magazine*, 1999.

[2] B. Crow, I. Widjaja, L. Kim, and P. Sakai, "IEEE 802.11 Wireless Local Area Networks," *IEEE Communications Magazine*, vol. 35, pp. 116–126, September 1997.

[3] J. Haartsen, "The Bluetooth Radio System," *IEEE Personal Communications*, vol. 7, pp. 28–36, February 2000.

[4] A. Urpi, M. Bonuccelli, and S. Giordano, "Modelling Cooperation in Mobile Ad-hoc Networks: A Formal Description of Selfishness," *wiOpt*, 2003.

[5] R. Powell, "Bargaining Theory and International Conflict," *Annual Review of Political Science*, vol. 5, pp. 1–30, June 2002.

[6] R. Morris, F. Kaashoek, D. Karger, D. Aguayo, J. Bicket, S. Biswas, D. D. Couto, and J. Li, "The Grid Ad-Hoc Networking Project." http://www.pdos.lcs.mit.edu/grid/.

[7] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris, "Link-level Measurements from an 802.11b Mesh Network," *SIGCOMM*, August 2004.

[8] E. Kohler, "The Click Modular Router." http://www.pdos.lcs.mit.edu/click/.

[9] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, "The Click Modular Router," *ACM Transactions on Computer Science*, vol. 3, pp. 263–297, August 2000.

[10] D. Waiting, N. Ventura, and S. Shepstone, "An Experimental Multi-Hop Ad-Hoc Network," *South African Telecommunication Networks and Applications Conference*, 2004.

[11] M. Pearson, "The CRCNet." http://www.crc.net.nz/.

[12] S. Marti, T. Giuli, K. Lai, and M. Baker, "Mitigating Routing Misbehavior in Mobile Ad Hoc Networks," 2000.

[13] T. D. Huynh, N. R. Jennings, and N. Shadbolt, "FIRE: An Integrated Trust and Reputation Model for Open Multi-Agent Systems," *Proc. 16th European Conference on Artificial Intelligence*, 2004.

[14] P. Michiardi and R. Molva, "Core: A Collaborative Reputation Mechanism to Enforce Node Cooperation in Mobile Ad-hoc Networks," *Proceedings of the IFIP TC6/TC11 Sixth Joint Working Conference on Communications and Multimedia Security: Advanced Communications and Multimedia Security*, pp. 107–121, 2002.

[15] P. Michiardi and R. Molva, "A Game Theoretical Approach to Evaluate Cooperation Enforcement Mechanisms in Mobile Ad-hoc Networks," *WiOpt'03: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, 2003.

[16] S. Buchegger and J.-Y. L. Boudec, "Performance Analysis of the CONFIDANT Protocol (Cooperation Of Nodes: Fairness In Dynamic Ad-hoc NeTworks)," *Proceedings of IEEE/ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC)*, 2002.

[17] B. Lamparter, M. Plaggemeier, and D. Westhoff, "Estimating the Value of Cooperation Approaches for Multihop Ad Hoc Networks," *Elsevier Journal of Ad Hoc Networks*, 2004.

[18] L. Buttyán and J.-P. Hubaux, "Stimulating Cooperation in Self-Organizing Mobile Ad Hoc Networks," 2001.

[19] L. Buttyán and J.-P. Hubaux, "Enforcing Service Availability in Mobile Ad-Hoc WANs," *Proceedings of the First IEEE/ACM Workshop on Mobile Ad Hoc Networking and Computing (MobiHOC)*, 2000.

[20] S. Zhong, J. Chen, and R. Yang, "Sprite: A Simple, Cheat-Proof, Credit-Based System for Mobile Ad-Hoc Networks," 2002.

[21] G. E. Bolton and A. Ockenfels, "A Theory of Equity, Reciprocity and Competition," *American Economic Review*, vol. 90, no. 1, pp. 166–193, 2000.

