A Self-Organising Distributed Location Server for Ad Hoc Networks

A comprehensive analysis of using self-organising agents for storing location information in ad hoc networks.

By

Gareth Owen

The thesis is submitted in partial fulfilment of the requirements for the award of the degree of **Doctor of Philosophy** of the University of Portsmouth

School of Computing
University of Portsmouth
Lion Terrace, Portsmouth, Hampshire
PO1 3HE, United Kingdom

Date: Sept 2007

Abstract

Wireless networks allow communication between multiple devices (nodes) without the use of wires. Range in such networks is often limited restricting the use of networks to small offices and homes; however, it is possible to use nodes to forward packets for others thereby extending the communication range of individual nodes. Networks employing such forwarding are called Multi-Hop Ad Hoc Networks (MANETS)

Discovering routes in MANETS is a challenging task given that the topology is flat and node addresses reveal nothing about their place in the network. In addition, nodes may move or leave changing the network topology quickly. Existing approaches to discovering locations involve either broadcast dissemination or broadcast route discovery throughout the entire network. The reliance on the use of techniques that use broadcast schemes restricts the size of network that the techniques are applicable to.

Routing in large scale ad hoc networks is therefore achieved by the use of geographical forwarding. Each node is required to know its location and that of its neighbours so that it may use this information for forward packets. The next hop chosen is the neighbour that is closest to the destination and a number of techniques are used to handle scenarios where the network has areas void of nodes.

Use of such geographical routing techniques requires knowledge of the destination's location. This is provided by location servers and the literature proposes a number of methods of providing them. Unfortunately many of the schemes are limited by using a proportion of the network that increases with size, thereby immediately limiting the scalability. Only one technique is surveyed that provides high scalability but it has a number of limitations in terms of handling node mobility and failure. Ad hoc networks have limited capacity and so the inspiration for a technique to address these shortcomings comes from observations of nature.

Birds and ants are able to organise themselves without direct communication through the observation of their environment and their peers. They provide an emergent intelligence based on individual actions rather than group collaboration. This thesis attempts to discover whether software agents can mimic this by creating a group of agents to store location information in a specific location. Instead of requiring central co-ordination, the agents observe one another and make individual decisions to create an emergent intelligence that causes them to resist mobility and node failures.

The new technique is called a Self Organising Location Server (SOLS) and is compared against existing approaches to location servers. Most existing techniques do not scale well whereas SOLS uses a new idea of a home location. The use of this idea and the self organising behaviour of the agents that store the information results in significant benefits in performance. SOLS significantly out performs Terminode home region, the only other scalable approach surveyed. SOLS is able to tolerate much higher node failure rates than expected in likely implementations of large scale ad hoc networks. In addition, SOLS successfully mitigates node mobility which is likely to be encountered in an ad hoc network.

Acknowledgements

A number of people have contributed indirectly to this thesis in their support, discussions, ideas and friendship, many more than I could hope to list. First and foremost I would like to thank my family for their continued support. Despite their apprehensions about my eternal life as a student, they have never wavered in their support. In everything I do they have supported me and it is without doubt I say that if it were not for this, I would not be where I am today.

Secondly, my thanks go to colleagues who have been instrumental in the completion of this thesis. My supervisor, Mo, for his enthusiasm and guidance throughout my PhD; He has always been positive and supportive, even when things were not proceeding as planned. I also wish to thank fellow members of the department who have looked after me by giving guidance on the academic minefield; namely Amanda Peart for her never faltering willingness to help me understand the intricacies of university procedures, teaching and for supporting me through my PgC in LTHE. Thank you to Penny Hart for giving me the opportunity to work in the tutor centre and meet so many other great people. Also, Jane Chandler, my first head of department, and Bart-Floris Visscher, for their open arms welcome to the department which made me feel at home very quickly.

Thirdly, I thank my partner Sam and my friends, all of whom have brought enjoyment to my time away from my PhD and without them I would not have enjoyed the last three years as much as I have. To Gary, Natalie, Maddie, Tanya, Tom, Alex, and all those whom I have missed out, thank you for your friendship.

Finally, I wish to thank my fellow PhD students Mouhammed, Athanasios, Vikas, Antoniya, Helen and Penny Ross for always being willing to listen and suggest improvements, and for bringing enjoyment to the office.

Declaration

Whilst registered as a candidate for the above degree, I have not been registered for any other research award. The results and conclusions embodied in this thesis are the work of the named candidate and have not been submitted for any other academic award.

SIGNED: _____

Table of Contents

List of figures

List of tables

Glossary of Terms

Table 1: Glossary of terms

Term	Description
AODV	Ad-hoc On-demand Distance Vector – an on-demand ad-hoc network routing protocol.
Beacon	A packet sent periodically advertising information such as that node's location.
Delay	The time taken for a packet to travel from one node to another.
DSR	Dynamic Source Routing – an on-demain ad hoc network routing protocol.
Edge/face	
Geodesic/geographical routing	Routing a packet using geographical information about the destination.
IETF	Internet Engineering Task Force – standards setting body for Internet protocols.
IP(v4/6)	Internet Protocol version 4 or 6. Version 4 is in current use for the Internet with 6 being currently rolled out. IPv6 provides a number of enhancements such as a larger address space and security.
Location server	A server responsible for storing the location of a node or set of nodes so that the information can be accessed to enable geodesic routing.

MAC Layer	The second level of the OSI Seven Layer network model whereby data is placed into medium specific packets. The layer also concerns medium specific protocols.
MFR	Most-Forward-within-Radius – a means of forwarding packets in a network using location information.
Neighbour	If node A is within wireless communication range of node B, then node A is a neighbour of B and vice versus.
Node	A wireless device such as a mobile phone, PDA or laptop.
Node degree	The number of neighbours a node has.
OLSR	Optimised Link State Routing – A proactive ad-hoc network routing protocol.
Overhead	Traffic that is used to maintain the function of the network; e.g. it is not application data.
Physical Layer	The lowest part of the OSI Seven Layer network model; e.g. the physical side of wireless communication; such as, modulation, interference, attenuation, etc.
Quorum	A replicated state machine; e.g. a set of servers responsible for storing data in the face of server failures.
RFC	Request for Comments – A protocol proposal/standard submitted to the IETF.
Routing	The process for deciding on which path messages will take through the network.
Stigmergy	The process of a group on entities co-operate by

	communicating through modification of their environment.
Void	An area with a lack of nodes.

1 Introduction

1.1 Background

The founder of Microsoft, William H. Gates, had a vision of there being a computer on every desk in every home (Gates, 1995). This vision has come to fruition recently, with most homes in Britain owning a personal computer (ONS, 2002). What he perhaps did not predict would be the dependability on computers for modern life and the subsequent desire for them to be carried around. Computers are now carried in the form of mobile telephones by a significant proportion of the population and their functions are exceeding far beyond their initial design. For example, many mobile telephones now include as standard calendars, games, and recently satellite navigation.

Traditionally computers communicated by using wired networks, requiring each PC to be connected by a wire. Recently, wireless networks have become a viable alternative with speeds up to 54Mbps already available and technology enabling speeds for 480Mbps being standardised by the IEEE 802.11 Working Group. Along with recent decline in laptop prices, wireless networks have become popular for use of the Internet anywhere in the home. This set up involves purchasing a wireless access point and connecting it to your Internet connection point, allowing you to access the internet wirelessly whilst within range. This range has been cited to be up to 300m but within a built up area 100m is a more practical expectation (Dynalink, 2004). If one has a particularly large house and wishes to have wireless access beyond this range, then they may install repeater access points at strategically placed points around the home. These access points, like the original, must then be manually configured. The task of connecting to the wireless network is much simpler with many client devices connecting and obtaining an IP address automatically (through DHCP (IETF, 1997)) when within range.

While this works well for the technically competent home user, those that are not will be unable to configure multiple access points and so will stay within range. More interestingly, let us consider a lecture theatre where students have brought their mobile

devices. If the lecturer wishes to share files with the students, or somehow interact with their devices, he would have to have previously set up an access point and all students would have to be within range. Equally so, at a large conference, if the organisers were to make files available wirelessly, all users would have to be within range of a previously configured wireless network. While this is not unreasonable and there are many companies that are now successful offering these services, it is costly and not strictly needed.

Consider the situation of a typical park at lunchtime, with a number of people using mobile devices whilst eating their lunch. Two people, person A and person B wish to communicate with one another, but they are at opposite sides of the park. If the distance between them is greater than 100m then they are unable to communicate at present. Figure 1 illustrates this scenario with node A having a transmission range shown by the dotted circle, and B being outside of this range. A has two neighbours, C and D, which are defined as those nodes that are within its transmission range.

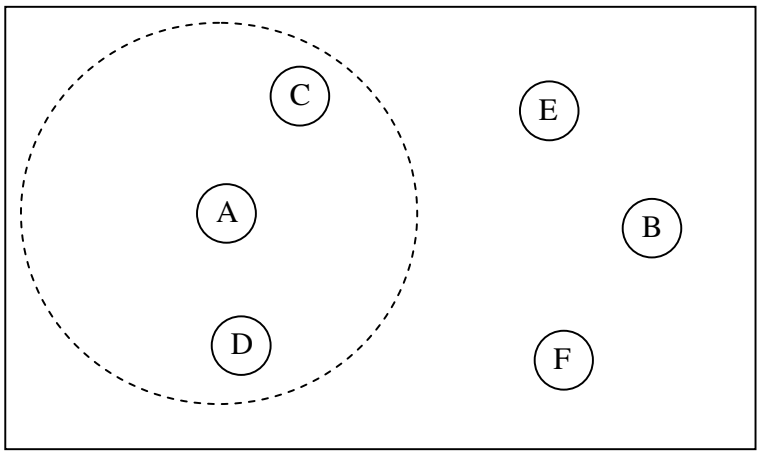

Figure 1: A simple ad hoc network

The solution to the problem is two fold: that of configuration, and of transmission range. In terms of configuration the devices need to agree how to address one another, such as in TCP/IP, a popular networking technology, each device is assigned an IP address when connecting to an access point based network. In this scenario however, there is no central authority to assign IP addresses and so the nodes need to agree so that they have distinct

IP addresses. When all nodes are in range simply choosing an address and asking if any other node is using it, and repeating this until an unused address is found would suffice. In larger networks where not all nodes are in range, researchers have proposed multi-hop agreement protocols (Thoppian and Prakash, 2006). Perhaps a more effective means of approaching this is that every wireless network device is assigned at manufacture a unique 48-bit Media Access Control (MAC) address, and this could be used to derive the IP address. With the current IP version, the maximum address size is 32 bits and so this scheme would be mostly ineffective due to the chance of two nodes choosing the same address; however, the next version currently undergoing deployment provides 128 bits which would allow a fraction of this address space to be set aside for wireless devices, using the MAC address to determine the low-order bytes.

If we assume that all devices in the park are agreeing on the same configuration through some means, but that they are not within range of all other nodes, the next task is communication beyond the transmission range. One could increase the transmission power as GSM mobile phones do when they become distant from the base tower, but this has two problems. Firstly is the issue of output power, the band in which current WiFi technology operates has strict legal requirements on Radio Frequency (RF) power output. Secondly is the issue of battery conservation, as discussed later nodes may more frequently participate in communication than mobile phones due to their routing function and so the higher RF output powers will reduce battery life.

If person A wishes to communicate with B but he is out of range then he is currently unable to given restrictions limiting the power output, however, he could ask a node that is within range to repeat or forward the message. This forwarding would then happen at every device along the path to person B's device. Figure 2 illustrates this scenario with node A sending a message through two intermediate nodes before reaching B (the dotted circles indication transmission range).

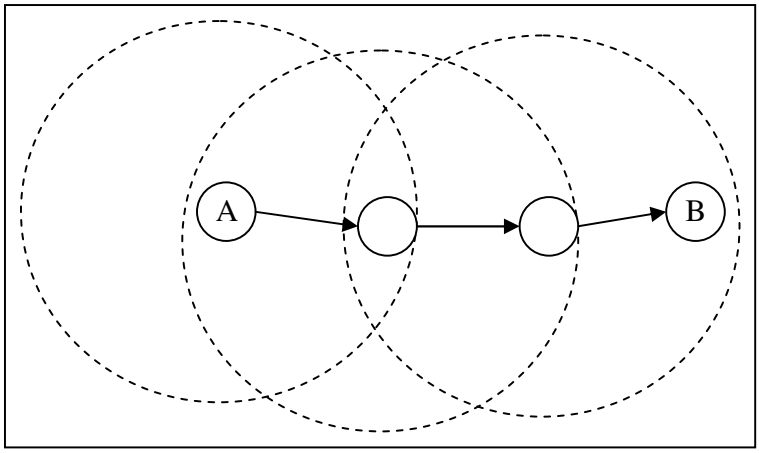

Figure 2: Illustration of multi-hop forwarding in ad hoc networks

Given the example in Figure 2 the question remains of how one finds the path between A and B so that nodes know which node to forward the message to at each hop. This issue is covered in more detail in the literature review but is briefly explained here. It is worth noting that a node can only see that its neighbours and no further, and so cannot see where B is or which node to send packets through. A simple way around this is to ask all of its neighbours if they know where B is, if they do not then they also ask their neighbours, and so on until eventually a node is reached which is a neighbour to B. At each stage a record of the path has been stored in the search request and so upon reaching B it is possible to construct a reply and send it back down the path with the correct hops to go through. This is similar to one of the most popular routing algorithms for ad hoc networks, Ad-hoc On-demand Distance Vector (AODV) (Perkins and Royer, 1999).

While the AODV style technique works very well for small networks with low mobility, when networks become large and/or mobility increases, the overhead of finding the path becomes prohibitively expensive. An alternative method to combat this is to require each node to know its location, that of its neighbours and the destination. Given this knowledge the message can simply be sent to the neighbour which is closest to the destination. This technique is currently the only technique which is highly scalable but leaves the question of how to find the location of the destination.

The focus of this work is on how to store location information for a particular node in an ad hoc network in a fashion that makes the approach scalable to an extremely large number of nodes. A simple technique would be to require every node to periodically broadcast its location to all other nodes (Basagni et al., 1998) but with increasing numbers of nodes the overhead becomes prohibitively expensive. The technique proposed in this thesis does not require dissemination amongst all nodes or an increasing number of nodes as the network size increases, and so is therefore is highly scalable.

1.2 Inspiration

Nature is incredibly complex and humanity still does not fully understand it. Humans have evolved from mere proteins over millions of years and are now beginning to understand the process which has brought about our being. If one looks at some of the functions that have evolved to make human and animal life possible, they are often incredibly intuitive and reliant on a precisely tuned set of events. Academics in artificial intelligence quickly realised that nature perhaps held the key to developing intelligent computers with the discovery of neural networks and genetic algorithms. Neural networks were designed to mimic the way neurons in the brain interact and process information. Genetic algorithms were designed to mimic the way life has evolved, taking a set of solutions and at each iteration choosing the best performing according to some metric and combining them for a new generation. At progressive iterations more effective solutions to the problem evolve.

An interesting phenomenon that we can observe in nature is that of self-organisation (Morowitz, 2003), the process by which selfish individuals co-operate or interact to achieve a global goal. Take for example humans in society with selfish goals of power, wealth and respect that drive them to interact with others and realise these goals. Although we are all individual, with our own desires, we have to co-operate to realise them to their fullest extent. This is observed everywhere in the animal kingdom, from birds flocking (Carlson, 2000, Reynolds, 1987) and ants building nests (Johnson, 2001) to human society, all are individual beings that have selfish goals but who co-operate to realise them. To further extend this let us examine how they co-operate. One might

initially assume that this is through direct communication through voice or a similar mechanism which is so obvious in human society but is only part of the story. Consider a visit to the cinema, you see a handful of seats with jackets placed on them, you automatically assume someone is sat there and find another seat. They have communicated with you indirectly which has caused a change in your behaviour. Another example is if you see a queue for a counter at a supermarket, you automatically walk to the back of the queue rather than to the counter, forming an ordered system to get served and maximise efficient service for society. You have not spoken with the people in the queue but you have observed their positions and altered your actions accordingly.

1.2.1 Ant-colonies

In an ant-colony, each ant acts independently seeking food when leaving the nest. Upon finding some food, the ant returns to the nest laying down pheromone molecules along the route. Other ants sensing this pheromone follow it to the food, and they too lay down reinforcing pheromones on their return. The shortest path to the food is traversed more frequently because it is shorter and so becomes more reinforced than the longer paths. Although the ants have acted independently, only changing their environment by laying down pheromones, they have achieved a global goal of finding the shortest path to food. This is an emergent property of their individual and independent actions (Johnson, 2001).

Ants communicate indirectly by modifying their environment by laying pheromones. Other ants then sense these pheromones and this information is used to permit indirect communication. This process of modifying one's environment to permit indirect communication is called Stigmergy. In a human environment, if one leaves a coat on a seat in a public area such as a cinema or bar, others assume this seat is taken and alter their behaviour by choosing to sit elsewhere (Tummolini and Castelfranchi, 2007).

As expected, most applications of ant colony optimisation have focused upon finding paths in networks, from the travelling salesman problem (Puris et al., 2007, Pop et al., 2007) to routing in communication networks (Zheng et al., 2007).

Ants were proposed as a solution to finding shortest paths in ad hoc networks due to their efficiency at finding them in nature. Simulation of such solutions have found them able to be highly adaptive, provide multi-path routing and data load spreading (Ducatelle et al., 2006, Di Caro et al., 2004).

Algorithms such as Antnet (Dhillon and Van Mieghem, 2007, Tekiner, 2004, Di Caro and Dorigo, 1999) and ABC (Rajagopalan and Shen, 2006) sent out virtual ants at regular intervals to randomly chosen destinations. The ants sample paths and assign quality whilst updating the routing tables as they pass. They assign a goodness value to each path based upon a virtual pheromone. For ant-based systems that work on this principle to be useful in higher mobility scenarios there must be enough ants moving across the network to disseminate and discover routes; however, this incurs a significant overhead but attempts to limit repeated path sampling results in the ants losing much of the explorative behaviour (Gunes et al., 2002).

1.2.2 Particle swarms

When birds flock, they observe their neighbours' direction and velocity. Using this information, the bird can make a decision on its direction and speed. Typically, the bird will introduce a certain amount of randomness in its decision making so that the flock as a whole can search for food and not move in just a straight line. The birds can change direction extremely quickly without any apparent central co-ordination; this is because there is no central co-ordination and their behaviour is emergent and self-organising. For example, if one observes a predator approach a school of fish, the school will change direction very quickly, in a wave like fashion from the closest point to the predator. Much like in human crowds, if there is a danger, a small number of people start running, others see this and then instinctively run too (Carlson, 2000, Reynolds, 1987).

The behaviour observed in flocks is referred to as an example of Particle swarm optimisation (PSO) (Eberhart et al., 2002) and differs from Ant-Colony optimisation in

that the entities do not necessarily modify their environment but rely more on observation of their peers. Research interest into PSO's applications has been largely applied to neural network optimisation (Yusiong and Naval, 2006) and also limitedly to routing in small ad hoc networks (Rajagopalan and Shen, 2006).

Particle swarm optimisation has been applied to ad hoc networks (Zhang and Xu, 2006, Yuan et al., 2006, Rajagopalan and Shen, 2006) in a number of techniques for disseminating routing information throughout the network.

1.2.3 Application of inspiration

It is indirect communication observed in nature as described in the last section that is the inspiration for this thesis. Communication capacity in a network is limited in wireless networks and much of the research literature focuses on ways to reduce overhead. If it is possible to incorporate a form of indirect communication into some of the algorithms used, then one can reduce the overhead on the network. If self-organising entities observe each other most of the time instead of directly polling or co-ordinating with one another, then this removes a significant overhead burden. In particular, we want a number of entities to co-operate with one another in a self-organising approach through indirect communication. Specifically the thesis will focus on trying to use this idea of indirect communication in self organisation to store location information in a network whilst incurring the minimum overhead.

1.3 Motivation

Large scale ad hoc networks have enormous potential in their uses and the list is never ending. Envisage being able to always be in range of a wireless hotspot when you move into a new home or visit a conference. Or consider cellular network providers not needing to invest on as much infrastructure and passing the savings onto its customers. Perhaps even more encouraging is doing away with cellular networks altogether, and eliminating the need for us to pay for a phone calls and Internet access. While not particularly favourable for the cellular companies, they can augment the network by

providing extra services or faster Internet access by moving their target market and still profit from this development; although it would be up to the user whether to utilise the services or not.

Consider disaster areas such as New Orleans where the infrastructure has been destroyed. The emergency services would have initially benefited from an ad hoc network for communication but now that this stage of the disaster is over, large scale ad hoc networks could play a pivotal role in restoring communication to the region. Equally so, in third world countries and villages where communications investment is not profitable, ad hoc networks can be beneficial in providing communication for free.

The main motivation for this work is to bring us one step closer to a large ad hoc network. This work solves one hurdle, the provision of location services, which although only one step in the path, it is a significant step. Currently much of the focus in the area is on small ad hoc networks although increasingly large scale ad hoc networks are being researched. With the addition of this work, only a few problems remain before we can all deploy a truly free network, such as capacity limitations.

1.4 Contributions

This thesis makes a number of important contributions to the use of geographical routing and in particular location servers. As the thesis is examining storing location information, first to be examined will be how beaconing and location age affects the forwarding of packets through geographical routing. If beaconing can be reduced and the distance between location updates can be maximised, then the overhead can be reduced further.

The main contribution of this thesis is in a self organising location server that uses mostly indirect communication to reduce overhead. For any system to scale in wireless network the overhead on the network must be independent of the number of nodes in the network. Even if the overhead is dependant upon the number of nodes, then with sufficiently large

number of nodes the capacity available will approach zero (Jain et al., 2001). Examining the existing location servers that match this criterion finds that they require the use of a region of the network to store information whether there are nodes present. When nodes are not present the system fails and therefore a paradigm shift is proposed to geographical points rather than regions, around which the data should be stored.

The next contribution is an examination of the possibility of storing data near a geographical point in mobile network and the parameters that affect the proximity to this point of a data hosting entity that is able to make migration decisions. This entity makes its own decisions on when to move from node to node (a migration) to try and remain as close to this point as possible. The thesis examines what factors affect its performance in this endeavour. It needs to move from node to node if the network is mobile otherwise it may become quite distant from the point.

Issues of fault tolerance are addressed and a number of techniques for producing a self-organising group of entities are examined. These entities all attempt to remain as close to the geographic point as possible but whilst ensuring that there are enough entities to survive failures of nodes. The entities only communicate through modifying their environment yet form a global intelligence of co-operation despite their independent actions.

Once these entities are able to tolerate mobility and failures of nodes whilst acting independently but communicating through observation of one another, the use of them for storing data in an ad-hoc network is examined. Techniques to query the information in a similar non-centralised manner are examined along with possible improvements.

The thesis also examines the performance of this data storage technique in storing location information for the use in geographical routing. The technique is compared with similar approaches to location servers and is critically analysed.

To enable analysis of such an algorithm several simulations were developed. A complete routing layer module was created for two ad hoc simulators. In addition, a prototyping simulator was developed to allow rapid testing of ideas allowing many ideas to be examined without costly development time.

1.5 Thesis outline

Chapter 2 will discuss traditional methods of routing in ad hoc networks and their pitfalls. A number of algorithms that provide routing for smaller scale networks will be examined and the problem of using them in larger networks explained. The chapter then explains routing for large scale ad hoc networks and the use of geographical routing techniques.

Chapter 3 elaborates on the current literature of large ad hoc networks and examined methods of discovering the location of another node. Geographical routing requires this location information and a number of techniques for providing this were examined in detail.

Chapter 4 examines the feasibility of the idea of a group of autonomous agents remaining near a geographical point. Methods for theses agents to replicate and co-ordinate whilst minimising communication are analysed for later use in storage of information. The technique is termed SOLS and is based upon the ideas of ant colony and particle swarm optimisation described earlier. SOLS incurred low overhead and provided high tolerance to node failures.

The use of the SOLS technique developed in Chapter 4 is used for storage of location information in Chapter 5. Initially, the task of updating and querying the server is examined. A number of closely related topics were also examined such as location age and its effect on routing success, and the performance of various methods for choosing when to update. SOLS was shown to be the only technique providing high failure tolerance and high scalability, and performed better in all realistic scenarios against the most similar approach, Terminode home region.

Finally, given a robust and successful location server using SOLS, Chapter 6 examines the implementation on computing devices. The chapter discusses how SOLS and associated routing is implemented in both the application and network layers of the OSI model. Chapter 7 concludes the thesis and discusses future work.

2 Routing in Ad-hoc Networks

2.1 Introduction

Routing in networks can be thought of as finding the path between two computers in a network; much like one finds a route across a road system. In wired computer networks such as the Internet this task is made much easier by IP addresses. Typically, a group of IP addresses will be allocated to a country, and then a subset of that to a region within the country, and so on. This is similar to how a telephone system works with area codes and country codes, but in IP networks the geographical segregation is not as distinct as there could be several IP ranges for one area depending of who the supplier was. The network can therefore be arranged in a tree-like fashion, with the individual computers at the bottom, and increasingly large address ranges as you move up the tree, with the whole address space being the head. For routers to be able to automatically discover this information, a HELLO packet is periodically broadcast between neighbouring routers containing information on which networks it can see. This information is recorded by receiving routers and sometimes shared with their neighbours.

Routing in ad hoc networks is a different matter altogether as there is no hierarchy to reduce complexity. Although each computer may have an IP address, the computer could then move to the other side of the network therefore negating any regional indicator the address provided. A node can only see computers in its transmission range and every computer must act as a router relaying packets for every other node. The routing task is flat in comparison to the Internet's hierarchy and this requires a different approach.

In an ad hoc network, each node needs to be able to discover a path of nodes between itself and the node with which it wishes to communicate. Traditionally in ad hoc research, this has been achieved through two methods: proactive and reactive. The proactive approach uses a similar technique to the HELLO messages mentioned earlier and information is slow to propagate through the network. The reactive approach uses a flooding style broadcast to discover a route to the destination on-demand.

Unfortunately, neither of these approaches are scalable to large scale networks and so an alternative method called Geographical Routing was discovered. This chapter will outline the traditional methods, the scalability problem along with geographical routing and the need for location servers if large scale ad hoc networks are going to be possible.

2.2 Traditional Routing

Traditional methods of routing within an ad-hoc network fall into several categories. The first of these are 'reactive' and initiate discovery as and when required. Whilst a proactive protocol will actively seek to have knowledge of routes to all possible destinations before the information is needed (Johansson et al., 1999, Tseng et al., 2002a).

A reactive protocol performs route discovery only when packets are waiting to be sent to a node where the route is not known. Upon a node generating a packet to such a node, the routing layer will queue it and initiate the route discovery process. In most protocols this consists on an n-hop broadcast to find the destination, with a reply being sent back once the node is found. Many of the protocols also have a route maintenance phase to maintain routes in the presence of link failures.

A proactive protocol performs route discovery all the time, attempting to maintain a table of possible destinations and the routes. Nodes share routing information amongst their neighbours either periodically or in response to topology change and this information is disseminated to all nodes.

Choosing a protocol to use depends very much on the application to which it is being applied. Reactive protocols cause a connection initiation delay due to the discovery process taking place at that time; however, the route discovered is more likely to be up to date. Proactive protocols often respond slowly to changes in topology as the information converges slowly by nodes sharing information. In addition, proactive protocols incur an overhead regardless of whether any communication is taking place in that part of the network.

Generally a routing protocol needs to address three key issues to provide a service in a communications networks:

- Route Discovery – This process involves the discovery of the path through the network to the destination. The routing protocol has to establish which nodes packets need to be routed through to reach the destination and this path will usually fit certain criteria. These criteria can range from the most common being the least number of hops to quality of service parameters.
- Route Maintenance – In a wired network, a route can become invalid at any point due to node failures or changes in network topology; however, in an ad hoc network, node mobility is the most common cause of a route becoming invalid due to the change in topology. Therefore, a routing protocol is expected to handle this scenario and rediscovery or alter the route as quickly as possible to maintain the communication path.
- Packet Relay – Once a path has been discovered between the source and destination, the routing protocol is responsible for passing packets along it. This includes handling dropped packets and sensing link failures so that route maintenance may be initiated. In addition, this part of the protocol is also responsible for initiating route discovery when a new packet without a route is received or created.

In this sub-chapter, two routing protocols will be described, the reactive Ad-hoc on Demand Distance Vector (AODV) protocol, and the proactive Optimized Link State Routing (OLSR) protocol. The advantages of each are laid out and then the reasons they are not suitable for large scale networks.

2.2.1 Reactive – AODV

Ad-hoc on-demand distance vector (Perkins and Royer, 1999) has currently been accepted as an experimental request for comments (RFC) by the Internet Engineering Task Force (IETF) Mobile Ad-hoc Networks Working Group (MANET-WG). A number

of other protocols have also been accepted by the group but a number of studies have shown AODV to excel especially in stressful situations of high mobility and traffic loads (Perkins et al., 2001).

AODV builds routes using a route-request (RREQ) and route-reply (RREP) mechanism. A node wishing to discover a route builds a RREQ packet and broadcasts it to all its neighbours. Every node receiving the packet then also rebroadcasts it after recording some information. Each node on receiving the packet, records a backward pointer to the source via the previous hop along with a number of other details about the request. On reaching the destination node, a RREP is generated and sent back to the source using the backward pointers. At each hop the RREP is sent by unicast and each hop records a forward pointer to the destination via the previous hop of the RREP. As soon as the RREP is received at the source node it may begin sending data having established backward and forward pointers on all nodes along the route.

Table 2: Example AODV forward and backward entry

Source node ID	Destination node ID	Backward node ID	Forward nodeID

The route is considered active as long as packets are being periodically sent along the route. If no packets are sent, then each node along the route will timeout the information stored. If a link fails, perhaps due to node mobility, a route-error (RERR) packet is sent by unicast back to the source so that it may reinitiate route discovery.

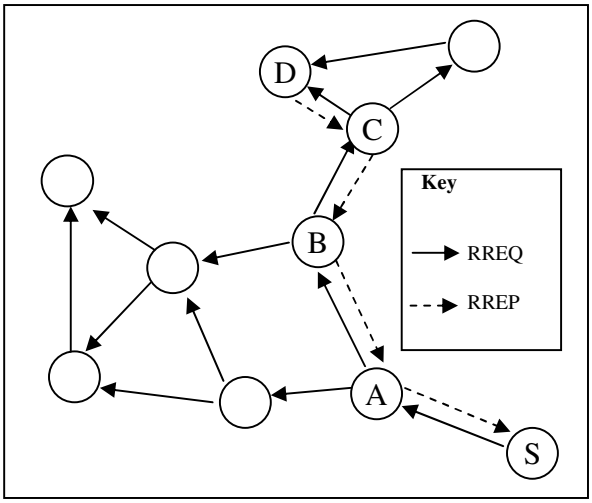

Figure 3: Illustration of AODV route discovery

Figure 3 gives a simple diagram of how route discovery takes place. S broadcasts a RREQ to its neighbours, who then also rebroadcast and so on. When the RREQ finally reaches D, it unicasts a RREP back to the source. On receiving the RREP at S, a route is established and communication can take place. Should any nodes in the route move from its position and compromise the path, then route maintenance will be initiated to correct the path by substitution of other nodes. In the example above, the AODV pointer table entries at each node would appear as:

Table 3: Example AODV pointer entries

At node	Source	Destination	Backward	Forward
S	S	D	-	A
A	S	D	S	B
B	S	D	A	C
C	S	D	B	D
D	S	D	C	-

During the RREQ phase, only the backward pointers in the above table will be filled. When the RREP phase starts then those nodes which are on the route will fill their forward pointers. It is worth noting that all nodes in the network will have a backward pointer to the source, even if they are not on the path because the RREQ is sent to all

nodes. Nodes that are not on the path can cache this information as an optimisation for later use.

2.2.2 Pro-active - OLSR

The Optimized Link State Routing (OLSR) protocol (Clausen and Jacquet, 2003) has also been accepted as an experimental RFC by the IETF MANET-WG. OLSR is an optimisation for wireless networks of the typical link-state (McQuillan et al., 1980) algorithms used in wired networks. Link-state routing protocols require a node to flood its neighbour information to all other nodes. Each node then uses the information it has received about all other nodes to build routes. Flooding can be costly in ad hoc networks and so OLSR is an optimisation of the flooding used in link-state algorithms.

Consider the example in Figure 4 where A broadcasts its update which is received by B, D and E, who then also rebroadcast the update which is received not only by each other but also by C who also rebroadcasts the packet. The cost of this is 5 broadcasts and can be greatly optimised to just two. A could broadcast the packet, then B, having the most neighbours, could rebroadcast it. All nodes have now received the update at a saving of more than half.

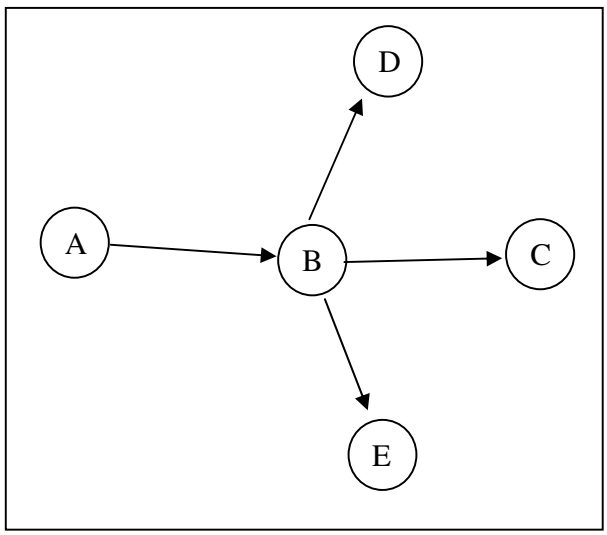

Figure 4: Illustration of redundancy in broadcasting

OLSR attempts to minimise the flooding overhead by using a technique similar to this. Each node chooses a set of neighbours such that, through them, it is connected to all nodes two hops away. The neighbours in this set are called Multi-Point-Relays (MPRs). If only a neighbour's MPRs rebroadcasts its update, then the overhead can be reduced whilst still reaching all nodes. Each node broadcasts a HELLO packet listing its MPRs so that another node knows if it is an MPR for another node, and should rebroadcast its packets. Consider the example in Figure 5, where node A has selected four MPRs that connect it to all nodes that are two hops away. Now an update flood throughout this network from node A will reduce the overhead to 5 broadcasts, the initial from node A, and then one from each of its MPRs.

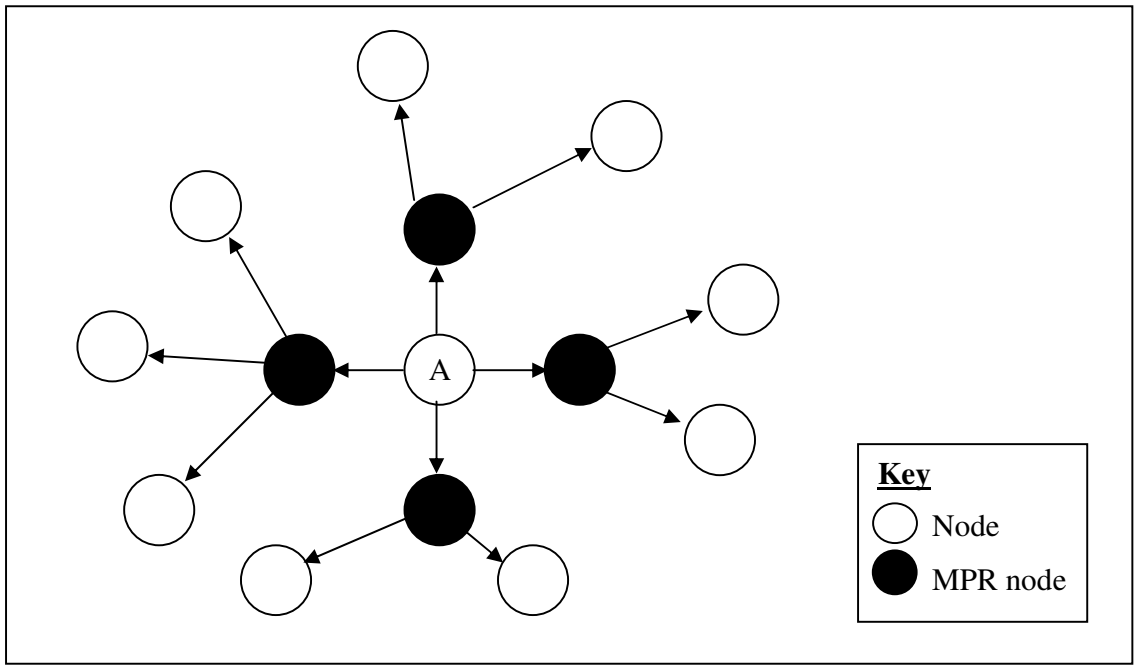

Figure 5: Illustration of MPR selection in OLSR

The difficulty with this approach is that there will be flooding overhead regardless of whether nodes are exchanging packets. The flooding is still costly and if communication is minimal or between a small subset of nodes, then much of the flooding is unnecessary.

On-demand protocols have been found to be more effective than proactive protocols in typical scenarios (Royer and Toh, 1999, Johnson, 2001, Perkins and Royer, 1999). The

main reasoning for this is that time between updates can be infrequent and in distance-vector schemes dissemination can be time consuming. As a result changes in the network take a long time to propagate and in a dynamic network this is not acceptable. Ad hoc networks are generally considered to be highly dynamic with links changing frequently and so applying a similar idea to that which works in the Internet is not suitable.

2.2.3 The scalability problem

AODV and OLSR both perform adequately for a number of scenarios used by small ad-hoc networks; however, because they require communication amongst a large set of nodes the traffic incurred is therefore related to the number of nodes. Many of the traditional protocols used for ad-hoc networks use a broadcast mechanism for route discovery and/or route maintenance (Lee et al., 2003). Whilst this is manageable for small networks, when one starts examining larger networks then the number of broadcast packets per discovery increases. Even if communication is happening only in confined sections of the network, the entire network will be involved in the routing process. One paper (Tseng et al., 2002b) has found that this 'broadcast storm' can cause the channel to be occupied most of the time with the control messages.

One example of an attempt to adapt AODV to function in larger networks is query localisation (Castaneda et al., 2002). This technique adapts AODV so that when a link breaks, a rediscovery is performed but within a defined region of the network only. This region is defined as a function of the last known distance to the destination, thereby avoiding a network-wide broadcast; however, limiting the discovery range reduces the likelihood of discovery success, especially if the node has moved outside of the search region. In spite of this technique adopting locality searches, it is still required to do network-wide broadcasts initially to find the node. Another method is the expanding ring technique whereby the search area is increased every time the route discovery fails; however, despite reducing the discovery overhead, due a number of initial requests failing, the discovery delay can be much larger (Lee et al., 2003).

On-demand protocols begin to fail with only a few hundred nodes. The reason for this is that path lengths invariably contain many hops, as a result of this the odd route failure in a short path becomes a much greater problem in larger paths due to the much increased likelihood of the entire path failing. In larger networks it is not uncommon for paths to be invalid by the time the route discovery is fully completed, or for the RREP to fail to complete traverse the return path due to link failures (Lee et al., 2003). Figure 6 is a graph from their paper illustrating the effect on delivery success that increases the network size has.

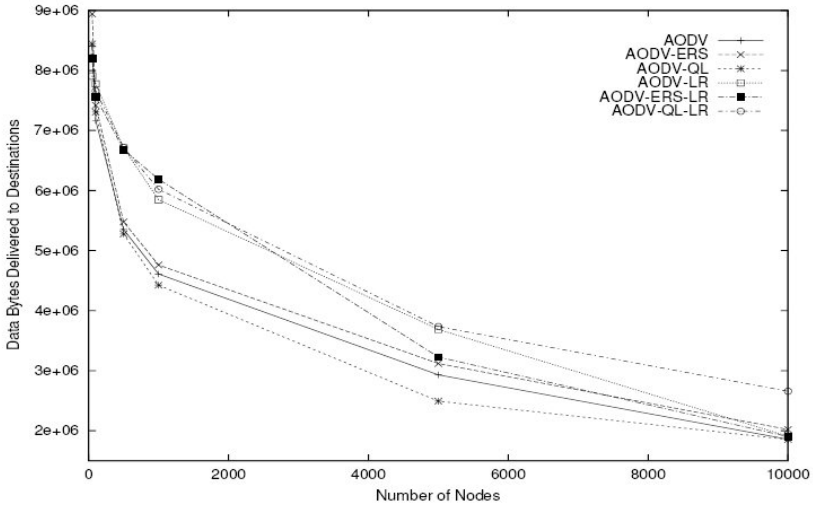

Figure 6: AODV scalability in terms of delivery success (Lee et al., 2003)

In a wired network, such as the Internet, scalability has been addressed by introducing a hierarchy. One can discern from the Internet Protocol (IP) address the paths which the packet needs to traverse, with the higher order numbers representing networks higher up the hierarchy. In a wireless network this simply is not possible due to the flat nature of the network. In spite of this, a number of authors have attempted to implement a hierarchy in such networks, called cluster-based routing protocols.

Cluster based routing protocols are those that group nodes in a locality together into a cluster, and assign one node to be a *cluster head* that is responsible for maintaining the cluster. Cluster heads then maintain routing information for nodes within the cluster and

packets are routed on an inter-cluster basis until the cluster containing the node is found, providing better performance than all-pairs routing protocols (Marshall and Erciyes, 2005, Erciyes and Marshall, 2004). Consider the example in Figure 7 which consists of a number of nodes grouped by locality into clusters, each cluster has elected a node to act as a cluster-head (coloured grey). A node A wishing to send a packet to node B in another cluster sends it to its cluster head, which then forwards it on to other clusters' heads until a cluster is found with the destination.

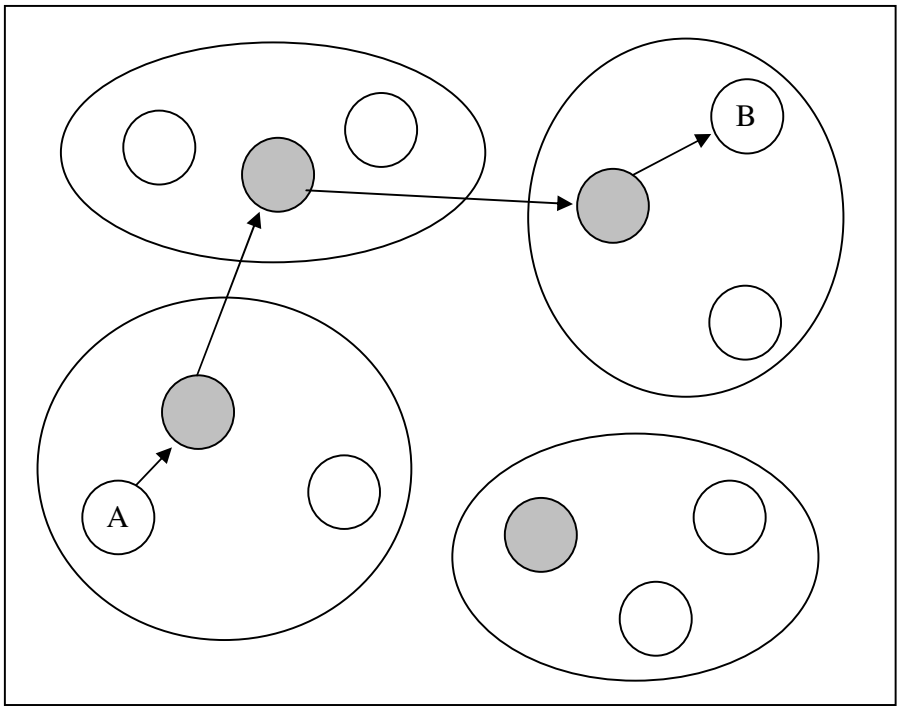

Figure 7: Example of cluster-based routing

Many algorithms choose the node with the lowest ID to act as the cluster-head, but this can result in exhausting of the batteries of such nodes because they take on a more active role. One paper examined a dynamic allocation based upon remaining battery life which showed a reduction in this problem (Gavalas et al., 2006).

2.3 Geographical Routing

It is evident that because of mobility in ad-hoc networks, any paths with high hop counts will fail with high probability. Therefore, one must consider that maintaining a list of nodes that make up a path is not an appropriate means of tackling large-scale ad-hoc networks, which inevitably involve paths with high hop counts. This chapter examines the concept of Geographical Routing, a method of routing packets using knowledge of nodes' locations as opposed to topological information. Individual algorithms are compared and their limitations discussed along with the simple beaconing protocol which is adapted in later chapters.

Before one can utilise geographical routing techniques, there needs to be some means by which nodes can obtain their location, or relative location to others. The simplest way in which this can be done is by fitting Global Positioning System (GPS) receivers into all nodes thereby allowing them to obtain their latitude and longitude to an accuracy of approximately 10m (Parkinson and Spilker, 1996). Recent GPS devices are highly sensitive and not only allow fast position acquisition but use in many indoor environments (MacGougan et al., 2002). GPS receivers are now quite inexpensive and it is feasible to fit them to most mobile devices without a huge increase in cost; in fact, there has been discussion of fitting them into all mobile phones to aid emergency services in finding callers (Zhao, 2002). In the absence of the availability of a GPS, there have been several other papers examining techniques where either none or a small number of nodes have GPS device. One such technique is using triangulation, such as that used in CRICKET (Priyantha et al., 2000, Smith et al., 2004) and the Ad-hoc Positioning System (Niculescu and Nath, 2001, Niculescu and Nath, 2004) to determine the location from nodes who are location-aware, using triangulation. Routing can be achieved without any location information held by nodes in the network by allocating virtual-co-ordinates that do not necessarily need to reflect the underlying geography, but do reflect the connectivity (Jadbabaie, 2004, Rao et al., 2003). All these techniques provide a means to obtain an approximation of one's location but do not necessarily guarantee the accuracy that a GPS device provides, and require increased complexity. In a large-scale network

of constantly changing topology the only means to efficiently provide the location accuracy provided is through use of GPS.

Using one's location to route information has potential privacy implications with anyone being able to obtain another's location. In light of this a number of techniques have been proposed to anonymise location information in such networks (Gedik and Liu, 2005). This issue will not be addressed further in this thesis and it is assumed any deployment can employ a location anonymisation technique if so required.

2.3.1 Greedy geographical routing

Greedy routing differs from traditional routing in the sense that there is no route discovery or topology discovery process. Packets are forwarded from node to node in the same fashion except that the geographical location of a hop's neighbours is used to make locally optimal decisions on where to forward the packet. Many of the techniques presented provide high packet delivery rates with no route/topology discovery overhead with the exception of that of the location discovery discussed in later chapters

Most Forward in Radius (MFR) (Hou and Li, 1986) is one of the simplest techniques for geographical forwarding. Routing decisions are made locally at each node encountering the packet and no global (or even significant local) knowledge of the network is needed. All neighbours' locations are known through information gathered from their beacon packets. When a packet is received at a node, it compares the distance to the destination between itself ad all of its neighbours. The neighbour with the smallest distance is chosen as the next hop.

```
ROUTEMFR(p)
1   bestDist ← DIST(self, p_dest)
2   bestNode ← none
3   while n in neighbours
4      d ← DIST(n, p_dest)
5      if d < bestDist then
6         bestDist ← d
7         bestNode ← n
8   if bestNode not equal to none
9      then forward p to bestNode
10     else return routing failure
```

Figure 8: Pseudo code for most forward with-in radius

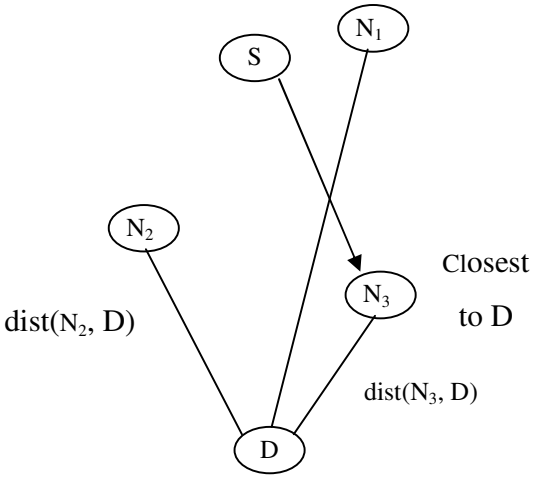

Figure 9: Illustration of most forward with-in radius

Figure 8 is a listing of the pseudo code for the MFR technique, executed for each packet received at each node in the route. Figure 9 is a diagrammatical illustration of MFR showing the choosing by *S*, of the closest node to *D*, N_3

Unfortunately, whilst MFR performs well in dense networks, there are some limitations in networks which do not reach the density requirements. These limitations will be outlined in the next section.

2.3.2 Limitations of MFR routing

MFR tends to fail or perform poorly in low density networks due to the inability of the algorithm to backtrack when reaching local maxima. Consider the example in Figure 10 where *S* sends the packet to the closer node *A*, who then decides B is the closest neighbour and sends it there. Upon reaching node B, who only has A as a neighbour, B finds that A is not closer to the destination and as a result the routing fails.

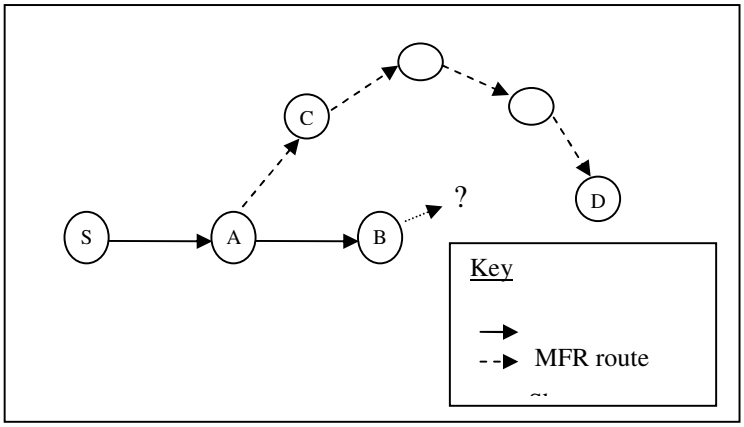

Figure 10: MFR routing failure illustration

One can see by examining the graph that the most suitable route would be via A and C but without any more information than local knowledge, A cannot see this. The only way to resolve this situation is to route around the void to discover if there is a closer node to D's last known location. Using global knowledge this is an easy task and one could have avoided the initial local maxima problem but as this is not possible in large scale networks a number of approaches have been examined that require only local decision making. The next section will examine these techniques.

2.3.3 Routing around voids

2.3.3.1 Right-hand rule

When a void is encountered, such as the earlier example, the only way to pass it is to route around it. This may also involve a certain amount of backtracking across nodes which have already been traversed. A simple method that satisfies the ability to route around with only local knowledge, and to backtrack, is that of the right-hand rule (Karp and Kung, 2000).

The right-hand rule states that on receiving a transmission at node B from node A, the next chosen hop is the node which is sequentially counter clockwise, about B, from the line $\overline{AB}$.

In the example given in Figure 11, the next chosen hop is D being the next sequentially counter clockwise node to the line. The rule will traverse the interior of a closed polygonal region in a clockwise edge order. In the case of an area void of nodes illustrated in the last section, the void is considered the closed polygonal region and will be traversed in a clockwise manner. The path taken around the void is called the *perimeter*.

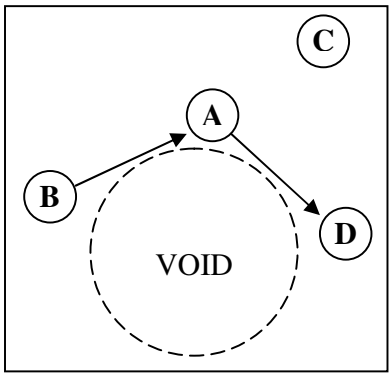

Figure 11: Right-hand rule illustration

The right-hand rule does not provide a traversal of all voids and in some cases fails. An example is non-planar graphs where links in the graph cross, which causes the right-hand rule to take a degenerate route through the network. One method to provide success in non-planar graphs is to ask nodes forwarding the packet to append their location to it and use this information to ignore links at a node which cross links already traversed, which the author terms the *no-crossing heuristic*. The no-crossing heuristic converts the non-planar graph into a planar one by ignoring selected links; however, ignoring such links may partition a small network and not find the possible path, although it finds over 99.5% of the n^2 routes in a network of n nodes (Karp and Kung, 2000).

Increasing delivery success further still is achievable by using additional decisions in the face routing technique, by generating a planar sub graph (Kuhn et al., 2003b, Kuhn et al., 2003a, Gao et al., 2001, Bose et al., 2001) and the use of face changes (Ryu et al., 2004).

2.3.3.2 Other methods of routing around voids

The Terminode (Blazevic et al., 2002) routing technique aims to find friends along the route that can be used to avoid voids. Nodes build up a table of friends with whom they can use to vary the path to the destination.

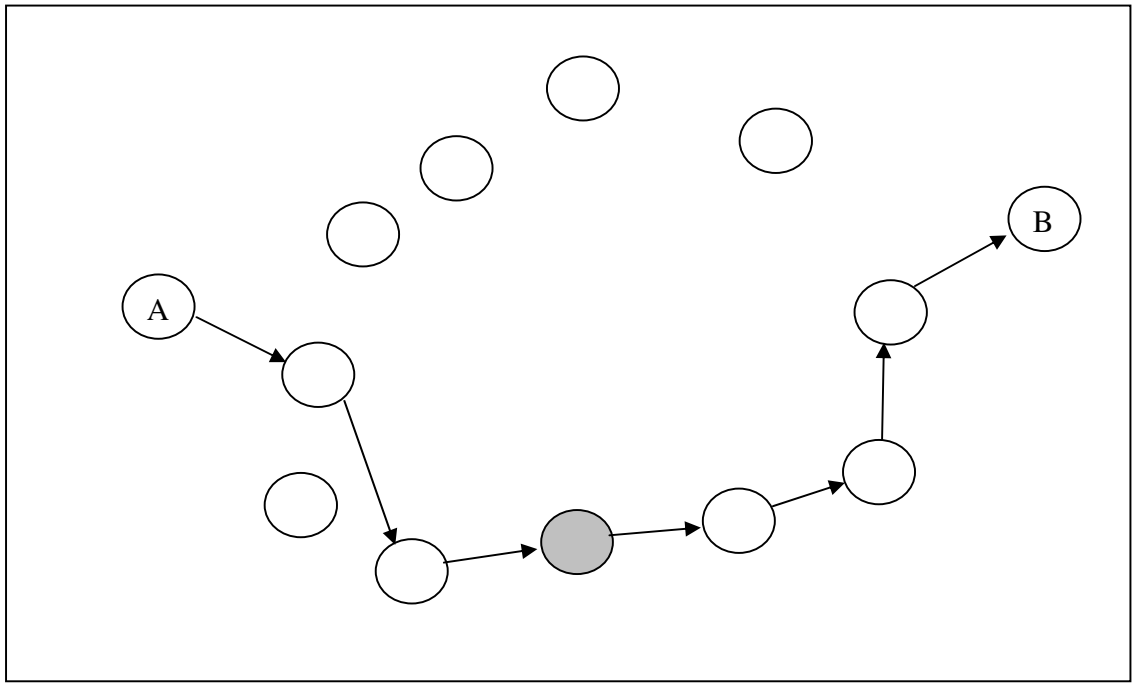

Figure 12: Terminode routing

The path to the destination is anchored at friends along the route that cause the path to avoid the voids. Their work proposes a local protocol that discovers and performs local routing, and a global protocol that uses MFR along with the friends to route across the network. Figure 12 illustrates communication between two nodes, A and B, via a friend coloured grey that causes the path to avoid the area void of nodes. The delivery success of their scheme is below that of others and incurs additional overhead due to the discovery of local routes and of friends, where earlier techniques rely on locally available information.