[22] T. Nguyen and N. Jennings, "A Heuristic Model of Concurrent Bi-Lateral Negotiations in Incomplete Information Settings," *Proc. 18th Int. Joint Conf. on AI, Acapulco*, 2003.

[23] J. von Neumann and O. Morgenstern, *Theory of Games and Economic Behavior.* Princeton University Press, 1947.

[24] S. R. Thampuran, "Routing Protocols for Ad Hoc Networks of Mobile Nodes," *Annual International Phoenix Conference for Computers and Communications*, 1996.

[25] P. Michiardi and R. Molva, "Simulation-based Analysis of Security Exposures in Mobile Ad Hoc Networks," *European Wireless 2002 Conference*, 2002.

[26] D. Gambetta, "Can We Trust," *Trust: Making and Breaking Cooperative Relations*, pp. 213–237, 2000.

[27] R. Axelrod, "The Evolution of Strategies in the Iterated Prisoner's Dilemma," *Genetic Algorithms and Simulated Annealing*, 1987.

[28] L. Telser, *Competition, collusion, and Game Theory.* MacMillan, 1972.

[29] T. L. Turocy and B. von Stengel, "Game Theory," *Encyclopedia of Information Systems*, 2002.

[30] C. A. Holt and A. E. Roth, "The Nash Equilibrium: A perspective," *Proceedings of the National Academy of Science of the United States of America*, 2004.

[31] M. Shor, "Game Theory Dictionary." http://www.gametheory.net, 2003.

[32] C. Perkins and P. Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers," *Comp. Comm. Rev.*, 1994.

[33] D. Johnson and D. Maltz, "Dynamic Source Routing in Ad-hoc Wireless Networks," *Mobile Computing, T Imielinski and H Korth, Eds.*, 1996.

[34] D. S. D. Couto, D. Aguayo, J. Bicket, and R. Morris, "A High-Throughput Path Metric for Multi-Hop Wireless Routing," *The Proceedings of the 9th ACM International Conference on Mobile Computing and Networking (MobiCom '03), San Diego, California*, 2003.

[35] "GNU Zebra." http://www.zebra.org.

[36] D. Aguayo, J. Bicket, S. Biswas, D. S. J. D. Couto, and R. Morris, "MIT Roofnet." http://www.pdos.lcs.mit.edu/roofnet/.

[37] M. Rahnema, "Overview of the GSM System and Protocol Architecture," *IEEE Communications Magazine*, vol. 31, pp. 92–100, April 1993.

[38] M. Conti, "Body, Personal, and Local Ad Hoc Wireless Networks," *The Handbook of Ad Hoc Wireless Networks*, 2003.

[39] S. Chakrabarti and A. Mishra, "Quality of Service in Mobile Ad Hoc Networks," *The Handbook of Ad-hoc Wireless Networks*, 2003.

[40] C. Systems, "Aironet 350 Client Adaptor Data Sheet." http://www.cisco.com/.

[41] J. Tamminen, "2.4 GHz WLAN Radio Interface," *Radionet Oy*, 2002.

[42] S. Savage, D. Wetherall, A. Karlin, and T. Anderson, "Practical Network Support for IP Traceback," *Proceedings of the 2000 ACM SIGCOMM Conference*, August 2000.

[43] C. Kent and J. Mogul, "Fragmentation Considered Harmful," *Proceedings of the 1987 ACM SIGCOMM Conference*, pp. 390–401, August 1987.

[44] J. Mogul and S. Deering, "Path MTU Discovery," *Request for Comments 1191*, November 1990.

[45] J. Postel, "Transmission Control Protocol," *Internet Request for Comments RFC-793*, 1981.

[46] J. Postel, "User Datagram Protocol," *Internet Request for Comments RFC-768*, August 1980.

[47] A. S. Tanenbaum, *Computer Networks.* Prentice Hall, Inc., 3 ed., 1996.