Geo-LANMAR is a geographical hierarchical routing protocol (Zhou et al., 2006). The nodes are divided into groups by proximity. When routing packets inside the group a

link-state algorithm is used, and geographical routing is used for inter-group traffic. This results in there being a tolerance to location inaccuracy where the destination has not wandered out of the group. Routing around voids is achieved by routing the packet to an intermediate group that is best placed to pass the packet on. A combination of local proactive routing techniques and geographic routing allows the packets to be routed around the void. The use of the link-state approach creates local overhead that is not present in pure geographical techniques that act on local knowledge.

2.4 Conclusions

Routing in ad hoc networks is a challenging task and whilst traditional methods work well for small networks, geographical routing is the only option for larger networks (Camp et al., 2002c). The difficulty with geographical routing, however, is that all devices need to have access to a GPS or localisation method. Fortunately, this is now readily available with the upsurge in the use of satellite navigation devices and the implementation of such features into mobile phones. Geographical routing is now a feasible alternative to traditional methods and certainly the only scalable method for millions of nodes.

The only remaining problem for geographical routing is the discovery of location information. The next chapter surveys the existing literature on location servers and the remainder of the thesis describes the new technique for providing this information. It will show the weaknesses with existing schemes and later the new SOLS technique will address many of these.

3 Location discovery in ad hoc networks

3.1 Introduction

As discussed in the last chapter, geographical routing is currently the only feasible solution for routing in large scale ad hoc networks. For this to be successful each node needs to be able to discover the location of the node it wishes to communication with. A number of routing protocols assume that location information is available and propose no method of obtaining it, with some assuming this is achieved at zero-cost (Camp et al., 2002b, Karp and Kung, 2000). This is not a realistic expectation except in simulation or static networks and therefore a location server is needed.

The approaches to location services are broadly broken into two categories much like routing protocols: reactive and proactive approaches. Reactive protocols aim to discover the information only when needed whilst proactive approaches have a continuous overhead in attempting to maintain location information. These proactive approaches are broken further into two subcategories: location databases and location dissemination techniques. Location databases provide a service for looking up a node's location whilst location dissemination schemes attempt to disseminate information amongst all nodes. .

An early technique for location services was to simply flood the network with location information. Distance Routing Effect Algorithm for Mobility (DREAM) is an early algorithm that belongs to this category (Basagni et al., 1998). DREAM improves on the basic flooding of the network by dividing the updates into two categories: short-lived and long lived. The short-lived updates are transmitted frequently and travel a certain distance from the node and no further while the long-lived updates are transmitted less frequently and travel the entire network. Therefore, the further one is from the node, the less accurate the position it knows but when sending location queries to its last know location, the query will encounter nodes that are nearer and have a more accurate position through the short-lived updates. The algorithm uses a basic form of prediction based upon the node's last known location, speed and direction from which it can calculate an

expected location some time later. Unfortunately, DREAM is not scalable to large networks due to its use of flooding.

The Routing Internet Protocol (Hedrick, 1988) requires routers to share information with one another on networks they can reach. Much like proactive ad hoc network routing protocols, its aim is to dissemination information amongst all routers. The Simple Location Service (Camp et al., 2002b) is similar to RIP and proactive ad hoc routing protocols in that it aims to dissemination information about all nodes' locations to all other nodes. In a small static network this scheme permits quick location discovery and minimises delay; however, like the routing protocols in this group, highly dynamic networks present difficulty in that changes are slow to disseminate through all nodes. In large networks, the overhead on each node and update cost is $O(N)$ where N is the number of nodes in the network. Such a scheme therefore does not scale to very large networks and is best suited to smaller less dynamic networks.

Reactive routing protocols were successful because information was always recent and overhead was only incurred when necessary. Drawing upon this to develop a location server, a number of authors have proposed approaches that are similar to their cousin routing protocols. The Reactive Location Service (Camp et al., 2002a, Fußler et al., 2001, Lochert et al., 2003) is similar to AODV (Perkins and Royer, 1999) in its technique. A node shares its location with its neighbours through hello messages so they can make local forwarding decisions. When a node wishes to find the location of another node, it launches a broadcast search to all nodes in the network for the information. The advantage of this technique is that the information is always up-to-date and recent which. The overhead in any broadcast search is $O(N)$ and although this can be reduced marginally through optimistic broadcast schemes (Fehnker and Gao, 2006, Ryu et al., 2004) it inevitably places a limit on the scalability of the network.

Using a proactive dissemination or reactive discovery protocol both require participation of all nodes, or a large number of nodes in the network. Subsequently, the overhead increases with the number of nodes and when nodes have limited communication

capacity this will result in a network which does not scale. The only remaining scheme is the location database and is examined next.

In these techniques, a subset of nodes is chosen by each node according to a set of criteria that will be used as its location servers. These techniques impose an overhead on a subset of nodes within the network rather than requiring the participation of all nodes. Typically, these nodes store the location information and are updated frequently so they may be queried by other nodes. This chapter will examine a number of location database location services as these are the only methods that have potential to scale to very large numbers of nodes.

3.2 Quorum system background

Before examining specific location servers it is important to briefly cover quorum system theory, upon which many of the techniques are based. Quorums are a tool for increasing the availability of data through replication. The data is replicated in some fashion to servers so that the failure of one or more servers does not necessarily result in the loss of the data. A quorum system is made of two sets of servers, a update/write set (U) and a query/read set (Q), whereby in a *strict quorum system* the two sets always intersect. The read set is a set of servers that are queried for the information, and assuming this set intersects with the write set that was sent the information, then there will be a server in R that can return the information (Amir and Wool, 1998).

Consider that the universe is a set of servers S, and a quorum is made up of two sets, $Q,U \subseteq S$. Updates are sent to a server $u \in U$ and queries are sent to $q \in Q$. In a strict quorum system, $Q \cap U \neq \emptyset$ with certainty while in a probabilistic quorum system they only intersect with high probability (ε). A quorum system where the two sets intersect with ε probability is referred to as a *probabilistic quorum system, or ε-intersection quorum system,* and is studied extensively in *(Abraham et al., 2004)*.

Quorums are traditionally assessed on three criteria:

- Fault tolerance – the ability of the quorum to deal with server failures (Barbara and Garcia-Molina, 1986).
- Load – the amount of communication required between the servers (Naor and Wool, 1998).
- Failure probability – the probability that the system is disabled when individual severs crash independently (Barbara and Garcia-Molina, 1987, Peleg and Wool, 1995).

Two types of faults may occur; a crash fault is a failure of the system such that it may no longer respond in any fashion; a Byzantine failure (Lamport et al., 1982) is one whereby the system is not functioning according to its specification which maybe due to a malfunction or malicious intent.

There is a trade off between load and failure probability/fault tolerance with it being impossible to achieve optimality for both (Abraham et al., 2004). Systems with higher overhead will provide higher fault tolerance and vice versus.

Strict quorum systems, whilst providing guarantees on queries and updates are difficult to implement in highly dynamic environments (Haas, 1997). This is due to the frequent membership changes of the update and query sets. Ad hoc networks are highly dynamic in nature and implementing strict quorum systems is not possible as it incurs prohibitively high overhead. Therefore, we can only examine probabilistic quorum systems as solutions to the problem.

Probabilistic quorum systems may give out of data due to their probabilistic approach to intersection (Karumanchi et al., 1999). As intersection is not guaranteed queries or updates may not complete fully and therefore return out of date information; however, it is worth noting that out-of-date location information is not necessarily a cause of routing failure. In fact, geographical routing systems can tolerate some inaccuracy in location and so it is important to ensure that this incomplete update is within these bounds.

3.3 Home-region

The virtual home-region (VHR) was originally proposed as part of the Terminode project (Blazevic et al., 2001, Giordano and Hamdi, 1999) to provide location information for their large-scale network protocol.

Server Selection: A certain geographical area of the network for each node is designated as a home-region. The nodes within this region are then required to act as servers, storing location updates from the node and answering queries for the information from others. In the Terminode approach (Figure 13), nodes within a radius R of a designated point C_{VHR} are required to act as servers. The region is defined by a publicly known hash function $C_{VHR} = H(nodeID)$ of the node identifier; however, this function is not defined. Nodes leaving this region lose their responsibility to store the location of the node who's VHR they were in.

The radius, R, is adjusted depending upon the node density to ensure that there are a certain number of nodes within the region, not so few as to be unable to maintain the information, and not too many so as to have high communication overhead; however, the author does not define how R is adjusted.

Querying location server: Assuming a node, A, wishes to communicate with another, B, then A calculates B's home region using the hash function H. A then sends a query towards B's home region and the nodes within the VHR reply with the last recorded location; A and B then enter into direct communication using geographical routing.

Updating the location server: If B wishes to update its location server it will send a location advertisement to the VHR. Nodes in the VHR will store this information for any nodes requesting it.

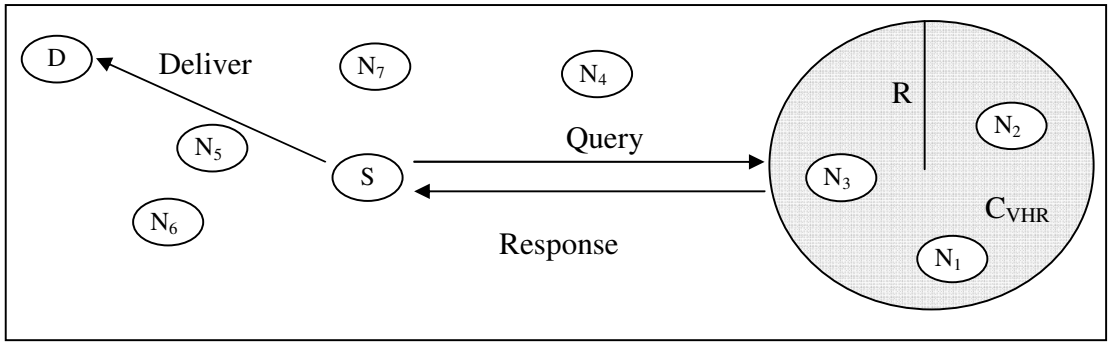

Figure 13: Home-region location server

The authors do not describe the mechanism for updating or querying the location server, nor do they provide a definition for the hash function. No mechanism is proposed to handle node mobility or failure and no simulation results are provided to indicate the performance of such a scheme. However, assuming a correctly chosen function H, and a sufficient distribution of nodes, such a technique would scale having fixed communication overhead relating to the size of the VHR. The only part of the system that may increase with the number of nodes is that of the number of hops to the VHR, assuming that the geographical size of the network increased with total nodes.

3.3.1 Improvements to Terminode's home-region

A number of authors have attempted to improve on the Terminode approach to home-regions. One author (Wu, 2005) proposes improving the approach by subdividing the VHR into seven hexagonal sub-regions. When updating the location server as a whole, the update packet is broadcast only once in each sub-region. This is opposed to the idea of every node in the VHR re-broadcasting the update and hopes to reduce the redundancy in flooding of the VHR.

Each sub-region will have a diagonal equal to the transmission range of the nodes. Given the diagonal of a sub-region (r), then the diagonal of the VHR will be R given as:

$$R = r\frac{\sqrt{7}}{2} \qquad (1)$$

This provides for 7 sub-regions inside the VHR. When a periphery node receives an update, it acts as a proxy, adding its sub-region number to the message and re-broadcasting it. Each node that receives it will not rebroadcast unless it is in a different sub-region not listed in the packet. This has the effect of there being only one broadcast per sub-region, and so in the scenario above the update overhead is seven broadcast packets. A query is achieved in much the same way searching for a node containing the information and is similar to that of a sequential paging system used in cellular networks (Rezaiifar and Makowski, 1997). When a query is received by a sub-region, it searches the other sub-regions sequentially until the information is found. The author proposes several means to improve system robustness but these are limited to increasing the number of servers participating in the VHR, and not to methods of recovery developed in this thesis.

An examination of a clustering technique for home region systems was undertaken by (Sivavakeesar and Pavlou, 2004). The technique requires each node to assess its association to the virtual home region by measuring the amount of time that it remains in it. Using this metric, and others such as its closeness to the centre the nodes elect a cluster head in a centralised or decentralised manner, depending on whether the present cluster head is available. This ensures that the cluster head has uniform coverage or is a bounded number of hops away and so can ensure efficient updating. Each cluster head monitors members in the region and periodically generates location updates for nodes who share the same VHR elsewhere, so that the number of location updates is minimised; however, this is only used for nodes with low mobility, others must generate their own updates. In addition, neighbouring cluster heads share information to aid resilience and to cope with the scenario of the home region being void of nodes. The authors compare their work with the Grid Location Service (GLS) described in section 3.5 of this thesis. Their approach performs significantly better than GLS in the scenarios they examined although they only examined the effect on increasing node count and not the effect of node speed which arguably is the most important factor. Their scheme is significantly different in design to GLS and their comparison is questionable because GLS stores

information network wide and provides location search schemes that do not depend on a hash function. In light of this, their approach would be expected to out perform GLS given that the traffic is mostly localised.

Scalable Location-based Routing Protocol or SLURP (Woo and Singh, 2001) is one of the earlier improvements to the home-region. The terrain is divided into a grid with equal size squares, and each of these squares can be a home-region for a set of nodes. The approach proposes that a node update its home region when it crosses square boundaries, and that the node informs its old region and new region of this movement. Informing other nodes about location changes is achieved by broadcasting the information. To handle areas that are void of nodes, the authors propose using neighbouring squares when the target square is vacant.

3.4 Horizontal/Vertical quorum (HVQ) location server

The horizontal and vertical quorum (Das et al., 2005, Liu et al., 2006, Stojmenovic, 1999) is similar to the home-region but provides a means to discover the server without the use of a hash function. A column of certain thickness is chosen within the network, and all nodes within this column become location servers; to query these servers, one simply sends a packet in both horizontal directions in the hope that it will intersect the column.

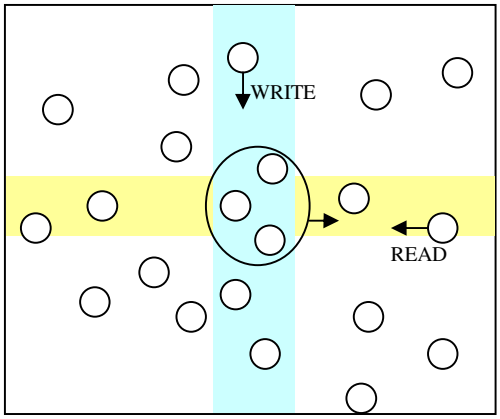

Figure 14: Horizontal and vertical quorum location server

Location Server Selection: A node creating its quorum sends two creation packets, one north and one south. Each node receiving the packet becomes a location server and forwards the packet on in the north/south direction. A parameter *p* is set defining the thickness of the quorum, where in the above example *p=1*. When p=2, any node receiving the packet also broadcasts it before forwarding it on, and every node receiving the broadcast also becomes a location server. For values of $p \geq 3$ this process is repeated at each node receiving a broadcast.

One problem with the above method is that the northernmost node maybe only locally most northern. To avoid such a scenario, FACE routing (Bose et al., 2001, Datta et al., 2001) is used to route around the problem, and then standard geographic routing is resumed. The drawback of this is that if the node is infact the northernmost node, then FACE routing will cause the packet to traverse the perimeter of the network. This will impose higher traffic demands on them.

Location Server Query: A node, *S*, trying to send data to a node, *D*, first checks its location cache to see if it has a recent location for *D*, and if so, begins sends data directly. If no location for *D* is known, or the location is out of date, the node initiates a broadcast search to nodes at most *q* hops away, so that any server *q+1* hops away will be found. If *D or D*'s server is not found within this range, then the search continues in an easterly and westerly direction. At each hop, the search packet is updated with the most recent location information available but the search continues until reaching the most easterly or westerly node. Once these nodes are reached, the strategy changes, and *S* starts a search for *D* based on the best information retuned during the *q*-hop search by routing a packet towards it. In addition, the most easterly and westerly nodes also route a packet towards *D* using the best information obtained during their search.

Location Server Update: An update is performed in the same manner as the deployment of the quorum. A node updates its quorum once a threshold number of links with its neighbour have broken, or new neighbours have been discovered.

Performance: The scheme exhibits near perfect load balance across the network with every node taking on near-equal responsibility. This is because updates are always sent in x or y dimensions, regardless of where the node is. When moving to mobile networks overhead remains approximately the same, whereas GLS's overhead almost doubles (Das et al., 2005). Therefore, this scheme is more scalable than the GLS.

Benefits: In a static network the technique provides an almost guaranteed way to discovering the location. In mobile networks the performance depends upon the deployment technique of the quorum, and increasing the thickness can improve this (Mauve et al., 2001).

Problems: Node mobility may cause the nodes within the column to leave, therefore making the data inaccessible through inability to discover it. The authors recognise this and propose the technique as a way to provide "almost guaranteed" location discovery in networks where all nodes are static, or moving in the same direction at the same speed.

Another problem is that of the locally northernmost identification problem. How can the algorithm determine that it has passed packets to the northernmost point and not to a local maxima if no more nodes are visible in that direction. Typically, perimeter routing will be used when encountering no more northern nodes to determine if it can be routed around; unfortunately, if indeed it is the most northern node, then this packet will be routed around the entire perimeter of the network. This causes extra load on these nodes and in the case of deployment of a quorum for every node, then each node on the perimeter will periodically encounter an update for every node in the network unless some means is employed to mitigate this. In a mobile network, this is not an easy task as it is frequently necessary to determine if the node is in fact the northernmost.

The nodes around the perimeter of the network are encountered during deployment due to the use of FACE routing and this can cause nodes around the perimeter to represent all nodes within the networks causing them to have traffic and memory capacity problems.

Also, in large networks, of tens of thousands of nodes, the cost of deployment and update can be exceptionally high due to the large number of nodes within the column. In addition, the number of hops required to traverse before encountering the column during discovery maybe excessive.

3.5 Grid Location Server (GLS)

The grid location server (Cheng et al., 2002, Li et al., 2000), also known as SLALoM, is a set of servers defined by a geographical grid and a predefined order of node identifiers. The system is composed of three components: location server selection, location query, and location update.

The area is first divided into a hierarchical grid that is assumed to be known by all nodes. The grid is such that an order-two square is composed of four order-one squares, and an order-three square is composed of four order-two squares, and so on. There is no overlap of squares, and so a node in any order-n square is a member of only one order-$(n+1)$ square, not four. This ensures that each node is a member of only one order-n square, for any n.

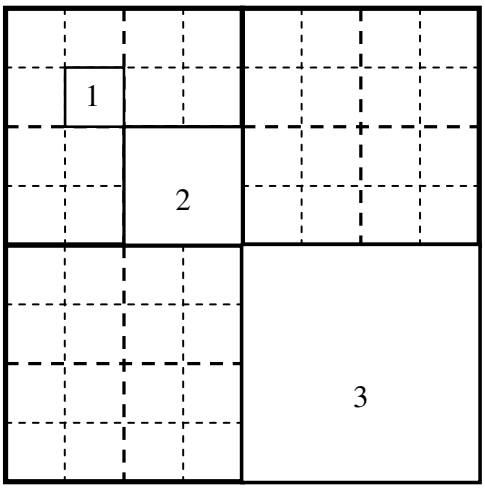

Figure 15: Grid location service

LS Query: The query process is initiated on-demand like other location services in its category. A node (A) wishing to communicate with another (B) will forward its location query to the node with the lowest ID which it is acting as a location server for. The node receiving this repeats this process until a node which is B's server is reached at which point the query succeeds.

This technique assumes that nodes are able to scan all nodes with in any order-n square; in fact this is not necessary as nodes can update and create their location servers without knowing the identities of the servers. Let us take the example of B trying to recruit a server in an order-*n* square. B sends a message towards the square and on being received by a node it is forwarded on in the same fashion as the query, with location servers being recruited in each order-n square.

Nodes within an order-one square use a local routing protocol to share information on their locations with one another in that square. Nodes can then recruit servers in their order-one square, and once this is complete, then nodes may then recruit servers in the order-two square. The only requirement before recruiting servers in an order-$(n+1)$ square is that servers have been successfully recruited in the order-*n* squares.

Query failures: The authors describe two types of failures**.** The first occurs when a node receiving a query has no information on other nodes' locations due to the last update failing and the information being timed-out. This occurs rarely and can be overcome by using old-information in a "last-ditch attempt." The second occurs when a node that the query is being forwarded to has moved outside of the square which it was previously know to reside. Upon leaving the square, the node broadcasts a forwarding pointer to all nodes within the old square, that they record and use to forward information to the new square.

Update period: To avoid excessive updates in mobile networks the author defines an update-scheme based on distance travelled. A threshold distance *d* is defined so that a

node updates its order-one square after travelling this distance, and its order-two square after travelling a distance of $2d$. In general, a node updates its order-n square after travelling a distance of $2^{n-1}d$. Therefore, updates are sent to closer servers more frequently than those farther away.

Caching: Nodes maintain two tables, that of locations for various node identifiers that it is a server for, and that of a cache containing nodes' respective locations for updates overheard. The cache table is used only when initiating a query and so the information is timed-out quicker than the recommended period.

Performance: Their analysis using the ns-2 simulation showed that the query success ranged from near 98% at 1m/s to approximately 90% at 50m/s when using the Random Waypoint Mobility Model. Overhead was measured as two to three protocol packets per second per node in all scenarios in contrast to DSR which ranged from less than one with a 100 node network to 25 packets/sec with a 500 node network. CBR delivery ratio fell from 98% with a 100 node network to 85% with a 600 node network, whilst DSR fell from 95% to 25% respectively. The scheme tends to exhibit higher load in the centre of the network due to some updates being sent across the network in diagonal grids (Das et al., 2005).

Problems: In large networks, discovery path lengths can be very long and a failed location server can result in discovery failure. The authors propose no approach to handle node failure and mobility. Additionally, because the approach requires an increasing number of servers as the network size increases, the approach is not scalable to a large number of nodes.

A scalability study (Philip et al., 2004) indicates that GLS performs marginally better than the home location system SLURP in terms of delivery success but incurs significantly higher overhead as one would expect.

3.5.1 Improvements to Grid location server

Elf is an approach that proposes to replace the idea of near and far servers with forwarding and terminal servers (Philip and Qiao, 2003). A number of servers are elected as primary servers, whilst the remainder are forwarding servers that forward the query to primary servers. Their technique outperforms GLS in the average case by reducing signalling overhead.

An adaptive version of the grid approach (Seet et al., 2005) permits the system to dynamically choose how many location servers it will have based upon demand for the information. The authors propose that the system always maintains at least one location server, but that when requests arrive from areas which currently do not have one in their order-n square, then one will be created to reduce overhead. In addition, the improvement permits the use of more than one square at borders of the network due to the often lower than average node density, although this is often an artefact of using the Random Waypoint Model in simulation; however, in a city lower densities are expected outside of the centre so the same applies. The simulation of these improvements suggests significant performance benefits in terms of delivery success and update cost over SLALoM and SLURP; however, the latency was larger than SLALoM due to the initially longer discovery paths.

3.6 Other location servers

The literature proposes many approaches to the problem of location servers with the above three being the most successful. Outlined in this section are a selection of a few other location services that are based upon similar concepts.

3.6.1 Location tracking quorum

A location tracking approach using quorum systems was proposed by (Lee et al., 2003). Their approach proposes four methods for choosing quorums for storing location information in an ad-hoc network. Queries are sent to the query quorum Q, and on receiving a reply from all nodes $q \in Q$, the query is said to be complete. Updates are also performed in a similar manner, by sending it to all nodes $u \in U$ in the update quorum U.

As the quorum is only updated by the node whose location it is serving then simply time stamping updates will enable the quorum to distinguish new information from old. The four methods used to choose the quorum used for updates and queries are as follows:

1. **Grid Bi-quorum (SQ1):** This system is very similar to the Horizontal and Vertical (HVQ) quorum presented in section 3.4. The network is divided into a grid $\sqrt{n}$ x $\sqrt{n}$ that is know globally, assuming n is the number of nodes in the network. The columns of the grid are used for sending updates, and the rows are used for sending queries. Nodes are placed and remain immobile, so that there is one in each intersection and thereby guaranteeing intersection of the row and column. As with the HVQ this system is not highly scalable, and may suffer low fault tolerance as node failures result in no guarantee of the intersection property. Quorums will be of size $\sqrt{n}$.
2. **Strict Quorum Construction with Increased Intersection (SQ2)**: This is the same as SQ1, except all the nodes in the column and row are used as location servers, rather than just the column. The quorums are then of size $2\sqrt{n}-1$.
3. **Using Unreachable Nodes List (SQ3):** All nodes maintain a list those nodes who are unreachable (Unreachable Node List: UNL) and use this information heuristics to choose the quorum:
    a. Eliminate quorums containing nodes in UNL, and choose a quorum randomly from those remaining (Eliminate-then-Select: ETS).
    b. Choose a quorum at random, and remove nodes from the quorum that are members of the UNL (Select-then-Eliminate: STE).
    c. A hybrid using ETS for updates and STE for queries. This hybrid approach is denoted **SQ3.**
4. **Dynamic Quorum (DQ):** Quorums are chosen independently by uniformly random selection from the network, excluding those nodes in the UNL, to form a quorum of size k where k is an input parameter to the system. When quorum members are not responding, other nodes are repeatedly tested for reachability and if found they replace it.

The authors simulate their system using ns-2 with a network of 100 nodes, 25 of which participate in the quorum system. Their key findings were:

- Larger quorums do not necessarily result in higher correctness; although this was attributable to network partitions, and the scenarios simulated were only small.
- Topology change has a significant impact on the performance of the quorum systems.
- SQ performed better in connected networks due to increased intersection.
- DQ performed better in all scenarios.

Overall, they found that a dynamic quorum system performs better in all scenarios tested for all measured parameters. Although their simulations were of small scale, their work highlights the impact node mobility and link failures can have on quorum systems, and any successful system will need to appropriately handle these.

3.6.2 Uniform Quorum System

Uniform Quorum Systems have been used for managing mobility in ad hoc networks, along with randomised databases (Haas, 1997). The authors propose creating several strict quorum systems connected by a backbone that nodes may use to obtain and update location servers. The quorums contain the location information, while the backbone is the interconnection of these quorums to aid discovery. A node wishing to discover or update its quorum sends the request to the nearest node participating in the backbone, which forwards it to the relevant quorum. The authors propose that the backbone is discovered using flooding to find nodes most suitable; however, the precise details of this, and maintenance of the backbone were outside the scope of their work.

The above approach is a form of strict quorum system as defined in 3.2 and as such may suffer from low availability in dynamic systems. Also, due to the use of flooding, and the maintenance of a backbone, this scheme is thought not to be scalable due to the high load potential on the backbone as the number of nodes increases, along with the proposal of

flooding the network for backbone discovery, which is not scalable in large-scale ad hoc networks. Their work examines a theoretical model of their system and provides no simulation or real data on the performance.

An improvement on the above scheme is (Karumanchi et al., 1999) which uses probabilistic quorum systems to tolerate node failures and uses multiple quorums to try to solve partitioning, but as noted in 3.2 this may result in old data being returned and accessibility cannot be guaranteed when the network is completely partitioned. They also keep track of nodes that are unreachable.

3.6.3 Atomic memory and distributed storage

Distributed atomic memory is an approach whereby one can simulate a memory device over a distributed set of devices. For example, one typically addresses memory in a PC by use of the *memory address*, the byte offset from the beginning of memory. This memory is only available to that computer and so unless one specifically writes software to share the memory then no other PC can access it. It is not unusual for several programs to be running on a computer that share access to a section of memory, termed *shared memory*, and use this to collaborate or operate on the same data set. If one wished to write software that worked in this fashion in a distributed scenario then extra software would have to be written for other PCs to access the shared memory. Consider the scenario where there is a hundred computers all wishing to have access to and potentially modify the same data, but if they rely on one node in an ad-hoc network to serve up this data this not only causes extra load on that node, but if it fails then all the data is lost. An alternative to this is distributed atomic memory, one still retains the idea of an address that points to a byte offset for the data, but instead of this offset being used it is passed to a hash function. This hash function will then return the node within the network that contains the data, and in effect this function simply maps byte offsets to nodes. There is an address space like traditional memory, but instead of looking at a byte offset we use a hash function to determine which node in a network holds the information. If any node fails then ideally their data would have been replicated at neighbouring nodes, and if not, then only that section of memory is lost.

There are several challenges with distributed atomic memory, the most pivotal of which is the hash function. In static networks, a hash function can be defined easily on creation of the network, but where the network is dynamically changing this becomes more difficult.

Researchers at Massachusetts Institute of Technology have looked at several approaches to provide data storage in ad-hoc networks. Their first proposal, *GeoQuorums* (Dolev et al., 2005a), provided atomic memory by assigning a fixed geographical point as a storage location for a particular datum. They present the details of their technique and a discussion of the expected performance; however, no simulation or real data is presented. Their next technique, titled *Virtual Mobile Node* (Dolev et al., 2005b), proposed a virtual node containing the datum that is able to move around the network, jumping from node to node. Finally, *Virtual Stationary Automata* (Dolev et al., 2006) are virtual immobile nodes distributed across the network in a uniform fashion. These immobile nodes may contain automations and data and may be interacted with by real nodes. Their systems propose means of providing atomic memory in ad-hoc networks and their technique could be used for location discovery; however, their focus is not on location discovery and their work provides only theoretical descriptions of the service with no simulation or real data.

3.7 Updating location servers

All the methods presented in previous sections provide a means to store location information. As a node moves this location information becomes out of date and so a node must update its location server so that the information retrieved remains useful. A location server needs to be updated frequently enough so that the information allows the routing protocol to be successful in delivering packets. Updating the location server continuously satisfies this criterion but clashes with the requirement to reduce the overhead on the network. An optimal update frequency is needed that minimises the overhead while maximising the usability of the data stored at the server.

Ideally the location server would be sent an initial position and an exact mobility profile of the node so that the node's location can be accurately predicted. Attempts at addressing this have been proposed in a number of papers from using neural networks (Capka and Boutaba, 2004) to model a user's behaviour and other user profiling techniques (Kwon et al., 2005) to using a node's speed and direction to estimate its current location (Rezaiifar and Makowski, 1997). Unfortunately, predicting a user's location is a difficult task and an incorrect prediction will result in total routing failure to that node.

Total routing failure is not an acceptable option in any communications network and small downtimes are often intolerable to users. Although mobility prediction techniques could be used to a certain extent, a backup or contingency is necessary should this fail. One method would be to have the node predict its current position given the profile stored at the location server, and if this differs significantly then to perform another update.

In this section, the methods for updating the location server without mobility prediction are briefly examined given that any mobility prediction scheme will be heavily dependant on the scenario that it is deployed in. These are broken down into two categories, unintelligent fixed periodic updates and those which take some element of the node's behaviour or environment to determine when an update is suitable.

3.7.1 Periodic updates

This is the simplest, and by far the most common approach used within the literature whereby a node periodically sends updates to its location server (Blazevic et al., 2001). Assuming an update-interval of time u seconds, and the current time t seconds, the next update will be at $u + t$ seconds. Typically a jitter is also introduced so that neighbouring nodes that happen to send updates at the same time, do not continue to do so and therefore avoid continued collisions.

The disadvantage of this approach is that updates will be sent regardless of the node's mobility. If the entire network is static, nodes will still send updates imposing an unnecessary overhead. Likewise, if the network is highly mobile then the chosen interval may not be sufficient and the network could experience total routing failure.

This technique is however popular due to its simplicity and when examining location servers' performance when node behaviour is strictly controlled it will succeed.

3.7.2 Distance

As previously stated, the time between updates should be dependant upon the nodes' mobility. If a node is moving quickly it should update more frequently than when it is moving slowly to allow routing to succeed. The distance-based approach (Blazevic et al., 2001) requires a node to update its location server once it has travelled a set distance *d*. This is usually achieved such that given a node *n* sending an update at time *t*, and the node's location ($n_x(t+c)$, $n_y(t+c)$) at time $t+c$, where *c* is a time between updates dependant on speed and direction, an update is sent if the following criteria is met:

$$\sqrt{\left(n_x(t+c)-n_x(t)\right)^2+\left(n_y(t+c)-n_y(t)\right)^2} \geq d \qquad (2)$$

A node frequently calculates the Euclidean distance between the last position sent to the location server and its current location, and if this is above or equal to a threshold *d*, then an update is sent. The value of *c* will therefore vary depending on the node's speed and direction.

This approach is slightly more complicated than the periodic approach but not significantly so and memory requirements are of the order of a few bytes. This method is therefore preferred over the periodic approach when node mobility is not strictly controlled.

3.8 Conclusions

A number of schemes were described in this chapter for serving up location information in ad hoc networks. The focus of this thesis is on large scale ad hoc networks, those networks that could scale to millions of nodes. The broadcast schemes become prohibitively expensive when networks are large, with every node needing to know the whereabouts of every other. The alternative is to store the information on a subset of nodes in a way that anyone can determine those likely to have it. Two schemes, Grid and Horizontal/Vertical quorum provide a method for discovering the location of the location servers but their overhead increases with network size, which means due to limitations in capacity they cannot scale very large numbers of nodes.

The only scheme examined which provides scalability to an large number of nodes is that of the Terminode home-region. A set of nodes in a circular region are utilised to store the information, and the location of them can be determined by use of a hash function; however, their approach does not address issues that will cause discovery failure in ad hoc networks. These include node failures and mobility, leaving the system likely to fail in realistic scenarios. In addition, when a network contains areas void of nodes, and the hash function returns a location inside the void, then the system fails.

SOLS, described in the next chapter, will solve many of these limitations addressing them specifically using a novel nature inspired approach.

4 SOLS: A Self-Organising Location Server

4.1 Introduction

This chapter will outline the proposal for a form of home-location server that differs from those already existing in a number of ways. One can draw similarities between this system and that of the home-region approach as it attempts to maintain the server in a particular location; however, this is where the similarity ends. The home-region approach broadcasts the data to be held to all nodes within a radius of a particular point in the hope that should anyone wish to query the server, these nodes would still be in the same location holding the same data. This chapter makes no such assumption and attempts to improve upon the home-region approach in the following ways:

- Fault Tolerance: Firstly, one cannot assume that nodes do not fail and can hold the information for an indefinite period of time. Therefore, this approach will address issues of fault tolerance.
- Mobility: Secondly, one can also not assume that nodes will remain in the same location whilst participating in an ad-hoc network. It is possible that a number of nodes will remain fixed but generally the worst case is that all nodes are moving at high speed.
- Overhead reduction: The home-region approach is not particularly costly in overhead, only generating a number of broadcast packets within the home-region per update: however, traditional quorum systems can require constant monitoring of a quorum's servers and so this approach attempts to reduce this by adopting a probabilistic quorum system whereby non-failure is not guaranteed.
- Self-organisation: Ad-hoc networks are inherently self-organising and distributed in nature because any node may leave, join, or move within the network. Therefore, using an approach that relies on nodes to perform management functions is not feasible unless the failure of that node to perform the function can easily be recovered from.

Initially, the inspiration is drawn from self-organisation observed in nature such as ants and birds. Ants do not communicate directly with one another but are able to perform complex tasks by modifying their environment to influence the behaviour of other ants. Ants lay pheromones as they roam which signal indicators to fellow ants on paths to food. Similarly in flocking behaviour observed in birds, the birds do not communicate directly but instead observe one another to make decisions. The birds observe each others' position and direction and use this information to choose their own path; they too communication indirectly.

Ad hoc networks have limited capacity and it is important to keep overheads to a minimum if more capacity is to remain for data. If ants and birds can communicate without direct communication it should be possible to enable a group of agents (Franklin and Graesser, 1997) carrying data to co-operate to remain in a particular area and to survive. If there is no central organisation, and each agent can only observe its local environment, then already the need to communicate is reduced. The challenge is turning that local self organising behaviour into something useful, and in this thesis it is successfully used to reliably store data at a specific point in an ad hoc network.

The system comprises a number of agents, each of which is independent and can move from node to node deciding by itself when and where to move to. These agents are assigned a geographical home-location (HL) and attempt to remain as close to this point as possible by moving from node to node. Upon reaching this location, they are able to serve up location information about the node which created them; however, because the failure of the node which the agent is currently on would result in loss of the data, then the agent creates clones, each of which too acts individually but they remain on neighbouring nodes. A group of agents, all serving up information for the same node, are called a Group of Location Agents (GLA). Two agents of the same group cannot be present on the same node, and so they observe their environment before making decisions on where to move so as to satisfy this criterion.

In this approach, a location server will be made up of a number of agents. Consider Figure 16 where there are five nodes, two of which have created a GLA for themselves and assigned them differing HLs (indicated by different colours). Each GLA should reside on nodes around its HL and the diagram shows that there are three agents in each GLA and they all remain on the closest nodes to it. GLAs are independent of one another and as a result two agents of differing GLAs may reside on the same node and be unaware of each other. In addition, an agent is only able to observe agents of the same GLA on other nodes.

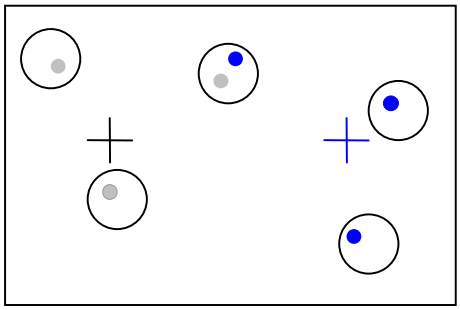

Figure 16: Two GLAs

To observe one another, each agent modifies its host node's beacon packet. In the packet it places the identifier of the node that created it. As each node stores every beacon it receives for a period of time, the packets are made accessible to the agents so that they can find out which agents are on which neighbouring nodes. It has been shown (Fußler et al., 2001, Chawla et al., 2006) that it is not necessary to use beaconing to facilitate geographical routing, so a justification for its use in large-scale ad hoc networks is provided in appendix section 8.1.

This chapter will examine the feasibility of the system outlined above and determine a possible implementation. Initially, how close the agents are able to get to the geographical point will be examined, because if agents are too far from it then nodes looking for the information will be unable to find them. How agents will self-organise to achieve their common goal of surviving and serving information will be examined along with methods of implementation. The number of agents required to successfully mitigate

node failure and a technique for determining this to minimise overhead without co-operation will also be examined.

The terminology and notation used through out the thesis is described in appendices 8.1 and the simulation parameters are described in 8.2.

4.2 Proximity to home location

As described in the introduction, the agents need to be as close to the HL as possible to avoid the need to perform broadcast searches to discover the information. Therefore, it is important that the distance an agent is able to get to the HL is determined. This section will examine how close an agent is able to get to a particular geographical point, and its ability to remain within the vicinity (Owen and Adda, 2005) over time.

4.2.1 Implementation

For a server to be queried with no more than a one-hop search, an agent must be within the maximum transmission radius of the geographical point. The agent's identifier is defined as a_{ID}.. If the agent is to store information about a node's location (n_x, n_y), then the data payload will be as follows:

$$a_{ID} = \left(n_{ID}, n_x(t), n_y(t)\right) \tag{3}$$

Whenever the agent detects a change in its hosting node's beacon table, or when it migrates to a new node, it executes the MIGRATECHECK() method. A node's beacon table will change when a neighbouring node has moved from it, or a new neighbour has moved into the communication range. The function examines all the node's neighbours to see if there is a node closer to the HL and if there is one then the agent will migrate to that node.

```
MIGRATECHECK()
1   dist ← ∞
2   closest ← none
3   for each n in neighbours
```

```
4       if DIST(n, HL) < DIST(self, HL) then
5           dist ← DIST(n, HL)
6           closest ← n
7   if dist != ∞ then
8       MIGRATETO(closest)
9       return true
10  else return false
```

Figure 17: Check whether a migration is needed

The object *self* refers to the current node, and DIST() calculates the Euclidean distance between two nodes. The code will cause the agent to move from node to node to try and remain as close as possible to the HL. The function looks for the closest neighbour to the HL if one exists and then a call to MIGRATETO() is performed, which sends the agent to it. A value of true or false is returned from the function indicating whether the agent migrated or not, and this value will be used in later parts of the algorithm.

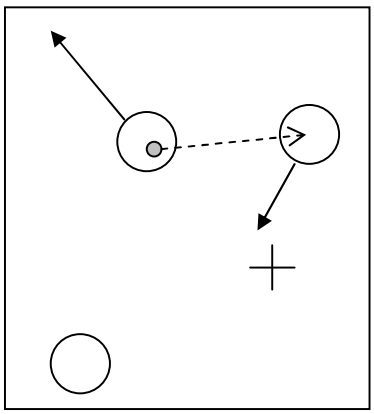

Figure 18: Migration of agents

Figure 18 illustrates an agent migrating when another node becomes closer to the home location. The simulation will examine the average distance of the agent to the HL by recording the location of the hosting node. Each node generates a server agent and allocates it a uniformly random home location where it is to reside. The parameters are varied to examine what effect speed and beacon rate have on the distance to the point and the overhead incurred in migrations.

4.2.2 Simulation results

Figure 19 shows the results obtained when speed and beacon rate were varied. The beacon rates were chosen to clearly illustrate the variance in distance and are an arbitrary choice. Beacon rate affects the accuracy of the information nodes hold about their neighbours, and lower beacon rates will result in less accurate information. This is illustrated in Figure 19 where lower beacon rates result in the agents being less able to remain near their home location. According to the results, for networks where the speed may reach 10m/s, a beacon rate of less than five seconds will allow the agent to remain within the maximum transmission radius of 100m. Using a beacon rate of one second resulted in the agent being less than 50m from the geographical point.

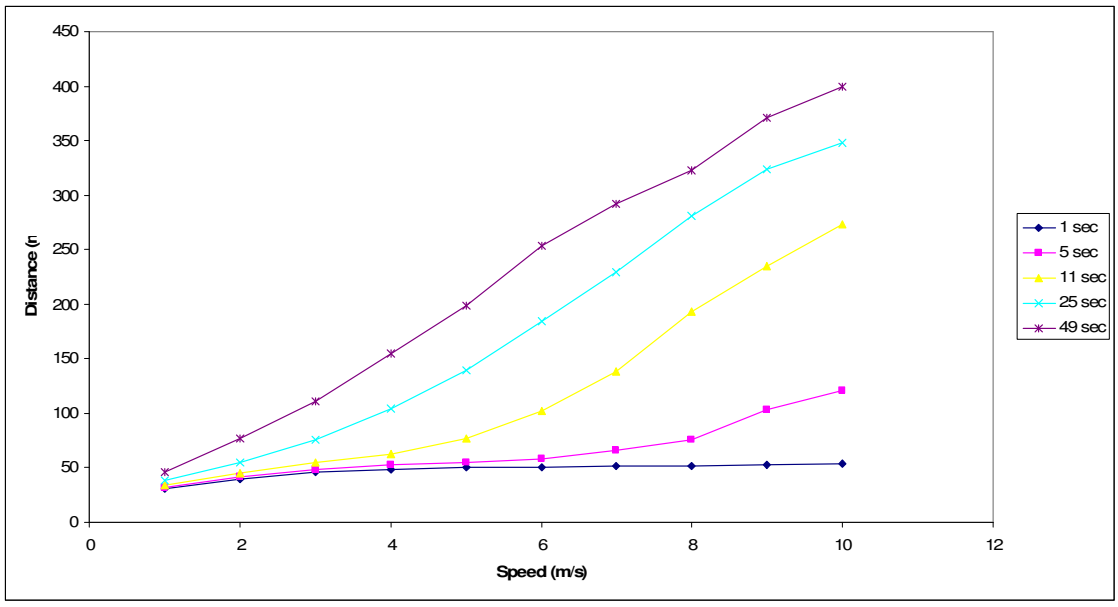

Figure 19: The effect of beacon-rate on distance from the centre

Figure 20 shows that with more frequent beaconing, the migration overhead is greater. When beacon rate is frequent the number of changes in the node's beacon table occurs frequently and therefore the agents migrate more often. When the beacon rate has low frequency then agents migrate less frequently and as a result are often further away from

the HL. The figure shows that with the least frequent beacon rates, the number of migrations levels off at quicker speeds; the reason for this is that with less beacon table changes agents do not have the information necessary to make informed migration decisions. This is an undesirable situation as agents can become quite distant from the HL.

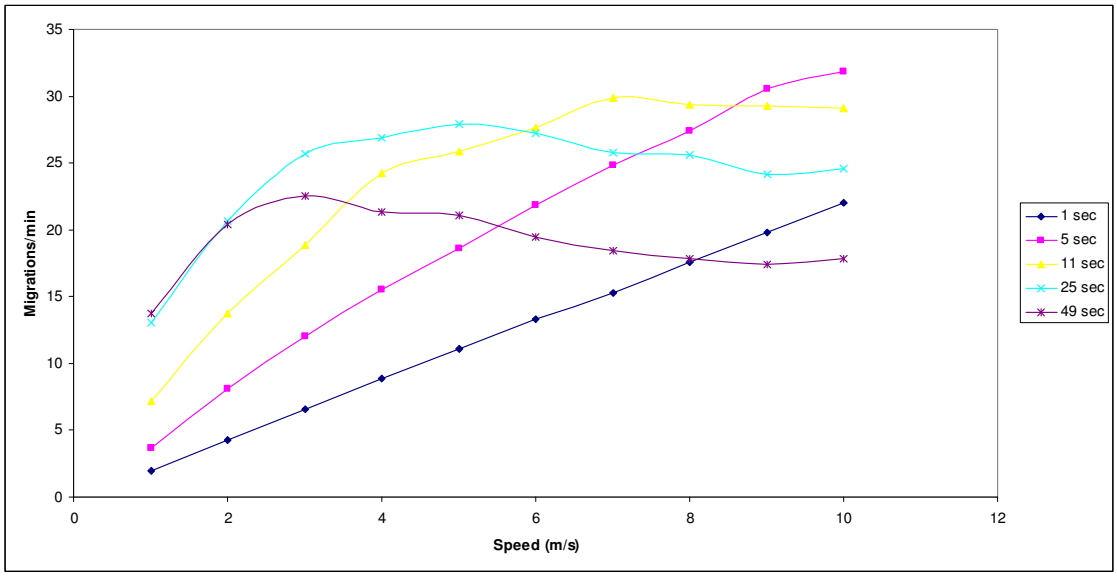

Figure 20: Measurements of migrations per minute

Overall, the simulation has shown that if low beacon rates are used, such as the one second, then the agents are able to remain with a transmission range of the HL. In addition, the overhead is sufficiently low to be negligible because the agent sizes are small. Agents consist of the location information, the node identifier and the program code; however, as all agents have the same code, this can be removed from the agent and placed on the node to reduce overhead associated migrations by only transmitting the agent's state.

4.3 Replication

When one considers that a node containing an agent may fail at any time then that agent would be lost. If a node's GLA only consists of one agent, then the location server is lost and it is not possible to discover that node's location. This loss can be reduced through

replicating the information to a quorum of nodes; however, in traditional quorum systems, there is usually a co-ordinating server (a master) and several slaves ready to take over should the master fail and this structure requires maintenance communication. Unfortunately, ad-hoc networks are highly dynamic and attempts to create traditional quorum systems for such networks fail (Luo et al., 2004).

In an ad-hoc network, a node failure will often be a result of battery failure or a node crash, and so the node may not come back online at all, or if it does then it may be in a different location (such as someone's phone battery becoming exhausted, and them returning to their house to recharge it sometime later). Therefore, any system needs to be highly dynamic and adaptable to mobile and failing nodes. The last section examined how one can handle mobility, by agents repeatedly migrating to maintain themselves close to a geographical point. This section will examine how to clone agents and maintain them as close to the point, and how to recover when an agent is lost (Owen and Adda, 2006b).

4.3.1 Mobility with multiple agents

In the previous section, agents migrated independently of one another and it was perfectly possible for two agents, of the same GLA, to reside on the same node. If this becomes the case, then despite having several replica agents, if they all reside on the same node and that node fails then the GLA as a whole will fail. Therefore, before making a migration decision, an agent needs to consider whether this will place it on the same node as another agent of the same GLA. At present, the only way an agent can do this is to initiate communication with each neighbour which would consist of at least two packets (request and reply). Earlier, the thesis described the idea of ants who modify the environment and stated that the idea would be applied to this system. In an ad hoc network, what nodes see are their neighbours' beacon packets and to them this is gives an indication of the environment. To avoid the need for agents to communicate with their host node's neighbours directly, the agents modify their environment, or in this case, the beacon packet to provide information to fellow agents.

The beacon packet is modified so as to give information about the agents residing on a particular node and which GLA they belong to so that the agents on neighbouring nodes can decide whether it is a suitable migration target. All that is required is for the addition to the beacon packet of a list of the agents' IDs, and then agents on neighbouring nodes can simply inspect their node's beacon cache to determine where fellow agents are. The beacon packet now looks as follows, with $<n_s(t)>$ representing the list of agent IDs on a node n at time t.

$$B_n(t) = \left(n_{ID}, n_x(t), n_y(t), < n_S(t) >\right) \tag{4}$$

In practice, the list size will depend on the number of agents on that node, and the size of the node identifier which will be discussed later. Methods of calculating the expected number of agents on a node and subsequently the size of the list will be examined later in this chapter.

Given this new information, the MigrateCheck() function must now be modified so that agents remove nodes which have agents of the same GLA from the migration eligibility list. An extra condition is added to the conditional statement requiring the agent to examine the node's location cache for agents of the GLA.

```
MIGRATECHECK()
1   dist ← ∞
2   closest ← none
3   For each n in neighbours
4       if DIST(n, HL) < DIST(self, HL) and
5          a.id is not element of <nS> then
6             dist ← DIST(n, HL)
7             closest = n
8   if dist != ∞ then
9       MIGRATETO(closest)
10      return true
```

```
11    else return false
```

Figure 21: Check whether a migration is needed.

The set n_S is the node n's beacon cache. The effect of this is that an agent will never migrate to a node that already has an agent from the same GLA. Given a group of agents then they will all reside on nodes around the HL, residing on the closest set to it.

4.3.2 Agent Cloning Techniques

Having established techniques to allow multiple agents to migrate without two agents from the same GLA migrating to the same node, it is now important to discuss how the agents will clone themselves. So far, agents do not communicate directly and rely on observation of their environment to move around. Keeping with this philosophy inspired by nature, duplication should be performed in the same manner. Typically in quorum systems, a central agent would control and monitor copies through regular communication. The beacon packet contains information about the number of agents on neighbouring nodes and so this can be used to determine the need to clone.

The GLA is created by the formation of one agent containing the information to be stored. This agent then migrates towards the HL, where upon arriving, the GLA is set up by the closest agent cloning itself to other nodes to form several copies.

Overall, the GLA system must perform three main actions to counter mobility and node failure:

1. **Migrate**: Agents must periodically examine the neighbours of the node it resides on, and if there is a node who is closer to the geographical point, then it must migrate there assuming a agent is not already residing there.
2. **Replicate**: Agents must replicate themselves to other nodes, so that node failures do not result in loss of the data.

Any agent making a replication decision must have a good idea on the number of other agents so that the number of agents can be kept optimal. Information is available by

inspecting the beacon packets of neighbours, so therefore the agent that makes replication decisions must ideally be the one that can see all, or most, of the beacons of nodes holding agents for that GLA, so that it may count them. This is likely to be the agent that is closest to the HL, as any clone agents are likely to be on adjacent nodes. Therefore, if an agent discovers that it is on a node that has no neighbours closer to the HL than it, then it shall elect itself as the *master* of the GLA and undertake the additional task of replication. Likewise, if an agent discovers that neighbours are closer to the HL and have agents of the GLA, then it shall assume the role of *slave* and no longer participate in the replication task. Figure 22 illustrates two GLAs with different home locations indicated by different colours. The master (double circled) is on the closest node to the home location, and the slaves (single circled) are on neighbouring nodes.

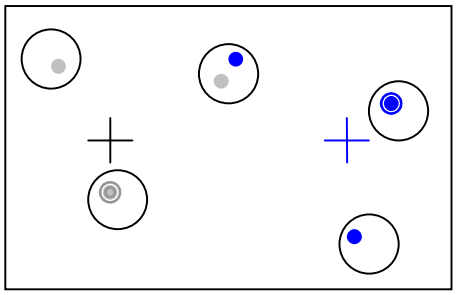

Figure 22: Two servers with different home locations.

If a two or more nodes hosting agents are of equal distance from the HL then the agent on the node with the highest ID becomes the master. If the master is unable to see a complete picture of the GLA because some agents are on nodes out of range, then it may incorrectly count the number of agents. If this happens then the master may clone itself unnecessarily.

For example, consider Figure 23 where one agent is on a node that is out of communication range of the node with the master agent. The master agent may count the number of slaves as two because that is all it can see, and if two is below the cloning threshold, it will duplicate even though this is unnecessary as the distant agent will quickly migrate to a closer node. This now leaves the scenario where we may have more

agents that is necessary and so pruning becomes essential, especially if this situation happens frequently because the number of agents can grow rapidly.

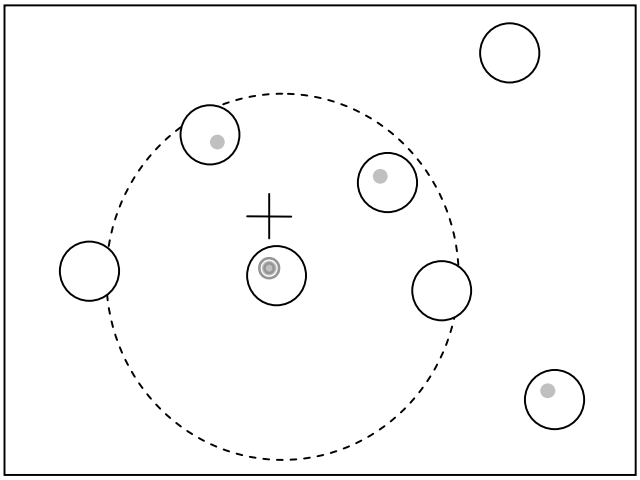

Figure 23: Agent out of range of master

Therefore, every agent performs *pruning* when it detects that there are more agents in the GLA than the cloning threshold. They will test this condition when they detect new beacon information on the hosting node, and if true they will delete themselves. In addition, this action will only be performed with a certain probability, as several agents could detect this condition at the same time and all delete themselves. The manner in which this probability is calculated is described later.