[48] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz, "A Comparison of Mechanisms for Improving TCP Performance over Wireless Links," *IEEE/ACM Transactions on Networking*, vol. 5, December 1997.

[49] U. Varshney, A. Snow, M. McGivern, and C. Howard, "Voice Over IP," *Communications of the ACM*, vol. 45, January 2002.

[50] G. Charness and M. Rabin, "Social Preferences: Some Simple Tests and a New Model," *Econometric Soceity World Congress*, 1999.

[51] T.-W. Chen and M. Gerla, "Global State Routing: A New Routing Scheme for Ad-hoc Wireless Networks," *Proceedings of the IEEE ICC*, 1998.

[52] M. Joa-Ng and I. Lu, "A Peer-to-peer Zone-Based Two-Level Link State Routing for Mobile Ad Hoc Networks," *IEEE Journal on Selected Areas in Communications, Special Issue on Ad-hoc Networks*, 1999.

[53] C. E. Perkins, E. M. Royer, and S. R. Das, "Ad Hoc On-demand Distance Vector Routing," *IETF Draft*, 1999.

Appendix A

Further Information on the Click Modular Router

The Click router configuration used in this work routes packets throughout the ad-hoc network eliminating loops and duplicate packets. However, Click can be used for many different router configurations, because of its modular and flexible architecture. In this appendix further information is presented regarding the functioning of Click and the elements used in this work.

A.1 Click Elements

Click configurations are made of *elements* linked together using *connections*. Elements perform simple router functions like scheduling, packet classification queueing and interfacing with network devices. They have input and output ports. These ports are either of type *push*, *pull* or *agnostic*. A push connection relies on the source element pushing a packet to a downstream element. With a pull connection the destination element pulls packets from the element upstream. It is illegal to form a connection between push and pull ports. Agnostic ports can be either push or pull depending on ports they are connected to.

A.2 Element Description

In the table below further documentation is provided for the elements used in the router configurations, including the two new elements, *SeenBefore()* and *RandomID()*.

Only the type of the elements and a brief description is included. Complete and current documentation is available on the Click website [8].

Element Name	Type	Description
ARPRessponder	Agnostic	Input is ARP request packets. Returns an ARP reply if it knows the answer.
CheckIPHeader	Agnostic	Checks that the packet's length is reasonable, and that the IP version, header length, and checksum fields are valid.
Classifier	Push	The Classifier has N outputs, each associated with a corresponding packet from the configuration string. The patterns are scanned in order, and the packet is sent to the output corresponding to the first matching pattern.
DecIPTTL	Agnostic	Decrements IP time-to-live field. Drops dead packets.
Discard	Agnostic	Discards all packets received on its single input.
EtherEncap	Agnostic	Encapsulates packets in the Ethernet headers specified by its arguments.
FromDevice	Push	Intercepts packets from a specified Linux interface and pushes them to output 0
GetIPAddress	Agnostic	Copies destination address from packet and sets the destination IP address annotation.
MarkIPHeader	Agnostic	Marks packets as IP packets by setting the IP header annotation.
Queue	Pull	Stores incoming packets in a FIFO queue of a specified length.
RandomID	Agnostic	Sets the identification field of IPv4 packets to a random 16-bit number.

RandomSample	Agnostic	Drops incoming packets with a given probability.
SeenBefore	Agnostic	Stores the last N packet identification fields. Outputs recently seen packets on output 1 and all other packets on output 0.
SetPacketType	Agnostic	Sets the packet type annotation. The packet type annotation tells Linux about the packet's link level characteristics (e.g. was it for this host, or a broadcast packet, etc.).
SetIPChecksum	Agnostic	Expects an IP packet as input. Calculates the IP header's checksum and sets the checksum header field.
Strip	Agnostic	Deletes the first N bytes from each packet. Used to strip off Ethernet headers.
ToDevice	Pull	Pulls packets from its single input and sends them out the specified Linux interface.
ToHost	Push	Hands packets to the ordinary Linux protocol stack. Expects packets with Ethernet headers.