To allow replication a little more information is needed to facilitate co-operation. An extra field is added to each agent's data payload so that it can determine its role in the GLA. This field is a one bit Boolean field indicating whether the agent is the master or not, and is used solely by the agent so that it may determine if it needs to perform replication functions.

$$a_{ID} = \left(n_{ID}, n_x(t), n_y(t), role(t)\right) \tag{5}$$

Each agent of the same GLA represents the same node and the only field that may differ between agents is that of the role. The master will have a different value for the role field than the slaves.

Given the current information, agents can now be described as having four actions that they may perform (Owen and Adda, 2006c, Owen and Adda, 2006d):

1. **Election:** Each agent tests to see if the current node is the closest to the geographical point, and if so changes its role to master. Otherwise, if there are closer nodes holding agents, the role changes to slave.
2. **Migration:** Each agent checks to see if it is on the closest node to the HL, and if there is a closer node not hosting an agent of the same GLA it will migrate to it.
3. **Replication (master only):** The master periodically examines the beacon packets of neighbouring node and counts the number of agents; if this number has fallen below a threshold a replication function is executed.
4. **Pruning (slaves only):** Each agent examines the beacon packets of neighbouring nodes and if the number of agents is above a threshold, then it deletes itself with a certain probability *P*.

P is chosen so that approximately the correct proportion of agents is deleted. This can be calculated as follows, given the desirable number of agents *threshold*, and the counted number *count*:

$$P = 1 - \frac{threshold}{count} \tag{6}$$

The final function that agents run every time the node's beacon cache changes is described in MANAGEAGENT() function. This function is executed when a node receives new information about the environment, such as new beacons being received and beacons timing out. The object *neighbours* is an array containing all the beacons received from the node's neighbours and *P* is the probability of an agent elected for pruning being deleted.

MANAGEAGENT()

```
1   count ← 0
2   role ← master
3   for each n in neighbours
4       if self.id is element of <n_S> then
5           count ← count + 1
6           If DIST(n, HL) < DIST(self, HL) then
7               role ← slave
11  if MIGRATECHECK() then return
12  if role = master and count < threshold then
13      REPLICATIONFUNCTION(self, count)
14  else if role = slave and count > threshold and RAND() < P then
15      DELETESELF()
```

Figure 24: MANAGEAGENT(*a*) function.

The REPLICATEFUNCTION() is an arbitrary method of replication. The next subchapter will describe several methods of replication and compare their performance.

4.3.3 Replication methods

Several methods for performing replication are examined. Evidently performing accurate replication and pruning in a highly dynamic environment is not possible so several techniques are examined to find the one providing optimal performance.

1. **Unicast:** When the master discovers that there is less than the threshold number of agents then the function unicasts a new clone to a neighbour that does not already have one. If the count falls below the threshold by more than one, then the replication is performed again at the next invocation of the function (when beacon cache changes). This allows the process of replication to be performed slowly avoiding most excessive replication. The function below simply finds the first neighbour without an agent and then sends a clone there. The CLONESELFTO() function clones the agent and sends the clone to the node specified.

```
UNICAST()
1   for n in neighbours
2       if n.id is not element of <n_S> then
3           CLONESELFTO(n)
4           break
```

Figure 25: Unicast replication pseudo code

2. **Broadcast:** Unicast recovers from more than one node failure slowly, so if several nodes fail then it will take several function invocations to restore the server to previous strength. An alternative to this is to broadcast new copies to all neighbours in the master's vicinity so that a loss is quickly recovered from. This may bring the number of agent significantly above the threshold, and so relies on the pruning to reduce the GLA size to the correct number.

```
BROADCAST()
1   CLONESELFTO(all neighbours)
```

Figure 26: Broadcast replication pseudo code

3. **Hybrid:** Both of the above methods have advantages and disadvantages, and the hybrid approach attempts to gain the best of both. If the number of agents is less than 50% of the threshold, then the broadcast technique is used to quickly restore the GLA to good strength. If the number of agents is above or equal to 50% of the threshold, then the unicast approach is used to more slowly restore the GLA.

```
HYBRID(count)
1   if count ≥ ½ threshold then UNICAST()
2   else BROADCAST()
```

Figure 27: Hybrid replication pseudocode

4. **Multicast:** The multicast technique is an attempt to restore the GLA quickly without taking the number of agents over the threshold. Upon sensing a lower

than threshold number of agents the master calculates the exact number of agents needed. It then examines the beacon cache to find the correct number of neighbours needed that are not hosting agents of the same GLA. A multicast cloning packet is then sent with these neighbours' addresses thereby restoring the server to full strength without exceeding the threshold. If the number of vacant neighbours is less than the number of agents needed, then clones will be sent to all neighbours that do not have one. The function MULTICASTCLONETO() clones the agent and sends it to the agents listed in the parameter.

```
MULTICAST(count)
1   <multicastList> ← ()
2   for each n in neighbours
3       if n.id is not element of <nS> then
4           <multiCastList>.ADD(n.id)
5           if |multicastList| > (count – threshold) then break
6   if |multicastList|> 0 then
7       MULTICASTCLONETO(multicastList)
```

Figure 28: Multicast replication pseudo code

4.3.4 Simulation results

A simulation is set up to evaluate the four methods proposed above. A total of 50 GLAs are deployed at random points in the area and the *threshold* for duplication is set at five for the initial experiments and analysed later to determine a suitable value. Three characteristics are measured: failure tolerance, overhead, and node occupancy. Overhead is the average number of packets sent per minute by each server and node occupancy is the average number of agents a node holds. Perhaps most importantly of these though is failure tolerance, which is the likelihood of a GLA surviving in the face of failures of nodes. This is most important because if the GLA is lost then no location information can be discovered for the node which owned it. Therefore, the simulation is run for 30 minutes so that long term failure tolerance can be examined.

Node failures are simulated by requiring every node to draw, every five seconds, a random number between [0,1], and if this number is smaller than a certain number (p), then the node will fail. One can therefore calculate the number of node failures per minute (f) given the number of nodes (n), as:

$$f = \frac{60}{5} pn = 12pn \tag{7}$$

Figure 29 shows the failure tolerance of the four approaches. The failure tolerance is measured by counting the number of GLAs which are lost over the 30 minute simulation. The broadcast approach has the greatest tolerance but this is because it often has a larger number than the present threshold of slaves due to the inability to precisely control the duplication process. The unicast approach had the least tolerance due to its inability to recover quickly from a number of node failures. Hybrid and multicast performed between the unicast and broadcast approaches because both more accurately maintain the current number of agents, rather than too few or too many. In spite of this, all the techniques survive exceptionally well. When every node in the network fails at least once a minute 20% of the servers are able to survive for thirty minutes with the unicast approach, whilst 80% survive with broadcast. Also worth noting, is that the GLA is likely to be updated more often than every 30 minutes and so this update process could recover from total loss in exceptionally harsh conditions. In real scenarios failure rates this high are not expected because mobile devices often have hours or days of battery life. In addition, their owners regularly anticipate battery failure and recharge them before this stage is reached.

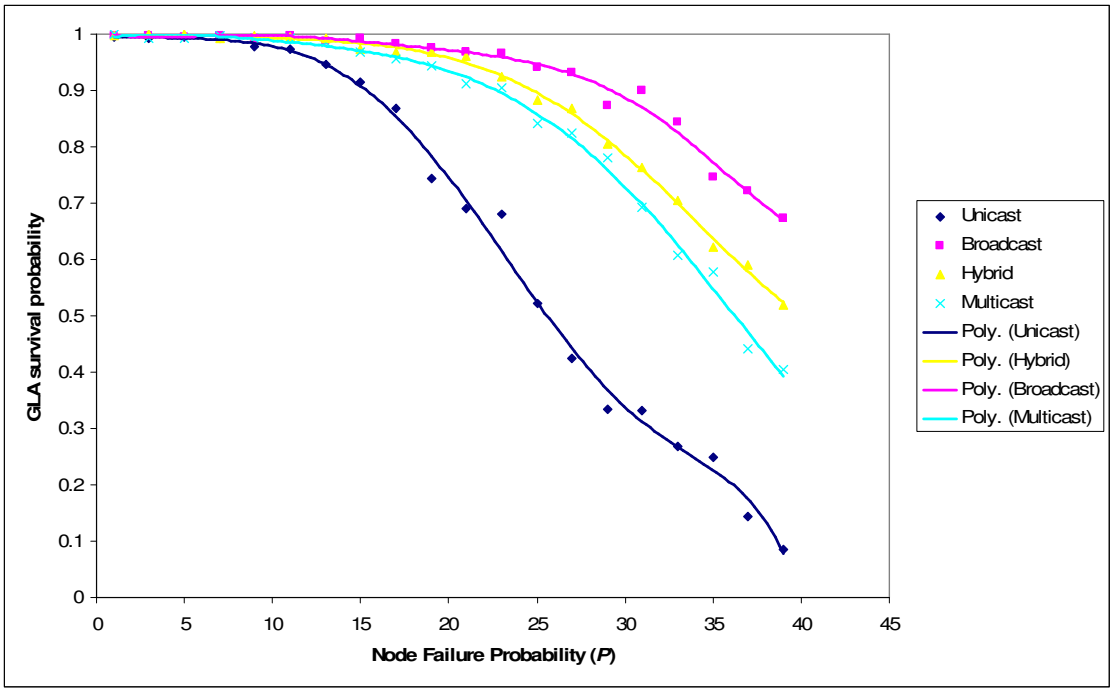

Figure 29: Failure tolerance

Figure 30 shows the number of packets incurred per minute for each of the agent replication methods. The poorest performing was the unicast approach due to its single packet single duplicate approach, whereas the hybrid performed the best. This is unusual at first glance as one would expect the multicast to outperform the rest as it performs duplication accurately by specifying the exact numbers of clones to create to reach the threshold. However, on further analysis it was discovered that the hybrid performs better because it broadcasts when the number of agents falls too low which quickly recovers and this rarely requires further cloning to reach the threshold. Pruning then takes care of the excess number gradually. But, given this line of thinking one would expect the broadcast approach to outperform as it always uses broadcast packets. This is not the case because broadcasts are used even if the count falls below the threshold by only one which results in a much larger number of clones on average. Pruning eventually takes care of this but undesirable behaviour is observed that is described in section 4.3.5.

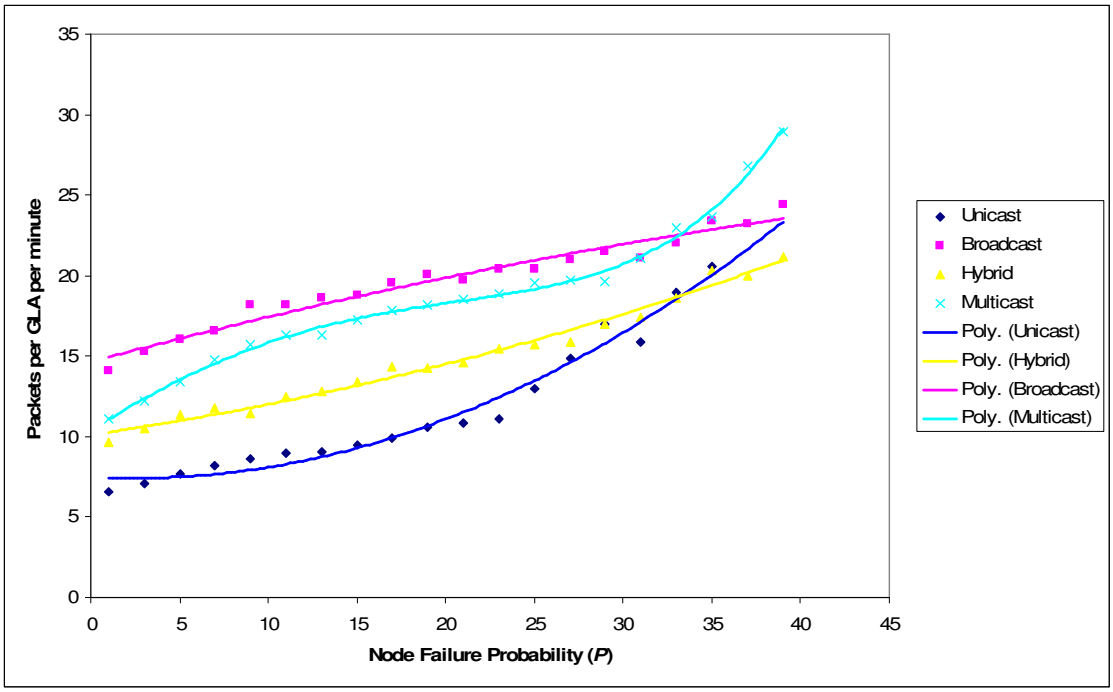

Figure 30: Packets per GLA per minute

Figure 31 shows the average number of agents that a node would hold when there are 50 GLAs in the network. The expected number of agents per node assuming the algorithm maintains the exact numbers of agents per GLA is qc/n; where n is the number of nodes, q is the number of GLAs and c is the number of agents per GLA.

The predicted number for the scenario simulated is 1.5 (q=50, c=6, n=200) agents per node which we see accurately predicts the number for the unicast approach. The multicast is not significantly different from the unicast because it too attempts to accurately maintain the correct number of agents; however, the hybrid and broadcast have slightly large values because they are both less able to control the number of agents due to the use of broadcast.

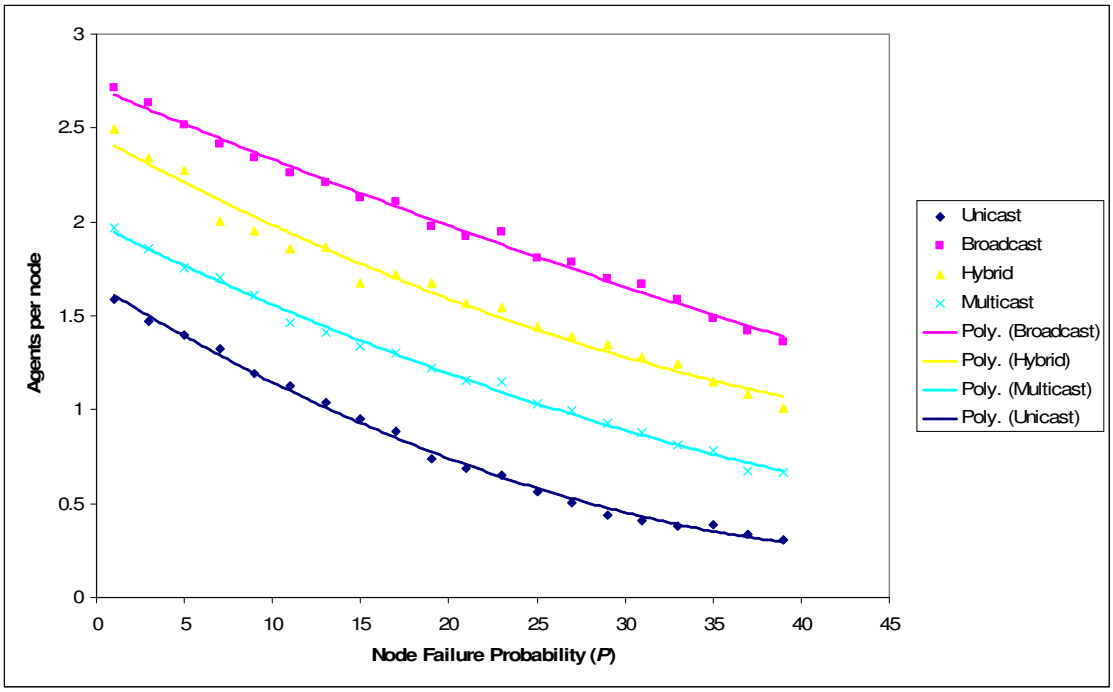

Figure 31: Average number of agents on each node

All the above scenarios use the arbitrarily chosen value of five for the number of duplicate agents (threshold). But it is important to examine the best choice for the threshold value and this is undertaken here. The simulation is set up the same as above with the exception that the threshold value is also varied along with the node failure rate. The threshold maximum will be the number of neighbours the master has because it is unable to duplicate if there are no nodes without any agents remaining.

Provided nodes are uniformly distributed, one can calculate the expected number of neighbours (d) using the following equation, given the number of nodes (n), the transmission radius (r), and the total area of the simulation (a^2):

$$d = \frac{n\pi r^2}{a^2} - 1 \qquad (8)$$

In the case simulated here, the expected number of neighbours is about 5 ($n=200$, $r=100m$, $a^2=1000^2$), and so one would expect the improvements caused by increasing the *threshold* would begin to taper off around this value.

Only the hybrid approach is examined here as this is adopted throughout the rest of the thesis due to its performance in the previous set of results over the alternatives in terms of overhead and failure tolerance.

Figure 32 illustrates the varying threshold has on the performance of the system. As one can see, when setting the value to around 5-7 duplicates the performance increase is not very significant as the master has fewer chances to increase the server population.

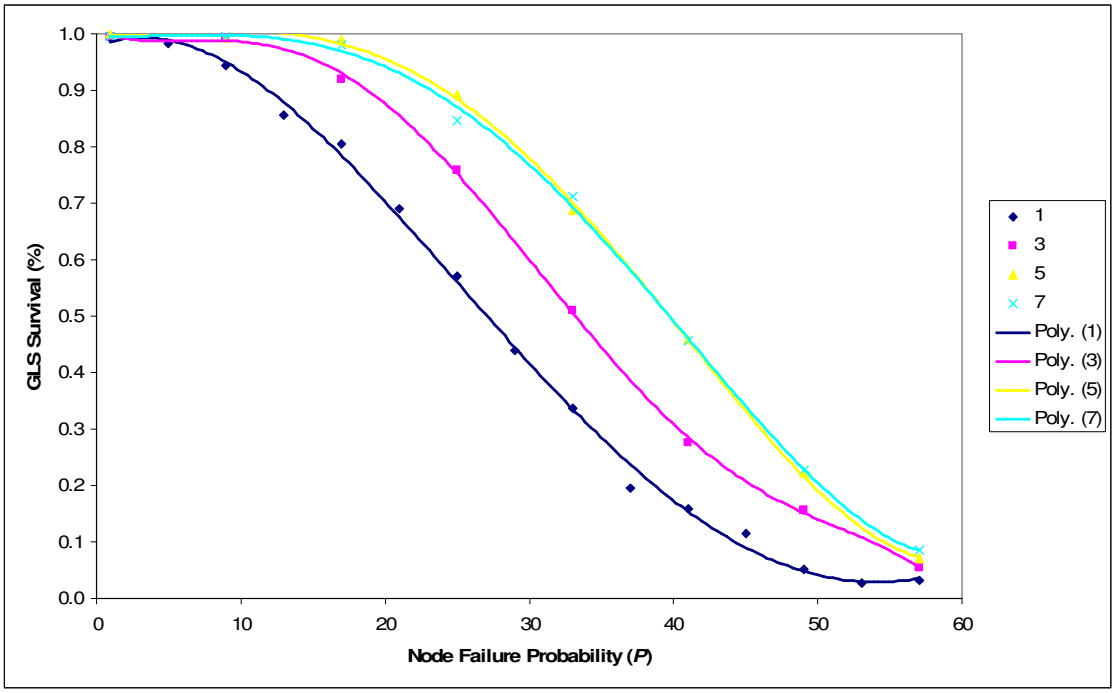

Figure 32: Failure tolerance of the hybrid approach with various threshold values

Figure 33 illustrates the overhead measured over the simulation of the fifty GLAs in total packets sent. The value chosen for number agents per GLA is chosen in increments of two to illustrate clearly the difference in performance. The setting of this value is crucial

in minimising overhead as larger numbers increased overhead significantly, with seven having almost twice the overhead of one.

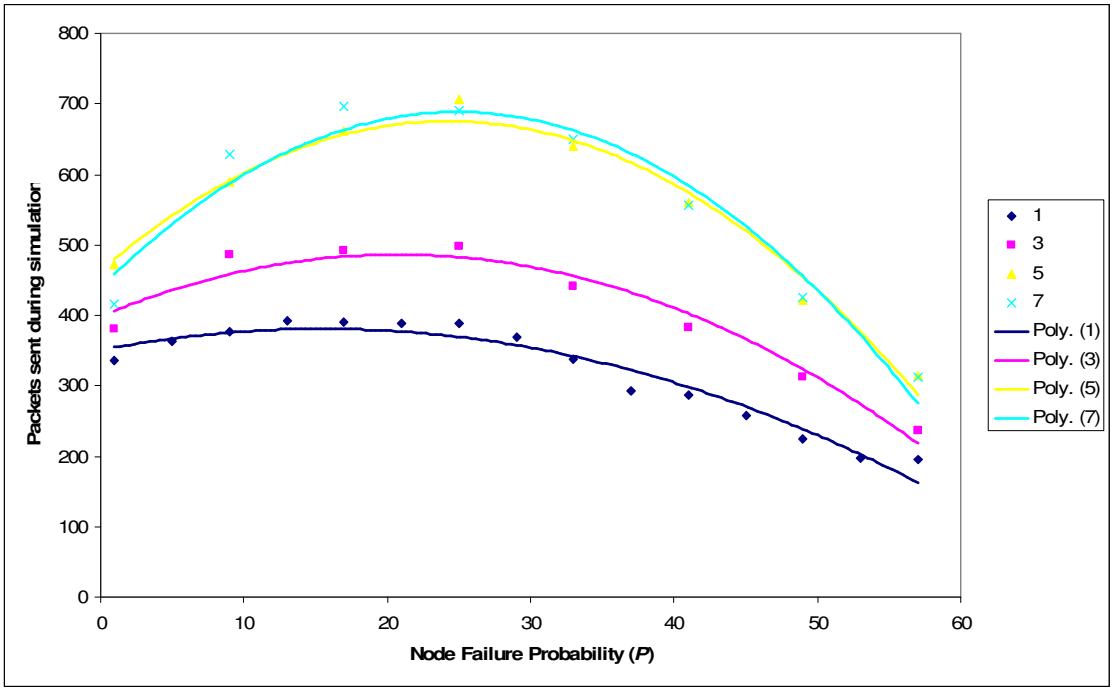

Figure 33: Overhead of the hybrid approach with varying threshold

Given the above results a threshold value of five provides the best survival as expected. The performance varies significantly with the threshold setting and so it is important it is chosen correctly, and so this will be examined in section 4.4.

4.3.5 Broadcast replication: The expand-and-collapse problem.

During simulations of the broadcast technique a phenomenon that is termed the *Broadcast Expand and Collapse Problem* (BECP) is noticed. If the GLA size drops one below the threshold a broadcast replication is initiated which may take the number of agents significantly above the threshold. When pruning then takes effect, the GLA size quickly decreases, but inevitably one or more agents that do not have an accurate beacon cache also delete themselves. This brings the GLA size below the threshold which then results in another broadcast replication. The result is that the GLA repeatedly broadcasts and prunes and this phenomenon was observed repeatedly throughout simulations. The hybrid approach solves this problem by the selective use of broadcast and unicast.

4.4 Adaptive Threshold Control

4.4.1 Introduction

Evidently one needs to customise the cloning threshold based upon the failure rate of the nodes and the average number of neighbours in the network. The average number of neighbours can easily be calculated to indicate the maximum value for the threshold. The node failure rate is more difficult to calculate in reality although it is easily set and observed in simulation. Nodes can only observe the neighbours around them to determine what this rate may be and its primary source of knowledge is timeouts of beacon information. Unfortunately, this beacon information regularly times out because of node mobility in addition to node failures and so a scheme is needed to determine between the two.

Throughout this thesis it has been assumed that node failures are a result of battery failure that could not be predicted. The reasoning for this is that it simulates the worst case scenario by trying to recover from unpredictable failures. Many mobile devices measure the voltage output from their battery and use this successfully to predict failure. As a result, before the device reaches a critical level, it could send out a message indicating failures to its neighbours which would not only allow the algorithm to move agents to more suitable nodes, but also to determine the node failure rate. However, this is an ideal case and cannot be guaranteed. For example, the user could simply unplug his device's battery or it could crash.

4.4.2 Determining the likelihood of failure

This section will examine the possibility of determining failure probability a node in the network, assuming all nodes have the same probability of failure. Each node beacons every second and keeps a record of the set of nodes it receives beacons from (N_t). Each node also retains the record of nodes whose beacons it received one second ago (N_{t-1}) so that it can determine the nodes which are no longer beaconing or in range. D_t contains all the nodes whose beacons were not received within the next one second time slot and so they can be assumed to have either failed or moved out of range.

$$D_t = N_{t-1} - (N_{t-1} \cap N_t) \tag{9}$$

The list of nodes for which we are no longer receiving beacon packets for contains both those that have failed and those that have moved out of range. Assuming a uniform transmission range (R), we can calculate if a node (d) has moved out of range of node n if it broadcasts its speed (d_s) and direction of travel (d_d) along with its location (d_x, d_y).

$$\sqrt{(d_x + d_s \sin d_d - n_x)^2 + (d_y + d_s \cos d_d - n_y)^2} < R \tag{10}$$

The set $F_t \subset D_t$ contains those nodes from D_t that satisfy the above condition. Therefore, all nodes in F_t can be presumed to have failed.

The failure rate is calculated as an average of all observed failures since the node has been active. Calculating based upon one observation is not accurate as with high beacon rates no neighbours may fail in that period. Therefore, a node calculates the average observed failure rate (n_f) as shown below:

$$n_f = \frac{1}{t} \sum_{i=0}^{t} \frac{|F_{i+1}|}{|N_i|} \tag{11}$$

To examine whether this approach is capable of detecting the failure rate accurately, several simulations are run with varying failure rates and the analyses of the nodes are recorded after five minutes. All 200 nodes may move at a random speed between 0.1m/s and 10m/s and the transmission radius is fixed at 100m.

The figures below present the observed failure rates as a percentage of nodes which held that belief. Figure 34 illustrates the scenario with a failure rate of 1% of nodes failing every five seconds. The observations are normally distributed with $\mu = 0.99\%, \sigma = 0.2$, showing the mean observation to be almost exactly correct.

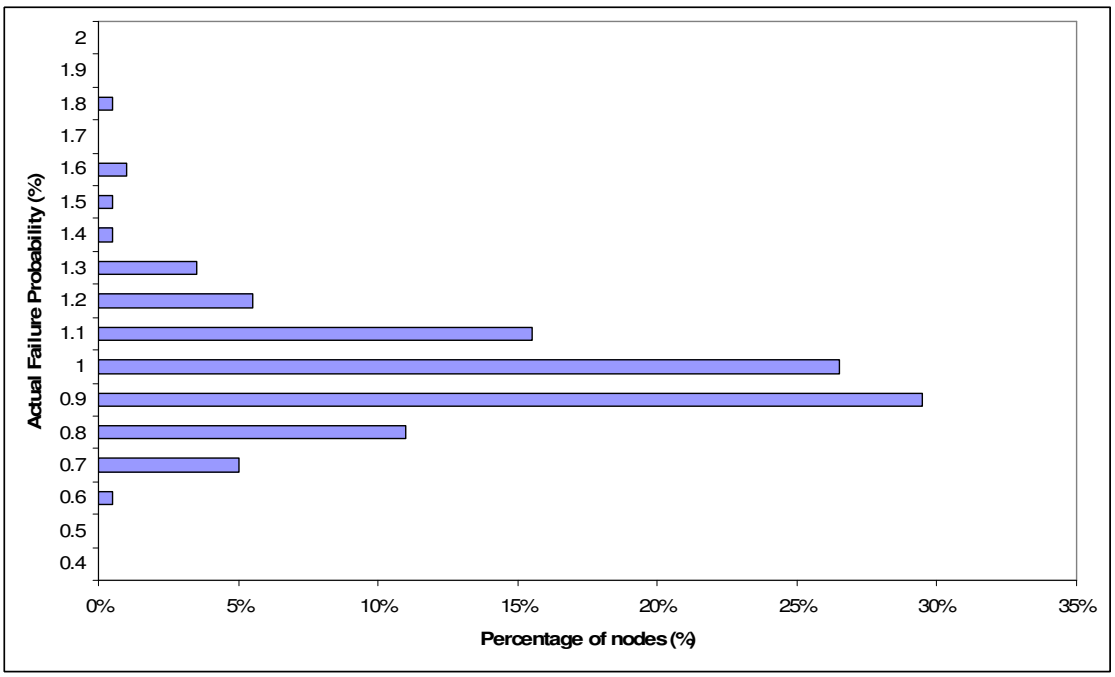

Figure 34: Failure rate analysis with rate = 1%

Figure 35 illustrates the nodes' analysis when the failure rate is 0.1, and the analyses are normally distributed with $\mu = 9.3\%, \sigma = 0.35$.

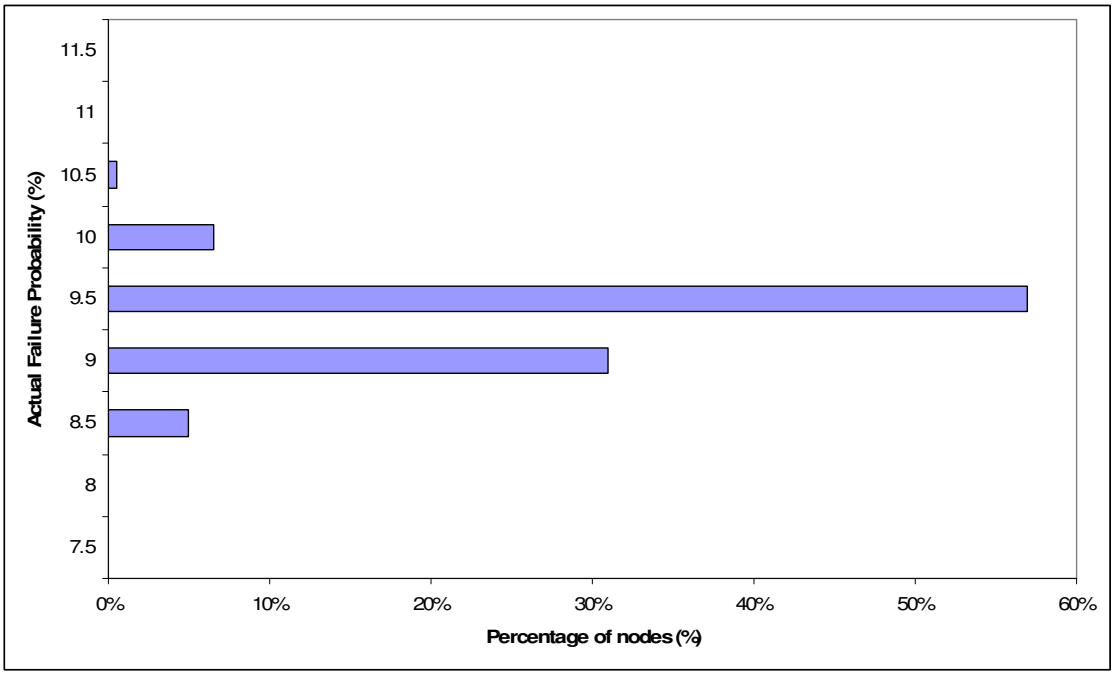

Figure 35: Failure rate analysis with rate = 10%

Figure 36 illustrates the nodes' analysis with a failure rate of 0.2, which is also normally distributed with $\mu = 18.4\%, \sigma = 0.5$.

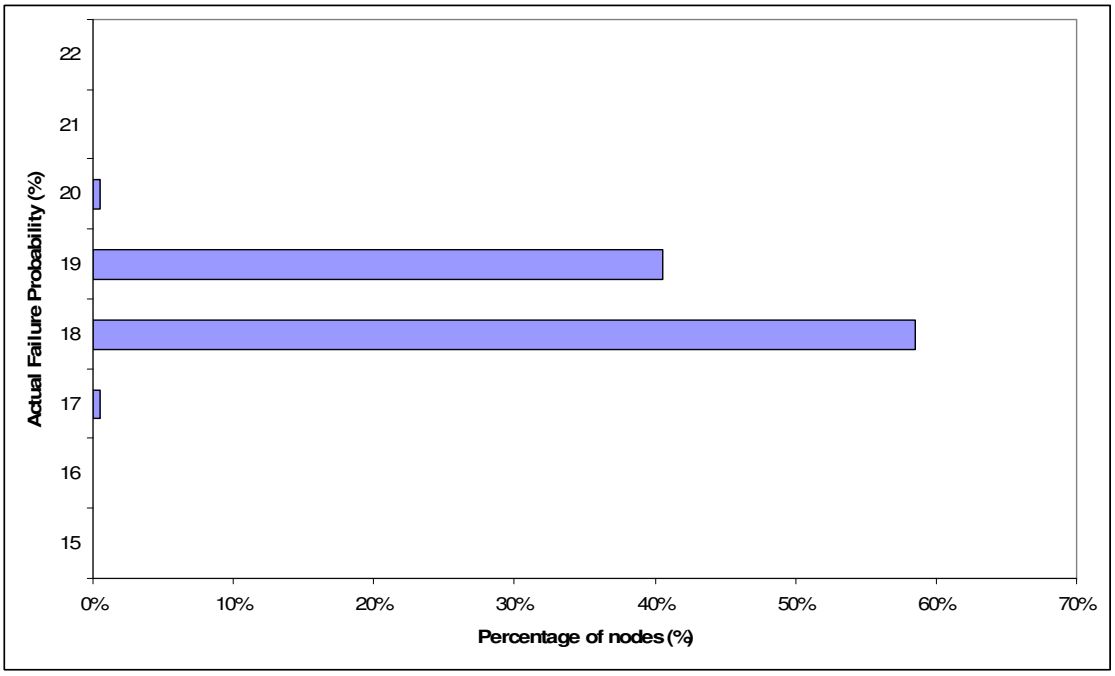

Figure 36: Failure rate analysis with rate = 20%

The results from the simulation show a slight underestimation in the failure rate which increases with the failures. The error (c) can be calculated given the actual failure rate (f) and the node's observations (n_f) where || denotes a positive value of the equation enclosed:

$$c = \frac{|f - n_f|}{f} \tag{12}$$

While overestimating the failure rate would only incur extra overhead and increase system failure tolerance, an underestimation may result in the loss of the system due to insufficient slaves being created. The errors observed in simulation using the above equation are:

Table 4: Predictions of failure rates by nodes

f	n_f	Error percentage (c)
1%	0.99%	1%
10%	9.33%	6.7%
20%	18.41%	8%

A slight underestimation in the failure rate has been observed and this is easily catered for by adjusting for the error.

4.4.3 Implementing Adaptive Threshold Control

This section details methods of adjusting the threshold based on observation of the failure rate. Evidently, one must have at least one duplicate of the agent to ensure survivability; even when observing a zero failure rate, it is recommended to have at least one in case of an unexpected failure. Therefore, the threshold value, *h,* will be in the range $2 \leq h \leq d$ where d is the number of neighbours of the master's node. As stated earlier, with one-hop duplication it is not possible to duplicate to more than d neighbours. This is a potential improvement to the scheme in exceptionally high failure networks but does restrict the monitoring the master can do of all the agents and may require a higher overhead and central co-ordination. In any case, failure rates of this level are not expected in the applications for which this system is proposed.

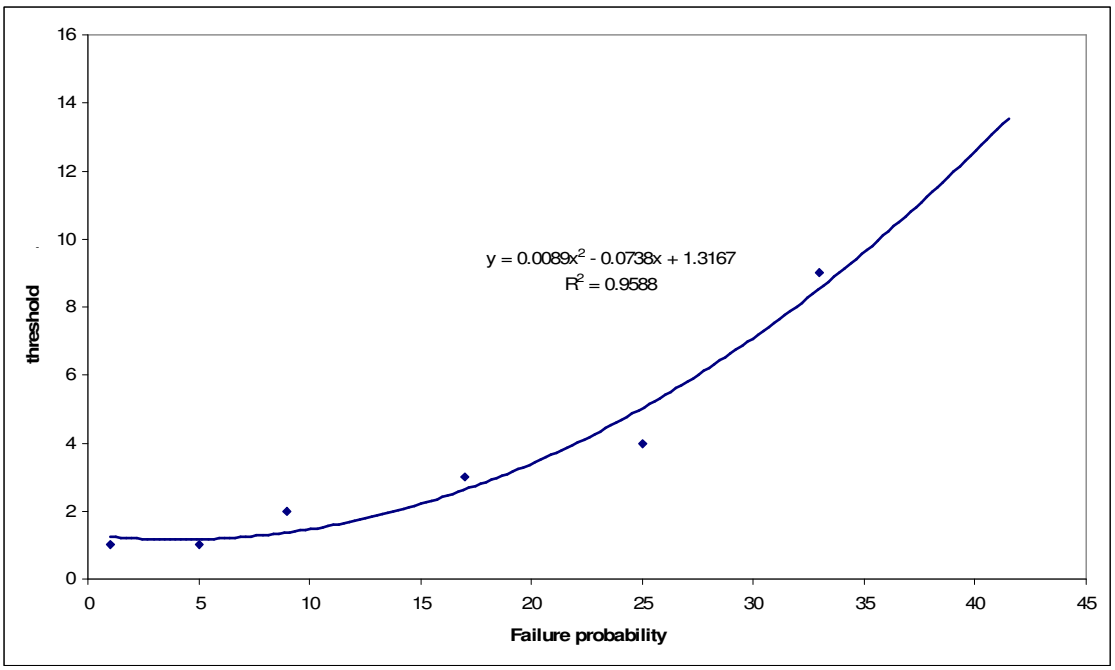

Figure 37: Threshold that maximises survivability for various failure rates

Figure 37 shows the minimum threshold value that maximises survival. Examining data from the experiments of how threshold affects failure, one can plot the minimum threshold required to provide maximum tolerance to failures. Therefore, the data is plotted and fitted to a standard polynomial equation ($y = Ax^2 + Bx + C$) which fits with a value of R^2=0.95. This equation can be used in simulation along with failure rate prediction to determine a threshold value so that overhead is minimised but survival is maximised:

$$threshold = 0.0089 f^2 - 0.0738 f + 1.3167 \quad (13)$$

Figure 38 shows the pseudo code used in simulation to determine the *threshold*. The threshold value is restricted in the limits [1, NEIGHBOURCOUNT()] for two reasons. Firstly, using a threshold of zero will provide no slaves and hence no failure tolerability should the unexpected happen. Secondly, the threshold is capped at the number of neighbours because the master is unable to create any more slaves than this, and repeatedly attempting to will increase the overheads unnecessarily.

Each node stores two global variables, *Failure* and *FailureCount* which are used to determine the failure rate. At each beacon interval, the algorithm examines the current list of neighbours, and that from the last invocation of the function to determine those that are no longer neighbours (Lines 3-4). Of those nodes, each is tested to see if its expected location (*n.expectedLoc*) is outside the limits of communication range (Line 5), and if so it is assumed to have failed. The percentage failed is added to the number of neighbours and is added to *Failure*, while *FailureCount* is incremented by one (Lines 6-7). This provides two global variables that are updated frequently with the latest information on failures to provide a value f that converges on to the true failure rate. Using the previously mentioned equation and limits, the *threshold* can then be determined (Lines 8-12).

```
DETERMINETHRESHOLD()
1   define globalVars Failure, FailureCount
2   lost ← 0
3   foreach n in oldNeighbours
4       if n not in neighbours and
5           DIST(thisNode, n.expectedLoc) < radioRange then lost++
6   Failure ← Failure + lost / size(oldNeighbours)
7   FailureCount++
8   f ← Failure/FailureCount
9   threshold ← 0.0089 f^2 − 0.0738 f + 1.3167
10  if threshold < 1 then threshold ← 1
11  else if threshold > NEIGHBOURCOUNT()
12      then threshold ← NEIGHBOURCOUNT()
13  oldNeighbours ← neighbours
```

Figure 38: Pseudo code for threshold determination

To assess the effectiveness of this failure prediction and adaptive threshold the algorithm is simulated with the above additions. The simulator only varies the failure probability and all other variables remain static, with nodes moving between 1 and 10m/s.

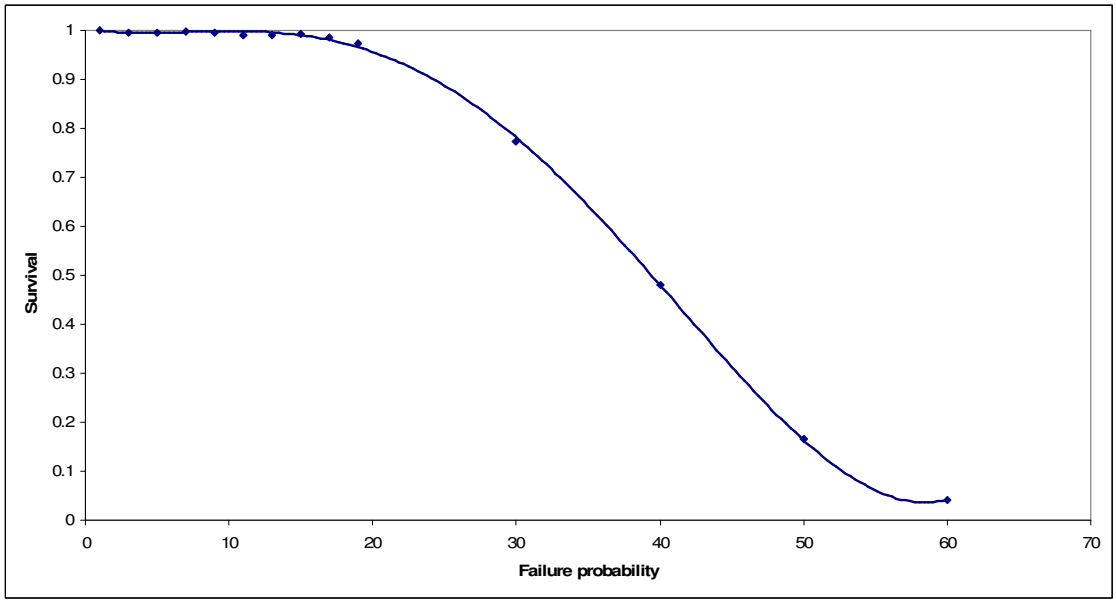

Figure 39: Survival with failure prediction and adaptive threshold

Figure 39 shows the results obtained from the simulation. One can see that the system is able to maintain near maximum survivability up until failure rates around 20%. This is quite promising as failure rates that high are unexpected in real world networks and show that for these scenarios the system is able to adaptively maintain near total survival. For example, one can calculate the average battery life (b) in hours given the failure probability (f), where 5 is the interval between failures, using the following equation:

$$b = \frac{5}{3600 f} = \frac{1}{720 f} \tag{14}$$

Given the example of f=0.2 (20% failure probability) then the expected battery life of the device would be just 25 seconds and failure rates of f=0.01 (1%) would be a battery life of just 8 minutes 20 seconds.

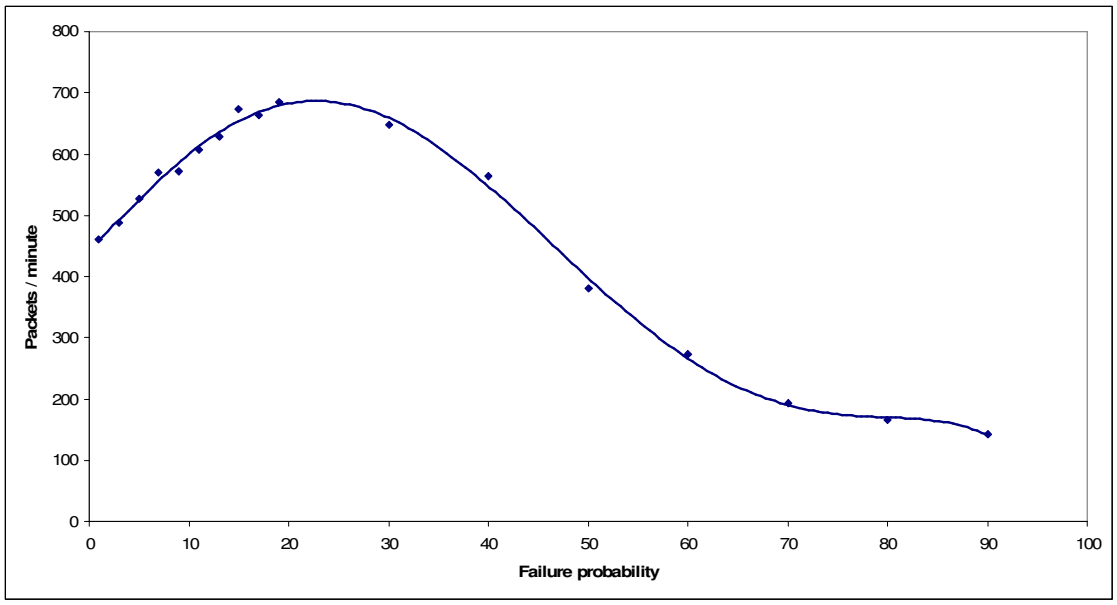

Figure 40: Packets per minute with adaptive threshold selection

Figure 40 illustrates the overhead of the system and that the adaptive threshold selection is working by the increase in overhead to compensate for increasing failure rates. The overhead peaks at failures of 20% where the maximum threshold is reached and then drops off as GLAs are lost.

This technique for determining the threshold makes the assumption that nodes have similar failure rates and examining situations where this is not the case is outside the scope of this thesis. The results are applicable to any size of network as long as the node density remains the same.

4.5 Conclusions

This chapter examined storing data using agents at nodes near the HL and found that with sufficient node density this was possible and incurred negligible overheads. A method of determining the number of agents necessary was explored so that overheads could be altered based upon the likelihood of node failures. It was found to be possible to accurately predict network node failure rates and to calibrate the number of duplicates of the data to maintain fault tolerance. Adaptive threshold selection is imperative in real networks where the failure probability is not known. Performing the task incurs memory

overhead of an extra neighbour table, one floating point number and an integer number. The network overhead is kept to an appropriate minimum and survival is maximised.

In conclusion, this chapter has shown that it is possible to store data at a static point in the face of node mobility and failures. In the next chapter, the task of querying and updating the GLA will be examined in terms of use as a location server.

5 SOLS as a location server for large ad hoc networks

5.1 Introduction

Ultimately this thesis has focussed on location servers in ad hoc networks as this is the most prominent problem for data storage in large scale networks; although this work could be used to store data for other reason, location servers is the primary focus of this chapter's analysis.

Figure 41 illustrates how a GLA will function in a large scale ad hoc network at serving location information for routing. Node S is wishing to communication with node D, and so S sends a query to D's home location, then an agent receiving the message replies with the information. This allows S to route packets to D using its last recorded location.

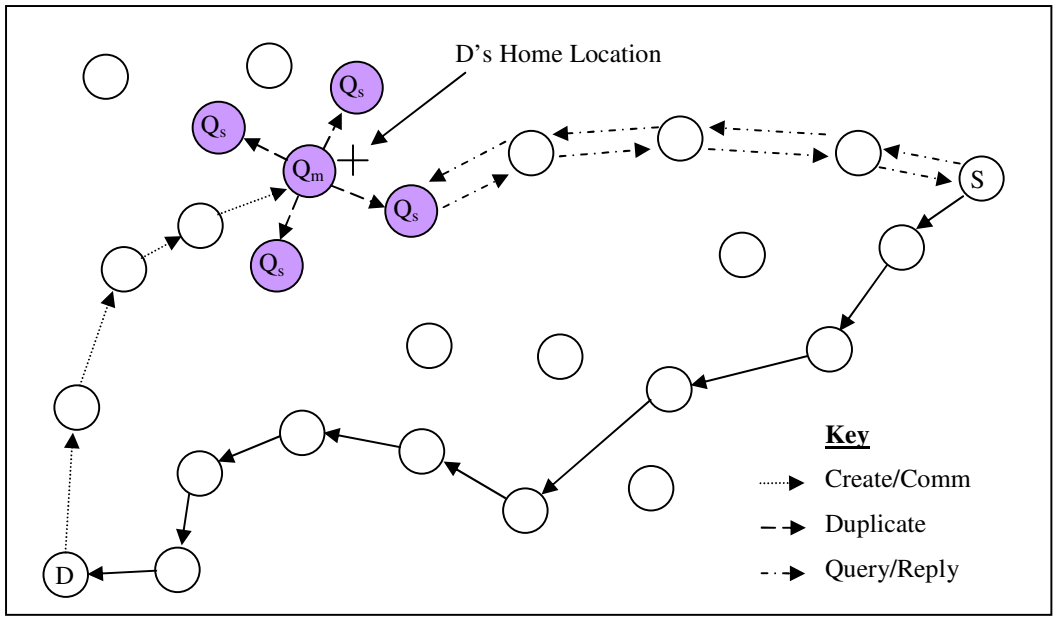

Figure 41: Illustration of server system

This chapter will examine how the SOLS scheme described in the last chapter will be implemented in a large scale ad hoc routing protocol. Unless otherwise stated perimeter routing is used when MFR fails so as to maximise delivery success.

SOLS is implemented exactly as described in the previous chapter. Specifically, this chapter sets out how the GLA is queried, updated, and incorporated into the routing protocol. The performance as a location server is compared with other location servers to examine the benefits. Finally, recommendations for its implementation into existing technologies are proposed.

5.2 Updating a GLA

5.2.1 Update technique

A node must periodically update its GLA so that nodes wishing to contact it have recent information on its whereabouts. To update the GLA a simplistic minimal knowledge approach is used that continues with the theme of avoiding central organisation. A node $n \in N$ wishing to update its GLA (Owen and Adda, 2007) creates an update packet $U_{ID}(t)$ which contains the node's ID (n_{ID}), and its current location (n_x, n_y) at time t. In addition, the time the update was created (t) is added to the packet to ensure that delayed updates do not rollback the GLA's information after a later update has been sent. So that the version difference can be determined, the agent also records the agent version (a_t).

$$U_{ID}(t) = \left(n_{ID}, n_x(t), n_y(t), t\right) \tag{15}$$

Once the update packet has been created then it must be sent to the GLA, which resides at the home location. The home location can be calculated from the hash-function $H(n_{ID})$ described in the literature review, or where the location is not defined by the function then extra fields can be included. The packet is sent using perimeter routing where necessary to the home location. Any node receiving the packet then executes the following function: RECVUPDATEPACKET($U_{ID}(t)$)**.**

```
RECVUPDATEPACKET(u)
1   a ← GETAGENT(u_ID)
2   if a is not null and u.t > a.t then
3       UPDATEAGENT(a, u)
```

```
4       BROADCAST(u)
5   else if u was unicasted to this node then
6       ROUTETOHL(u)
```

Figure 42: Pseudo code for update packet handling

Initially, the function checks to see whether the current node hosts an agent of the GLA which belongs to the source node. If an agent is present ($a_{ID} = u_{ID}$), and it has location information that is older than the update packet's, then it performs two actions: update the agent's information (UPDATEAGENT()) and broadcast the update packet to all neighbours. Otherwise, the packet is forwarded toward the home location by the ROUTETOHL() function. This update procedure results in the following behaviour:

1. Forward update to the home location until the closest node is reached.
2. Update any agents encountered on route, and broadcast update to their neighbours (who also execute the function).
3. Only nodes with agents that have old data will rebroadcast the packet. This results in a self limiting broadcast that does not travel beyond the GLA.

Broadcasting updates only when encountering an agent with an older version results in an optimal broadcast scheme. If one agent broadcasts the packet which updates all other agents, then no more broadcasts will be sent. In the worst case, the number of broadcast updates will be equal to the number of agents in the GLA.

This approach does not rely on central co-ordination to update the GLA and is entirely decentralised; however, it does not guarantee a complete update or indeed an update at all because of the following reasons:

- The GLA may have been carried away from the home location due to node mobility. But, because the agents migrate automatically this should be unlikely. If area around the home location is void of nodes, then the use of perimeter routing should mitigate this problem.

- The update may not reach all agents in the GLA if some agents are more than one hop away, such as in Figure 23. Upon an agent re-broadcasting the update, the broadcasts may not reach all members of the server due to node mobility; however, as agents do migrate this should be minimised.

Lack of update is unlikely as the GLA constantly attempts to remain around the home location. Even where there is a void present the GLA will sit around the perimeter where the use of perimeter routing will find it, as examined in section 5.4.

Incomplete updates could be mitigated by the application of mobility prediction or similar schemes, allow. In addition, the effect of out of date information has on delivery success is not overly significant because later results (section 5.2.3) demonstrated that small error in the position accuracy does not result in routing failure.

To determine the likeliness of incomplete updates the results from simulation measuring these is shown in the next two figures. Updates are most likely to be incomplete in low node densities where nodes travel out of range of the GLA. Therefore, the simulation varies radio range from 50m to 350m to simulate different node densities. Agents were updated every ten seconds in a 1000 second scenario and the simulator examined all GLAs' agents' data versions in between updates. The percentage of agents that had failed to receive an update is shown in Figure 43. The node density is calculated as the expected number of nodes (d) in a transmission radius (r) given the area (A^2) and the number of nodes N:

$$d = \frac{N\pi r^2}{A^2} \tag{16}$$

Figure 43 shows the percentage of agents of a GLA updated successfully each time, averaged over all GLAs. Node densities are calculated based upon radio ranges used earlier in the thesis.

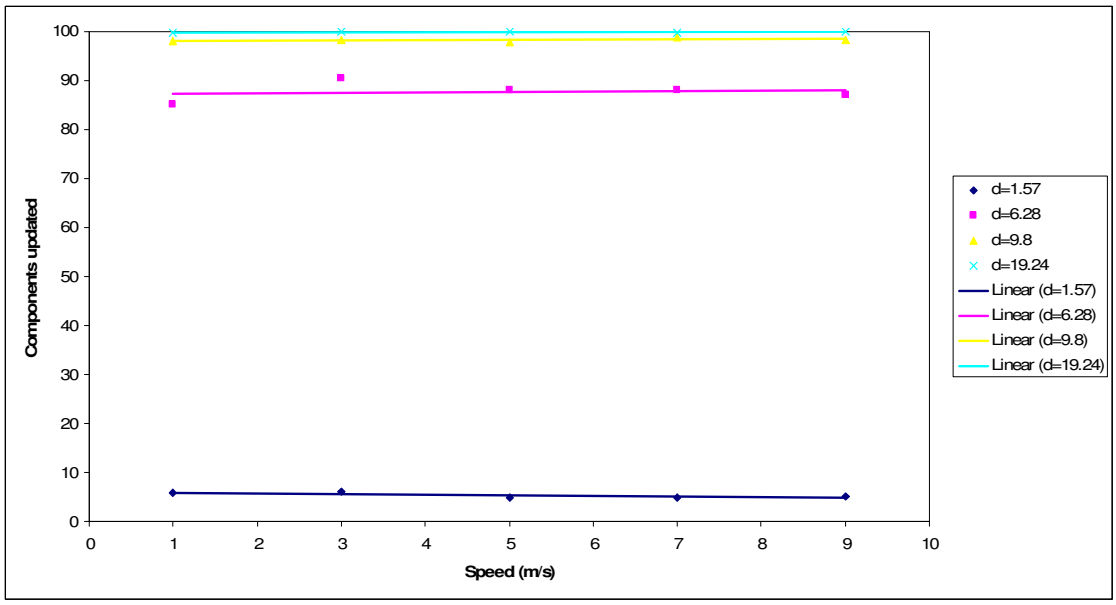

Figure 43: Update completeness

As one can see node density appears to be the only influencing factor on update completeness. With low node densities update completeness is very low but this is due entirely to the disconnected state of the network and the inability for the update to reach agents on disconnected nodes. As the density increases to a fully connected network then updates approach completeness regardless of mobility speed.