A.3 Click Syntax

Every element has an element class that is specified by its name and optional arguments [8]. Elements are connected by their input and output ports. In the Click language an element declaration is made as follows,

name :: class(config-string);

and a connection between elements is written as

name1 [port1] -> [port2] name2;

The two constructs above are sufficient to describe any configuration graph. Elements are always declared before they are used in connections. Below a simple example is given of a

Click configuration that takes in packets, decreases the time-to-live field and outputs them on another interface. Note that when port numbers are omitted they are assumed to be zero, and empty configuration strings can be omitted.

// First declare the elements

src :: FromDevice(eth0);

decTTL :: DecIPTTL;

outputQueue :: Queue(1024);

dest :: ToDevice(eth1);

// Then make connections between them

src -> decTTL -> outputQueue -> dest;

Alternatively, the router could have been defined as:

FromDevice(eth0) -> DecIPTTL -> Queue(1024) -> ToDevice(eth1);

Both the above configurations are valid in Click and demonstrate how flexible and simple the language is.

A.4 Element Implementation

In this work it was necessary to implement new elements to perform different tasks. Every Click element is simply a subclass of the C++ class *Element*, which has about twenty virtual functions. An example element class is presented below:

class NullElement: public Element {

public:

NullElement() { add_input(); add_output(); }

*const char *class_name() const { return "Null"; }*

*NullElement *clone() const { return new NullElement; }*

*const char *processing() const {return AGNOSTIC; }*

*void push(int port, Packet *p) { output(0).push(p); }*

*Packet *pull(int port) (return input(0).pull(); }*

};

The above element, simply passes packets unchanged from its single input to its single output.

Appendix B

Evaluation Software

This appendix details the software utilised to conduct the various evaluations detailed in Chapter 5. Four different utilities were used: a UDP traffic generator, a TCP traffic generator, the Linux ping utility, and a program to facilitate the Prisoner's Dilemma Tournament.

B.1 UDP Traffic Generator

dgram-gen.c

The *dgram-gen* and *dgram-rec* utilities were written in the C programming language by the author. It takes the following arguments: destination address, port number, packet size, number of packets, and time between sending packets.

The program sends User Datagram Protocol (UDP) datagrams at the specified rate to the specified destination IP address and port. Note that the destination IP address is that of the final destination node and not the next-hop node in the ad-hoc network.

dgram-rec.c

The *dgram-rec* program receives UDP datagrams on a specified port and prints the contents of the packets to the screen. It counts the number of received packets and the packet jitter.

B.2 TCP Traffic Generator

stream-gen.c

This program takes in the same arguments as the UDP generating program. It generates a stream of Transmission Control Protocol (TCP) traffic, and measures the time taken between sending the first datagram and receiving an acknowledgement for the last datagram sent. The *stream-gen* and *stream-rec* utilities were written by the author.

stream-rec.c

This utility waits on a specified port for incoming TCP traffic and sends acknowledgements for each successfully received datagram.

B.3 Linux Ping Utility

ping

The Ping utility makes use of ICMP's ECHO_REQUEST datagrams to elicit an ECHO_RESPONSE packet from a host machine. The ping utility measures the time taken between sending the ECHO_REQUEST and receiving the ECHO_REPLY. This is a simple tool to determine the presence of a path to a particular host, and the expected latency of traffic between two hosts. This utility is standard with most distributions of Linux.

B.4 Iterated Prisoner's Dilemma Tournament Software

ipd.c++

The prisoner's dilemma game runs on every node during the tournament. As soon as the program executes, the node announces itself to every other player in the network by issuing an announcement packet. The other nodes then add the new node to their list of known nodes in the network.

Every node then proceeds to play a prisoner's dilemma game against another player in their table of known nodes. If a node is playing using the trust strategy then it issues a trust metric request to the rest of the nodes in the network. The node that is the subject of the trust request replies with its bragging information, while everyone else offers their

respective interactive trust information about the suspect node. The replies are tallied by the requesting node, and its strategy is chosen accordingly.