The age of data of the agents which did not receive an update is measured next to establish how useful their information is. The age is measured in terms of revisions, with the initial data having a revision of zero, and each subsequent update increasing the revision by one. The results shown in Figure 44 demonstrate that with low node densities where connectivity is extremely low then one can expect data to be in most cases unusable, although increasing node mobility does reduce this as the network connectivity changes frequently; however, even at 10m/s the data is still over 30 revisions old which may not be useful. Once node density increases to provide a mostly connected network then the age of the data drops to between 1 and 2 revisions. In the scenario tested this would make the data between 10 and 20 seconds older than the last update which may render it still useful especially in networks with low mobility. It is worth noting though that this data is being returned by less than 15% of the agents and the rest are returning

current information. Finally, with densities which provide high connectivity then updates are complete and so the age of data is always zero revisions old.

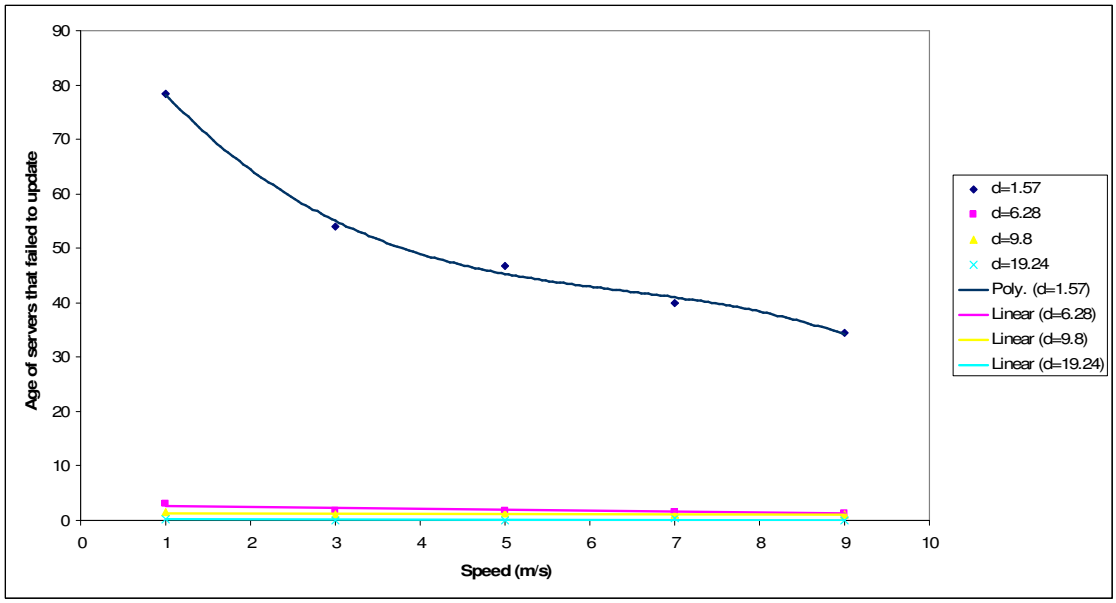

Figure 44: Age of data in agents that failed to update

In conclusion, with the exception of poorly connected networks which are likely to result in delivery failure anyway, updates complete fully with high probability and even in cases where the update is not complete the data returned by that small section of the sever is still potentially usable (see section 5.2.3). One could implement a multiple agent query to mitigate the problem of old data from the minority of agents.

To illustrate the likely density in a city-wide network two cities are compared using population density. The average population density of the UK was 383 people per square kilometre in 2001 according to the 2001 Census. If each person has a wireless device then the node density will be d=12 with r=100m, which would provide update completeness according to the results. Cities have much higher population density (London 4699/km² d=148) which are the most likely application area of large scale ad hoc networks and so the results show that in the average case updates are near complete.

5.2.2 Determining when to update

As discussed in section 3.7, several techniques exist to determine when to update a location server: Distance and Timer-based schemes

In this section, several experiments are analysed to ascertain the technique that provides the highest delivery success and the lowest communication overhead. Ten nodes are chosen at random and paired with one another; they then send packets bidirectionally which are routed using Most-Forward-within-Radius. The location server is simulated as a zero-cost globally accessible database of nodes' locations. This server is updated according to the scheme used in the following subsections.

5.2.2.1 Timer-based update

In this scheme the information is updated at regular intervals dictated by the update interval. Each node creates its GLA and then sends an update after waiting a specified number of seconds. After each update, the node then waits the specified time again and sends another update.

Figure 45 illustrates the delivery success of the timer-based scheme. As expected lower update intervals will result in higher delivery success.

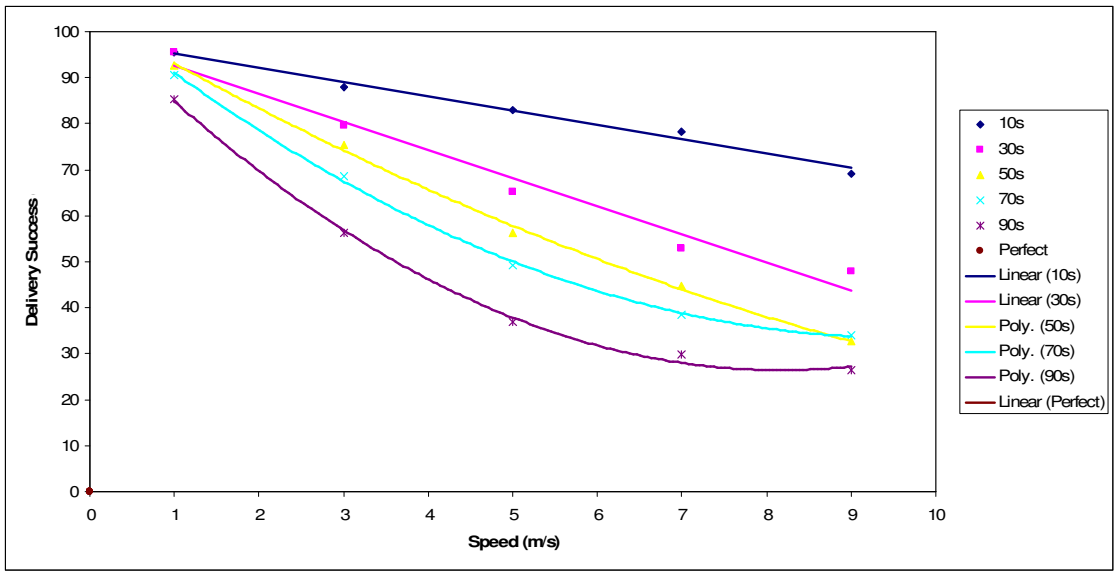

Figure 45: Delivery success of timer-based update scheme with one second beacon rate

Figure 46 illustrates the number of updates sent per minute with those observed in simulation. There is a slight variation due to the introduction of jitter when sending updates. This is to avoid synchronised nodes transmitting at the same time and causing a collision.

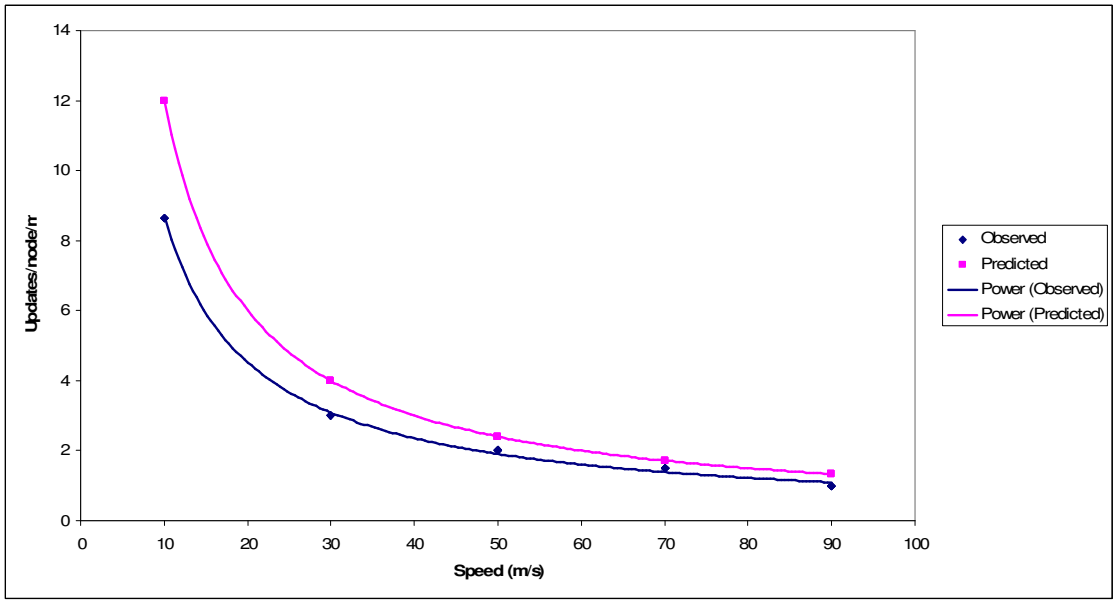

Figure 46: Timer-based update overhead

5.2.2.2 Distance-based update scheme

This scheme requires that node update their location server once they have travelled a certain distance. The reasoning for this is that if a node has not moved then there is little point in sending an update. When the timer-based scheme was used then the update would be sent every period regardless of the node's movement. Therefore, using a scheme based upon distance can provide an adaptive way to controlling overhead based on network mobility. Lower speed networks will incur lower overheads. In addition, this is further supported by later results showing that geographical routing can tolerate small errors in location. To examine the performance of distance based update scheme, a simulation is performed similar to the last subchapter, but varying the distance threshold between updates.

Figure 47, examines the distance at which one should set the threshold to. The radio ranges are chosen arbitrarily based upon ease of implementation and are not significant choices. One can see that a 10m interval is almost indistinguishable from the perfect scenario where the location server is updated constantly. Increasing the distance causes the performance to drop off as expected.

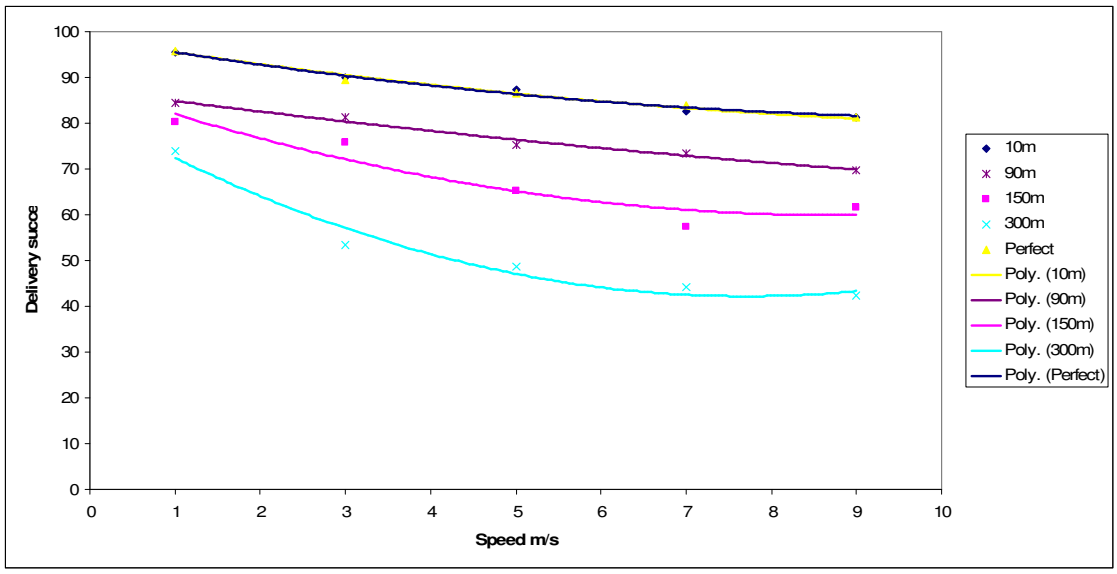

Figure 47: Distance-based update with one second beacon rate

Figure 48 illustrates the number of updates per minute required for various thresholds.

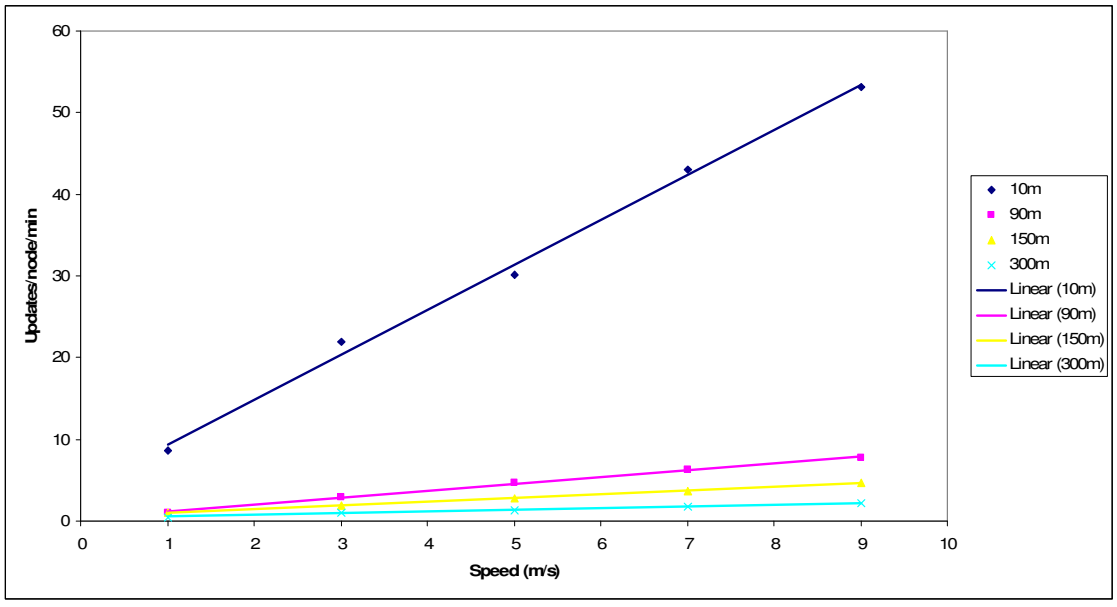

Figure 48: Distance-based update overhead

Previous authors (Thomas et al., 1988, Wu, 2005) have estimated the time between node updates given the node's speed and assuming that it moves in a random fashion. The

time a node stays within a region is distributed exponentially with the mean t (Wu, 2005), given the area A, perimeter L and average speed v:

$$t = \frac{\pi A}{vL} \tag{17}$$

Therefore, one can deduce the time that a node stays within a circular region with a radius d equal to its distance threshold, will be defined as exponentially distributed with mean $t_{d,v}$:

$$t_{d,v} = \frac{\pi d}{2v} \tag{18}$$

Therefore, this can be used to predict number of updates per minute and verify the simulation. The number of updates per second is given as:

$$u_{d,v} = \frac{60}{t_{d,v}} \tag{19}$$

To confirm that this method does indeed provide a close approximation of the number of updates sent, the results from earlier simulations are compared with a plot of the theoretical equation above. .

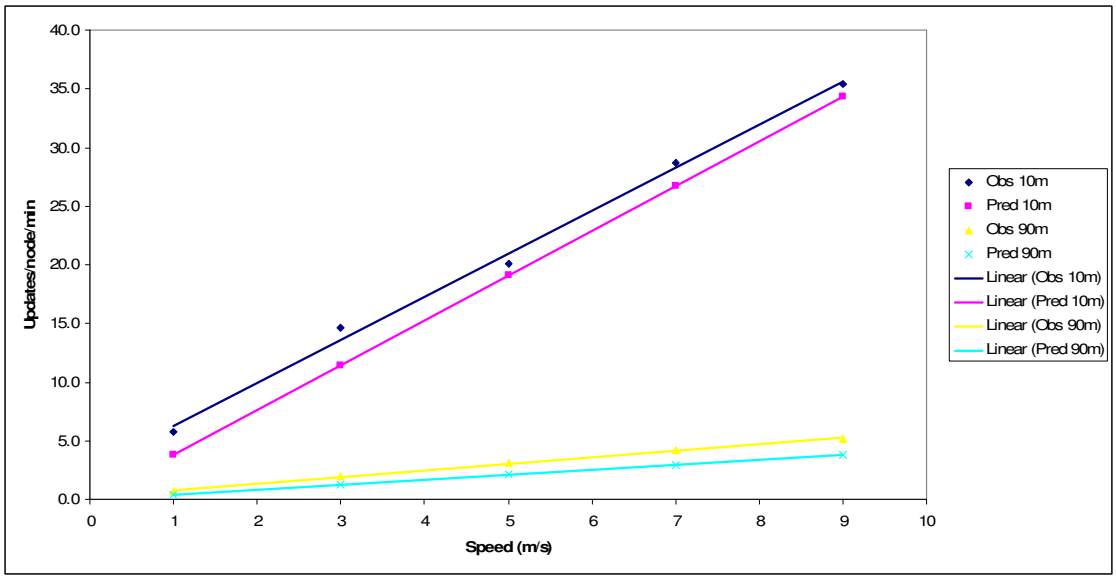

Figure 49: Distance-based overhead: predicted and observed

Figure 49 shows the comparison and demonstrates that the equation provides a close approximation. There is a slight difference due to the randomness introduced in simulation, but both predictions follow the same trend.

5.2.3 Tolerating Location Inaccuracy

The location stored in the GLA maybe out of date because it may have been some time since the last update was received. This is unavoidable, as even if a node sends an update every second then a query could return a result just below a second old. This is unavoidable due to the discreet nature of networks and so therefore it is important that the effect this has is examined, then the location inevitable as even a small amount of time will have passed in the situation where the information is received immediately after the update has completed. The question remains how MFR and perimeter routing will cope in the face of a small position error, and just how much error is tolerable before routing failures occur.

To examine this, a simulation with parameters described in section 8.2 is created. Location information is provided at zero cost so as to eliminate any other causes of failure. An artificial error is introduced into the location returned and this error is

constant amongst all nodes in the network. To eliminate the case where the routing techniques successfully handle a fixed position offset, such as to the bottom right by 50m, the value chosen is either added or subtracted from the x and y value of the nodes' location The method is chosen randomly from the four possibilities, where the (x, y) denotes the multiplication of the error for each axis: (+1, +1), (+1, -1), (-1, +1) and (-1, -1).

Only radio ranges that guarantee a connected network are chosen to eliminate disconnectedness as the cause of failure. Initially, MFR routing is compared and the results are presented in Figure 50 and it is observed for small accuracy errors there is little or no difference in delivery success.

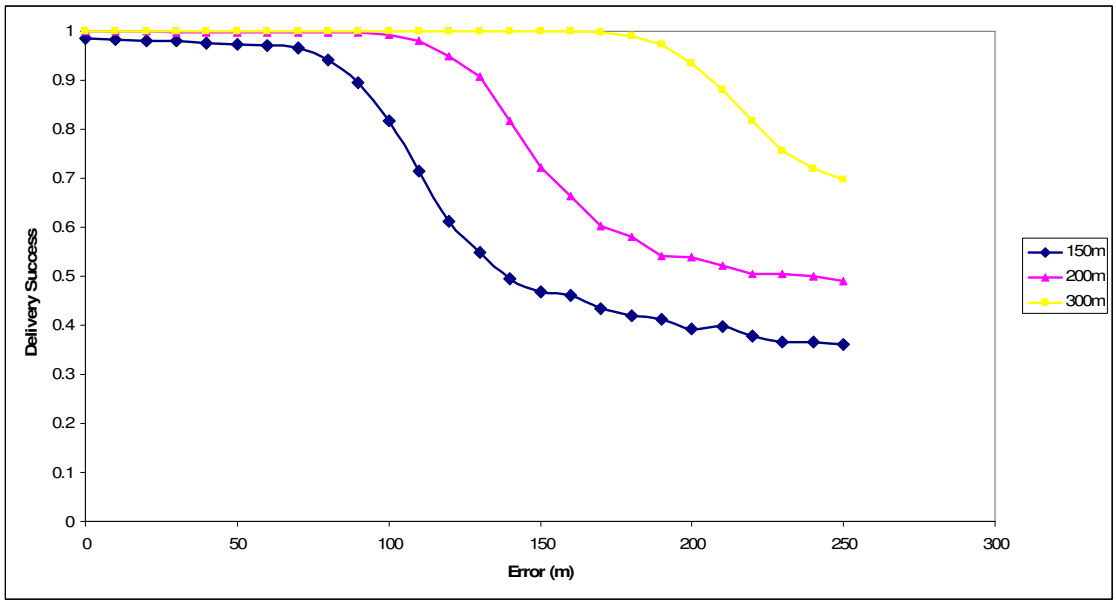

Figure 50: MFR routing only

Figure 51 demonstrates the effect of position error when perimeter routing is enabled. The delivery success drops off much later than MFR alone.

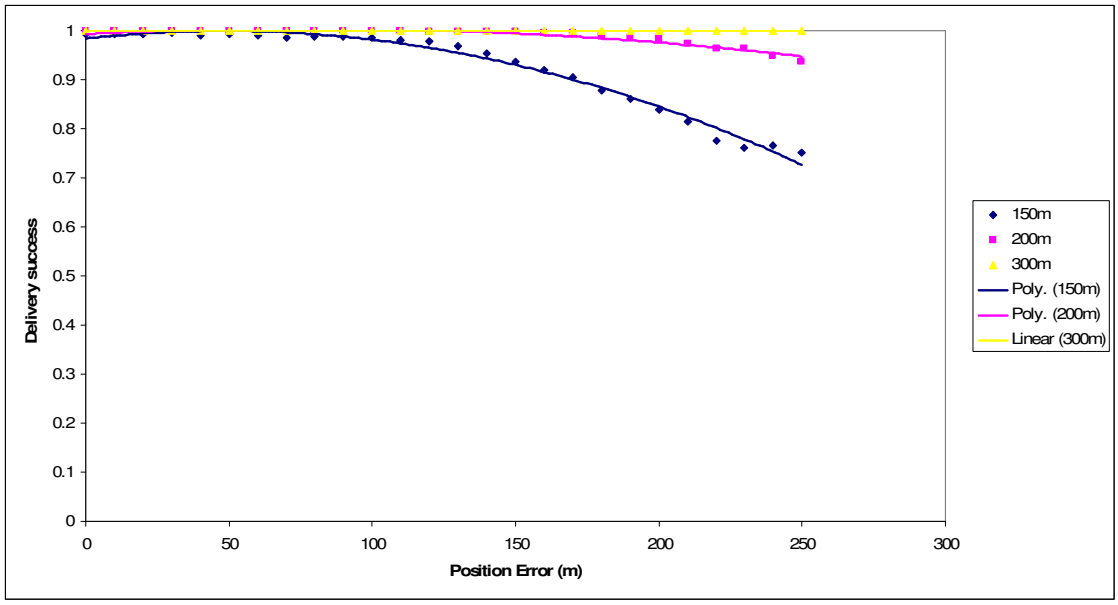

Figure 51: Perimeter routing

Both sets of results are plotted in Figure 52 as delivery success against the error as fraction of the radio range. The results above show that perimeter routing does improve tolerance to location accuracy by approximately two-thirds. MFR routing required nodes to be within 60% of the radio range's distance of the expected location, whilst perimeter could tolerate 100% of the transmission radius.

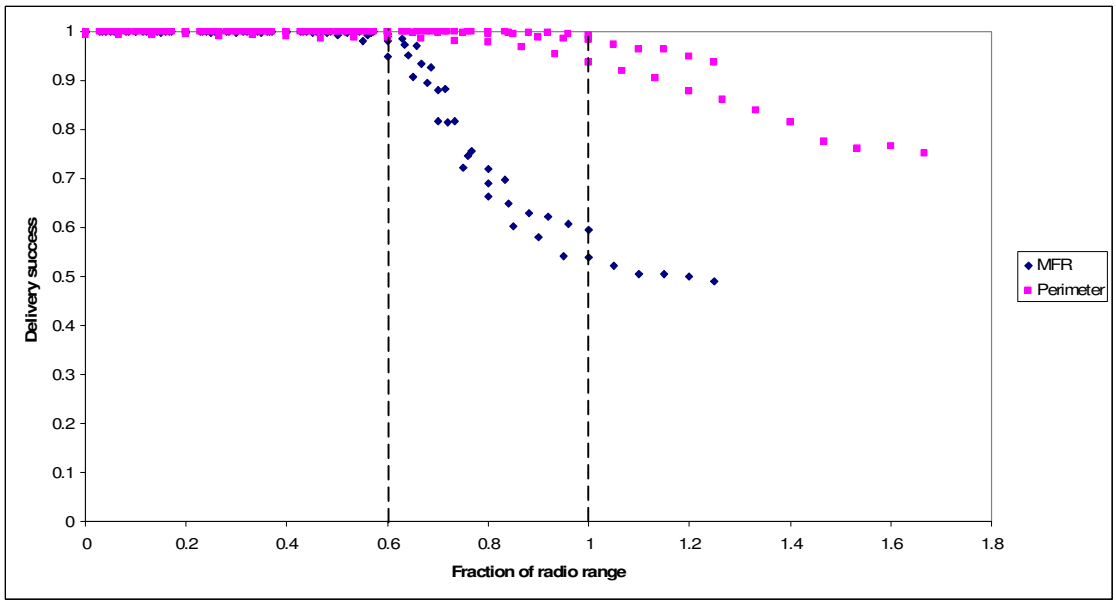

Figure 52: Position error as a fraction of radio range

Figure 53 is an illustration of why perimeter routing is more tolerable to location inaccuracy. The large dotted circle is the radio range of a node S, who is sending a message to node D. Node D has moved from the location S believes it is at, which is shown as a dashed circle. When S is unable to route the packet any closer to the believed location, then it assumes there is a void and employs perimeter routing. Node A is chosen as to route around this void which knows the true location for D from the beacons it has received. The packet is the sent to D and recorded as being received.

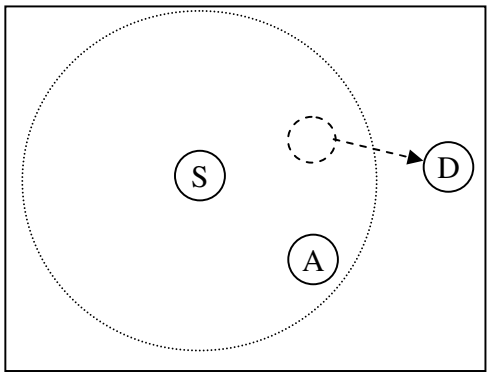

Figure 53: Perimeter routing's tolerance of location inaccuracy.

Using the results above we can estimate the minimum time (t_{MFR}) between updates required for MFR to succeed as a function of velocity (v) and radio range (r) for random motion by rearranging the equations for time between updates from section 5.2.2:

$$t_{MFR}(v,r) = \frac{\pi \frac{3}{5} r}{2v} = \frac{3\pi r}{10v} \tag{20}$$

Equally so, the time for perimeter routing is given as $t_{perimeter}$ below:

$$t_{perimeter}(v,r) = \frac{\pi r}{2v} \tag{21}$$

Both provide high tolerance of location inaccuracy, and when a node's range is 100m and it is moving at 10m/s (the worst case simulated in this thesis) then updates are required only every 9.15 (MFR) and 15.8 (perimeter) seconds. In the best-case scenario used of 1m/s then updates would be required every 94 and 157 seconds respectively. Figure 54 illustrates this graphically showing the required number of updates per minute against the node speed.