If a node finds that a suspect node's trust rating has dropped above or below a certain threshold then it initiates a rumour packet. Every packet can be identified by the first character, shown in Table B.1.

Table B.1: Prisoner's Dilemma Tournament packets.

Packet Identifier	Packet Type
1	Announce myself
2	Request trust information about a suspect node
3	Reply with trust information about a suspect node (or bragging about myself)
4	Initiate a rumour
5	Initiate a Prisoner's Dilemma game
6	Respond to a Prisoner's Dilemma Game

The nodes all utilise a packet format that identifies the sending node, and the suspect node's IP addresses. The relevant metrics or game strategies, are also included in the packet. The PD packets are encapsulated in UDP packets, and sent via the flood mechanisms outlined in Chapter 4.

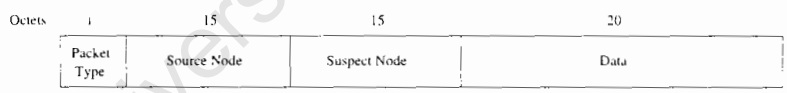

Figure B.1: The PD packet format.

Every node maintains it own scores against its opponents. The program's only argument is the particular strategy that the node is playing. This software was written by the author in the C++ programming language.

Appendix C

Further Information on the ERC Model

The basic premise of the Equity, Reciprocity and Competition (ERC) model is that individuals are motivated by pecuniary payoffs and their relative payoff standing [21]. The ERC model describes the behaviour of players in laboratory games where reciprocity is thought to be a factor, such as in iterated Prisoner's Dilemma (PD) games. There are distinct parallels between the analogy of the PD and the behaviour of nodes in ad-hoc networks. In this appendix more information is given about the specifics of the ERC model.

C.1 Reciprocity

Several studies into social preferences have indicated that individuals are not only driven by absolute payoffs. They are also driven by reciprocity. In particular some individuals are willing to sacrifice an increase in payoff for all recipients, but especially for the lowest-payoff recipients [50]. However, people are reluctant to sacrifice their own payoffs to reciprocate good or bad behaviour beyond what they would sacrifice for neutral parties. When these other parties are unwilling to sacrifice payoffs themselves, people withdraw willingness to sacrifice for them to achieve a fair outcome.

C.2 Finding the Equilibrium

In a two-player game (such as the Prisoner's Dilemma) a player is motivated by the following function:

$$v_i(c\sigma_i, \sigma_i) = a_i c\sigma_i - \frac{b_i}{2}(\sigma_i - \frac{1}{2})^2;\ a_i \geq 0,\ b_i > 0$$

The first term represents the individual's preference for pecuniary payoffs. The second term delineates the influence of the comparative effect. Thus as a player deviates further from its opposition's payoff the comparative effect is greater. A player's type is characterised by the marginal rate of substitution between absolute and relative payoffs. This ERC preference is equal to a/b.

Where the ERC model and the distributed cooperation enforcement model differ is that with ERC the players have incomplete information. Thus the nodes do not know the preference structure of the other nodes. In the cooperation enforcement model a full information framework is available to the nodes.

In the one-shot PD game of many players, if the marginal per-capita return for a node is m, then the optimal decision rule for a node of ERC preference a/b is:

$$\frac{a}{b} < \frac{p - \frac{1}{2}}{4(1-m)(1+2m)^2}$$

p is the probability that the player cooperates. Cooperation is influenced by the ERC preference of the player, the magnitude of m, and the proportion of cooperating players in the game. ERC dictates that there is always some equilibrium where no players cooperate, but there may also be equilibria where a proportion of subjects cooperate, while other defect. In this way it is possible to show that a sufficiently dense ad-hoc network of nodes with different ERC preferences can succeed, because there will be a coalition of cooperating nodes.

Analysing a the sequential PD game is interesting. In a simultaneous game the choice of the opposition is unknown until the player makes its choice. But in the sequential game the choice of the first mover is to defect or cooperate and the opponent knows this choice before playing. A first mover can ensure equal payoffs between the players by defecting. Then the second mover will defect for sure. However, if the first mover is sufficiently interested by the absolute payoff, the first mover may cooperate take the chance of being exploited. Hopefully this entices the second mover to cooperate too. This altruistic behaviour is important to ensure that networks can converge to an equilibrium of at least some cooperating nodes.

Appendix D

Ad-hoc Routing Protocols

Two ad-hoc routing protocols were reviewed in Chapter 4. For interest, three other ad-hoc routing protocols are summarised below, and the choice between using on-demand and table-driven protocols is discussed.

D.1 Table-Driven

The *Global State Routing* (GSR) protocol [51] is similar to DSDV but reduces network congestion by avoiding flooding routing messages. Every node in a GSR network maintains a neighbour list, a topology table, a next-hop table, and a distance table. The nodes that are in radio range are included in the neighbour list. All the destinations nodes in the topology table have associated link state information as reported by the destination, and a timestamp that indicates how fresh the information is. Each destination in the table also has a next-hop entry where packets should be sent to reach it. The shortest distance to each node can be found in the distance table. Upon receiving a routing message, a node will update its topology table if the sequence number of the routing message is newer than the one stored in the table. The node then broadcasts this information after altering its own tables.

Another table-driven routing protocol is the *Zone-Based Hierarchical Link State Routing Protocol* (ZHLS) [52]. In this protocol the network is divided into non-overlapping zones. Two levels of topology are defined in ZHLS: node level and zone level. The node-level topology describes how nodes are physically connected. If a node is physically connected

to a node of another zone then a virtual link between the zones exists. Two types of link state packets (LSP) exist: node LSP and zone LSP. The node LSP of a device contains its neighbour information and is only propagated within its zone. Zone LSPs contain zone information and are propagated globally. In this way every node only has topology information about the nodes in its particular zone, and has zone connectivity information about all the zones in the network. To route a packet a node requires the zone ID and node ID of the corresponding node. Thus the packet is routed based on the zone ID until it reaches its respective zone and then it is routed using the node ID.

D.2 On-Demand

Ad-hoc On-demand Distance Vector Routing (AODV) [53] is an on-demand version of the DSDV protocol. AODV determines path routes to destination node by sending route requests. These packets are flooded throughout the network until they reach an intermediate node that has recent route information about the destination, or until they reaches the destination itself. To avoid loops and inefficient routes, nodes discard route requests that they have already seen. Sequence numbers in the route request ensure that only fresh routes are propagated back to the sender. On forwarding a route request packet, a node will store the node from which the first copy of the request came. This information can be used to formulate a reverse path for the route reply packet. As the route reply makes it way back to the sender, the intermediate nodes enter the forward route information into their routing tables. If a source node moves, changing the network topology, then it will re-initiate a route discovery. If intermediate nodes move then its neighbours will inform the upstream nodes until the source node receives the message and initiates a new route discovery.

D.3 Routing Protocol Discussion

There is a great deal of literature discussing which ad-hoc wireless routing protocols are optimal. It is clear that in highly mobile networks on-demand routing protocols place less load on the network than table-driven routing protocols. This is because table-driven schemes require that the nodes have up-to-date information all the time and frequent topology changes will cause the network to be overloaded with all this routing informa-

tion. On-demand schemes only require a node's routing table to be fresh when there is information to be sent. Thus they are suitable for networks that are highly mobile, or for nodes that do not send packets regularly. However, if there are delay constraints on the message to be sent, then the extra overhead of route discovery may make on-demand routing unfeasible. In the case of networks that have quality of service requirements, a table-driven protocol should be used to ensure the routes are ready for use when a message is to be sent.