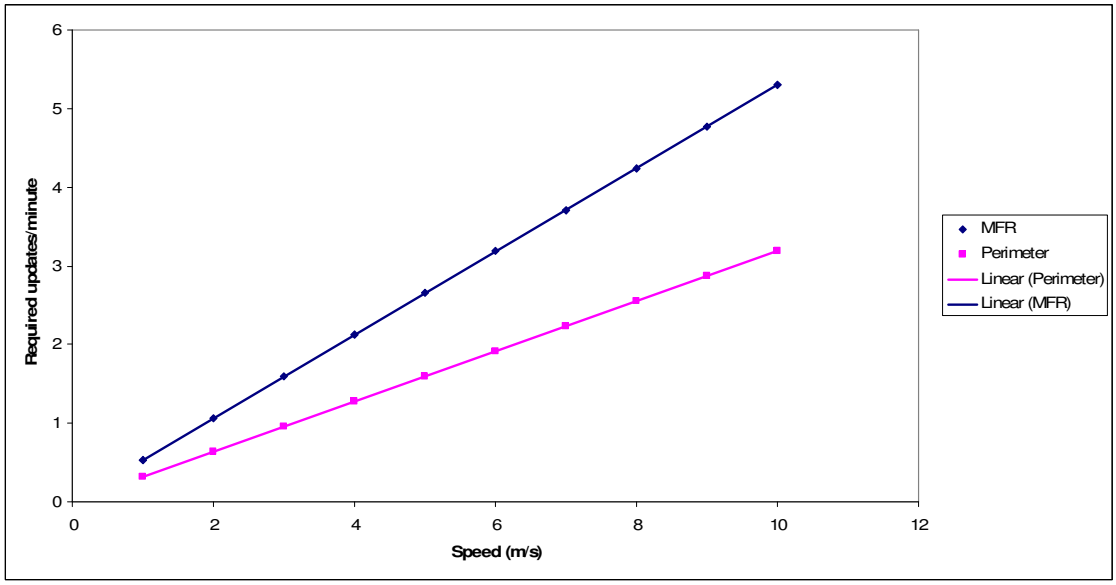

Figure 54: Updates required per minute vs. speed

This chapter has shown that the system is tolerable of large location inaccuracies and so failed updates to the location server are not necessarily detrimental to routing success.

5.3 Querying the server

The GLA is now able to store data at or near a specific point in an ad hoc network and so it is now necessary to be able to query this information. Firstly, one must know where a node's GLA is so it can be achieved through a variety of methods show below; however, this is only mentioned to give context to our work and the focus of the thesis is not on home location discovery.

1. Hash function: Given a node ID, one performs a function H(n_{ID}) which returns the home location. In an ad hoc network, this could simply map home locations in a grid like fashion, with each location being equidistant from neighbouring locations, so as to maximise the load distribution across the network. This of course assumes that the location of the network is known and will not change.
2. Discovery methods: Broadcast search and other more optimal search schemes have been proposed for discovering location servers; however, none consider how a discovery mechanism with a flat, or indeed hierarchical (Wolfgang et al., 2004), approach would work with millions of nodes.
3. Social dissemination: Mobile phone numbers are disseminated by two main processes: Social dissemination and Telephone number lookup services. For personal mobile phones, social dissemination is by far the most used means for sharing phone numbers, where individuals pass their number of to people.

Much of the literature concentrates on techniques for discovering the home location for a particular node ID; but, if someone wishes to communicate with another, what node ID does he use if he only knows the person by name? VoIP applications on the Internet typically use social dissemination and a telephone lookup service. So, in reality, the problem of discovering the node ID is the same as that of discovering the home location, and one could potentially substitute node IDs for home locations. If one is considering the application of a cellular network, then social dissemination for the most part, and an

Internet or telephone-based lookup service would suffice because the home locations would not change. One could consider implementing a distributed database to allow ad hoc network lookups, but as home location information will not change, this would place unnecessary load on the network for a service that would be best provided elsewhere.

Throughout the remainder of the thesis it is assumed the HL is known. A query consists of two messages. Initially a node will create a request for the information (GET), and when the server receives it, it should create a reply and send it back to the requesting node.

Two messages are defined for communication with a location server. A querying node ($s \in N$) will issue a GET packet, and the server will reply with a REPLY packet. When sending a request, the information needed to be known is: the querying node's ID (s_{ID}), its location (s_x, s_y) for sending the reply, and the ID of the node for which we're requesting information (n_{ID}). In addition, the location of the HL can be included if this cannot be calculated globally, but here it is assumed to be known. The reply packet needs to contain a requesting node's information so the reply can be routing (s_{ID}, s_x, s_y), and the information requested (n_{ID}, n_x, n_y).

$$GET(n_{ID}) = \left(n_{ID}, s_{ID}, s_x(t), s_y(t)\right) \tag{22}$$

$$REPLY(n_{ID}) = \left(s_{ID}, s_x(t), s_y(t), n_{ID}, n_x(t), n_y(t)\right) \tag{23}$$

Using these packets the next section will examine how the GLS is queried.

5.3.1 Querying process

Querying of the GLA is achieved by sending a GET packet towards the home location, routing it by the most suitable scheme (e.g. perimeter routing where necessary). Along the packet's route to the home location, it should encounter a node hosting an agent of the desired GLA. If the packet reaches the closest node to the home location, and has not met any nodes hosting agents then the query fails, or some other method must be used to find an agent (e.g. localised broadcast search). Upon encountering a node with an agent,

a REPLY packet is created and routed back to the requesting node. This has the effect that slave agents are queried most often as usually the master is surrounded by slaves in all directions. When a node receives the get packet (p), it executes the RECEIVEQUERYPACKET(p) function. This function retrieves the agent from the desired GLA if it exists and creates a reply; if no agent exists the packet is forwarded toward the HL.

```
RECEIVEQUERYPACKET(P)
1   a ← GETAGENT(p.n_ID)
2   if a is not null then
3       ROUTETOHL(p)
4   Else
5       INITIATEQUERYREPLY(a)
```

Figure 55: Receive Query Packet function

Figure 56 illustrates the number of queries that are answered by slaves using the query scheme described above. In the simulation results below (Figure 56), 80-90% of queries are answered by a slave which is due to the slaves surrounding the master in most cases. Certain node arrangements or high numbers of nodes in the locality (more than the number of agents) will leave a path through the slaves to the master in some circumstances which accounts for the 10-20% of queries to it.

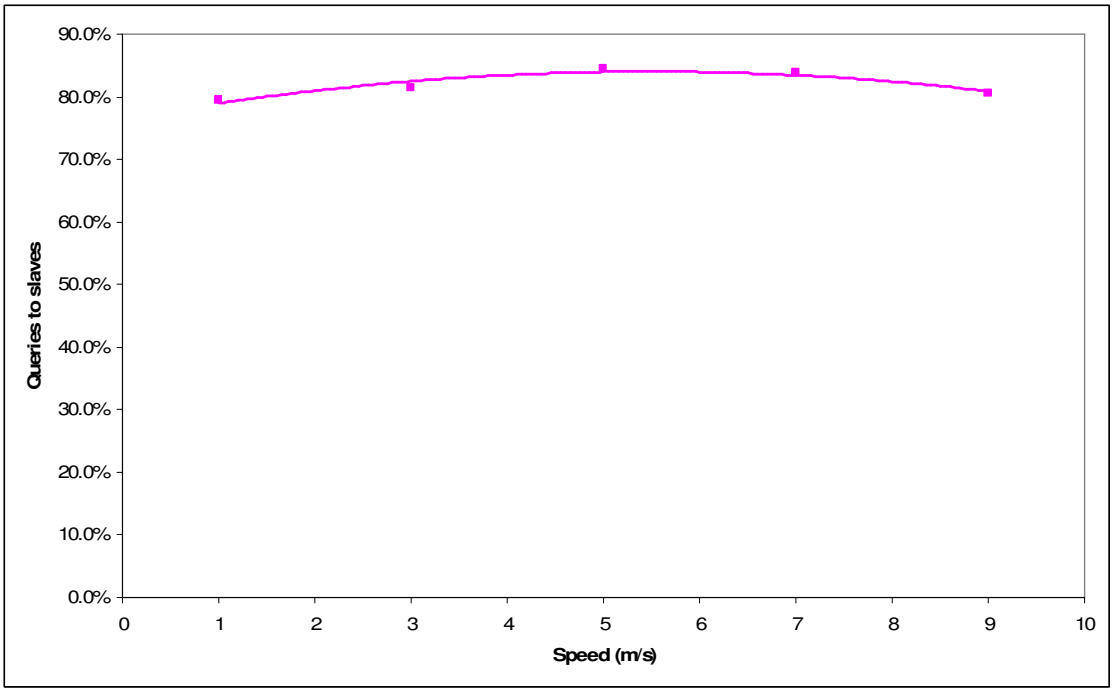

Figure 56: Percentage of queries that are received by slaves

Probabilistic quorums, such as the one proposed here, are known to return out-dated information (Abraham et al., 2004). Therefore, the results of queries to the GLA are examined to find what percentage of queries return old information. The reason this may happen is due to an incomplete update which will be caused by some slaves being out of range of others. The higher the speed the more likely this situation is likely to occur. Figure 57 shows the number of queries which returned old information and illustrates that this increases slowly with speed. At slow speeds around three percent of queries return old information, although this does not necessarily have a significant effect on delivery success as discussed earlier in the thesis.

The update success of the GLA was examined in an earlier section, and the age of data held by agents that had failed to update. The results here differ in that they show the age of data returned which may differ from the previous results if, for example, the failed to update agents are always queried due to the query method. To measure the age of data returned by the GLA, a definition is first established of correctness rate for queries:

Definition: *The **Correctness rate** (C_r) for a GLA Q, is a measure of the number of up-to-date queries (Q_u) against total number of queries (Q_t). The correctness rate is therefore defined as:*

$$Cr(Q) = \frac{Q_u}{Q_t} \tag{24}$$

Correctness rate indicates the percentage of queries that return current information. To examine this, a simulation with the standard parameters is configured and the age of data returned in queries is measured. Figure 57 is a plot of the results showing that even at high speeds over 90% of results are current.

.

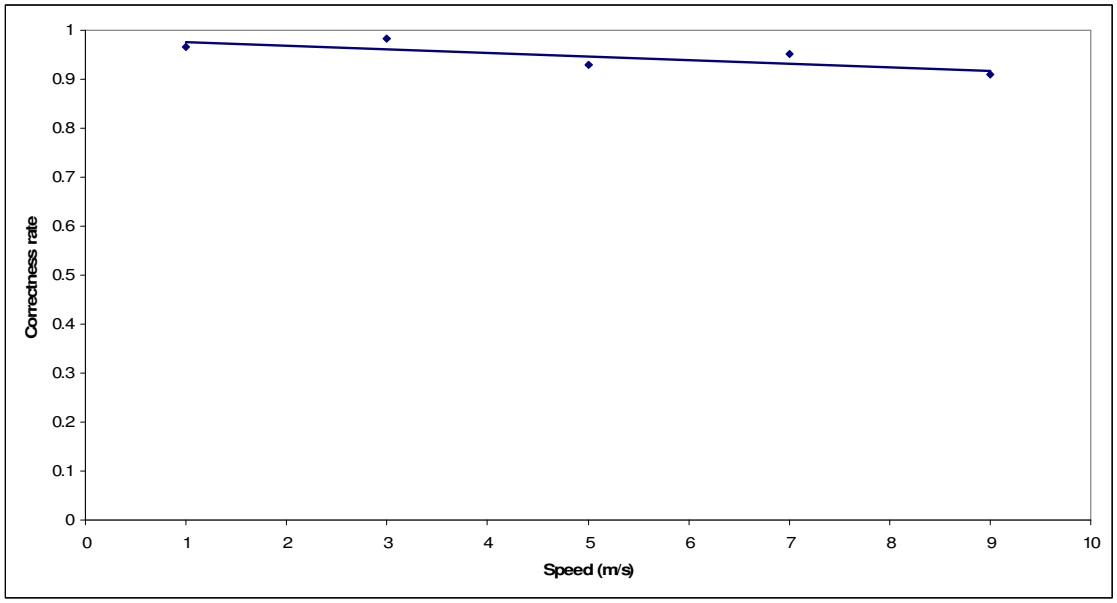

Figure 57: Percentage of queries returning old data

The results above differ slightly from the examinations of GLA update success which is attributable to update packet loss due to the nature of the wireless environment.

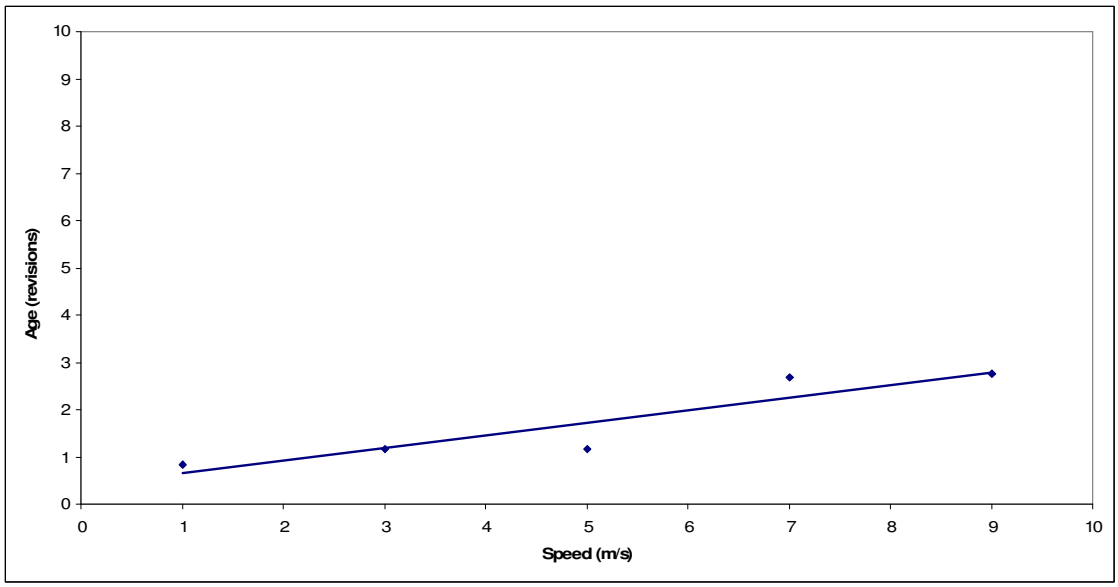

Figure 58: Age of old data returned by query

Figure 58 shows the age of queries that returned old data. The age is measured in revisions, where an age of one indicates that the query returned information from the update before the current one. The results show that for low speeds the age can be approximately one revision. Again, this is the age of data returned by the few agents that returned old information.

5.4 Handling voids

5.4.1 Non-convex voids

The Terminode home-region approach is unable to work when the home region is void of nodes; however, because the SOLS approach does not rely on a region but instead on a geographical point indicating the intended home it is able to cope with voids. Let us take the example in Figure 59, the area around the HL is void of any nodes. D wishing to send an update, or set up the server routes the packets towards the point. On encountering any void the algorithm switches to a perimeter routing strategy such as those described in section 2.3.3. The packet is routed around the perimeter and upon finding the node around the void to the HL, it sets up the GLA there. All the agents continue to attempt to move closer and should the void disappear or shrink then the GLA will move closer to

the HL. To query the GLA, S sends the query to the HL and upon reaching a void perimeter routing is employed and will traverse the perimeter until the GLA is encountered. Irrespective of how big the void is, or even if the geographical point is outside of the network area because the server will always migrate to the closest point on the perimeter employing perimeter routing will provide query success.

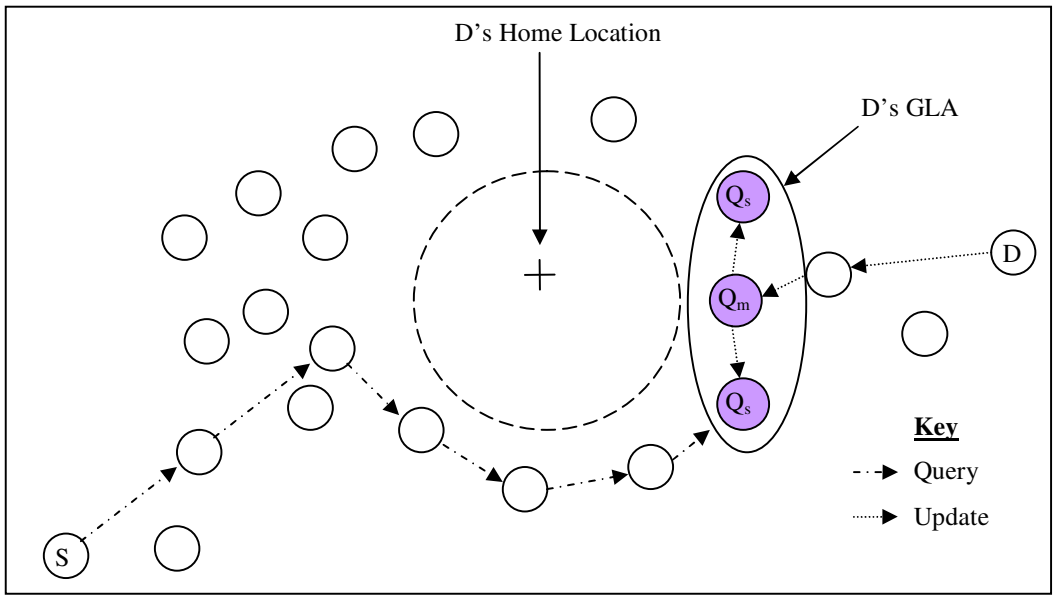

Figure 59: Handling voids when querying and updating the GLA

To examine the tolerance of voids by SOLS, the scenario illustrated in Figure 60 is simulated. This represents the worst case scenario when the centre of the network, where most of the traffic is routed through, is void of all nodes. Consequently some of the home location points will also fall within the voids and so the servers should locate themselves around the perimeter of the void.

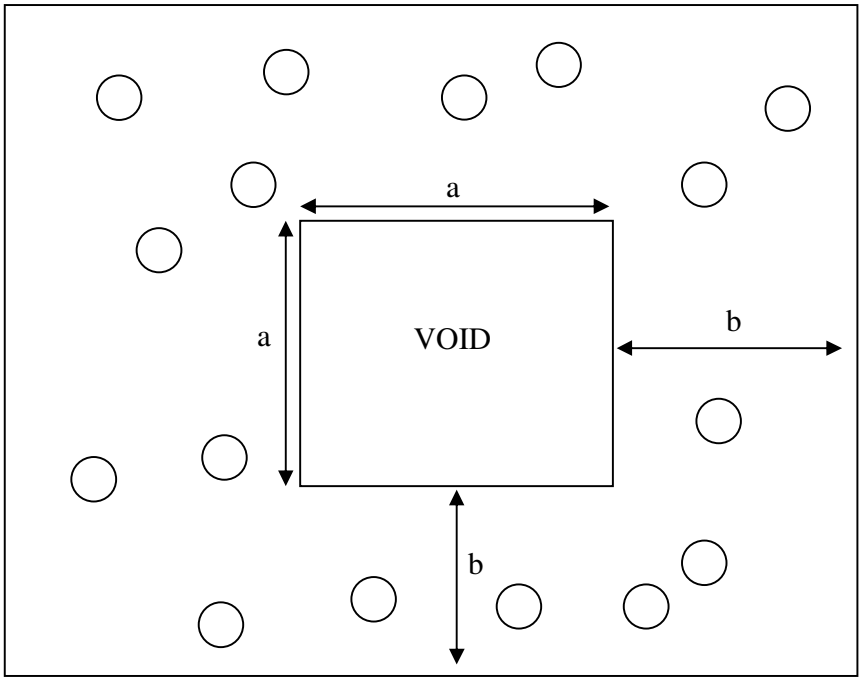

Figure 60: Simulation set-up for tolerance of void scenarios.

Before the scenario is simulated it is necessary to address a problem that SOLS encounters called the 'convex void problem'. This problem is outlined and countered in the next section.

5.4.2 Convex void problem

The convex void problem is encountered when the home location is inside an area void of nodes, which is bounded by a convex polygon. The reason this happens is illustrated in Figure 61 and the two steps that cause it are illustrated 1 and 2. At step one, the agent M reaches the closest point, assumes the role of master and clones itself to the nearest node(s). At point 2, one of the slaves sees a node which is closer than itself to the point and does not contain an agent. The slave then migrates there and can not see anyone else closer to the point and so assumes the role of master. This agent then initiates a cloning phase creating more slaves.

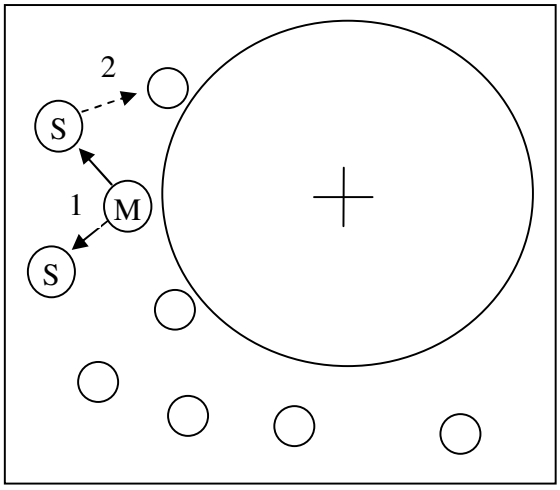

Figure 61: Illustration of the convex void problem

Figure 62 shows a simulation snapshot, with empty nodes as black dots, nodes with a master being purple and nodes with a slave being blue (circles indicate radio range of master). As one can see, this problem of slaves migrating closer and assuming the master role and subsequently replicating again has caused three masters to be created around part of the void's perimeter. Whilst this is good for failure tolerance, the task of updating the server becomes more costly and potentially incomplete should the three sections part. Therefore, this must be eliminated and can be achieved by attaching slaves to their master, rather than to the geographic point.

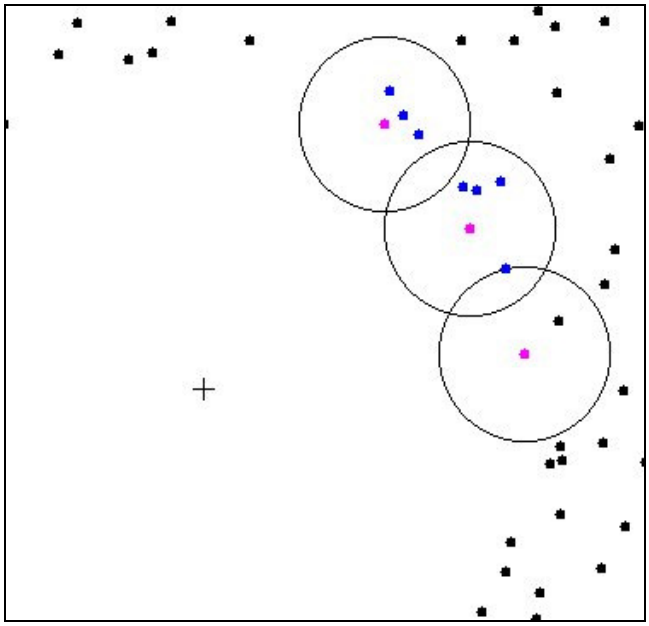

Figure 62: Simulation snapshot of the convex void problem

New slaves' target location to migrate to is set as the current location of the master. In addition, the slave also stores the home location; however, this is ignored initially. When the master is lost, a slave quickly migrates into its place and assumes the role of master. At this point, the new master retrieves the stored home location and sets this as its destination. Making this simple modification solves the void problem by preventing slaves from migrating away and attaching them to the nodes that are the closest to the master. If the void should disappear, one may wonder if the slaves would stay at the suboptimal point. This would not be the case as each agent in turn would move into the master's old position, be elected as master, and then follow the migration process to the home location. On reaching the home location, they would realise that they are not the closest to the point and resume the role of a slave. As replication only happens on reaching the closest point to the home region and this is never attained by the slaves they do not enter the replication phase.

A further problem is foreseeable, whereby nodes hosting the agents move in opposite directions away from the home location, leaving a void in its place. The agents would sit at different points on the perimeter and replicate. Such a scenario is unlikely in

simulation because the random movement of nodes means that voids are unlikely to form, or if they do, they do not last any significant amount of time. However, if this were to happen, the use of perimeter routing when attempting to get close to the home location would solve the issue, as agents would enter close proximity again on traversing the perimeter.

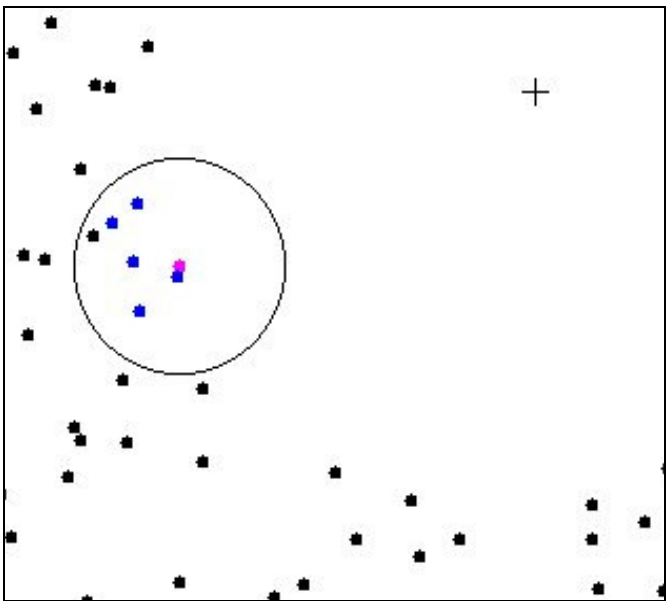

Figure 63: Simulation of solution to convex void problem

Figure 63 illustrates the effect adding these two techniques to the algorithm have on the void scenario. Simulation of voids with these two techniques was unable to replicate the earlier problem of multiple masters forming, even when duplicating the original scenarios.

A comparison on Terminode and SOLS in countering the void problem is simulated in the section 5.7.4.

5.5 Overhead comparisons

This subchapter will compare the overhead experienced with SOLS and a number of other location servers. It is important this overhead is kept to a minimum and is independent of the number of nodes in the network, so that it may scale.

A number of the theoretical analyses of the other location servers are taken from (Das et al., 2005). SOLS has a fixed creation overhead that is not dependant on the number of nodes in the network. It simply attempts to create a minimum number of agents and so therefore the overhead is presented as *O(1)*. Equally so, updates and queries have fixed overhead and are also not dependant on the number of nodes merely attempting to query one or a number of agents and to update all. The only task that is influenced by an outside variable is that of system maintenance that makes sure a GLA survives in a reachable location. The maintenance overhead increases linearly against velocity and so is proportional to $O(v)$.

This overhead is similar to the overhead encountered by Terminode with the notable addition of maintenance overhead; however, SOLS only utilises a small fixed number of nodes to store data whereas Terminode requires all nodes in an area to be utilised. Terminode is the most similar scheme to SOLS that is examined as it stores location in a particular area / region. The difference is that SOLS provides high fault tolerance by actively attempting to recover from failures where as Terminode relies on the survival of at least one server.

The horizontal and vertical quorum's overhead is $O(\sqrt{n})$ because it requires a horizontal and vertical column of nodes to be occupied. Whereas the grid location service is proportional to $O(\log n)$ because the number of servers used reduces the further you are from the node. The DREAM approach has overhead proportional to $O(n^2)$ due to the flooding technique used it requires every node to broadcast to every other.

To compare the failure tolerance of the schemes, a simple definition is established to differentiate between them, as follows:

Low: No attempts are made to recover from failed nodes / node mobility. The limited failure tolerance is derived purely from several copies of the data being available, although in a mobile network the data is likely to move away from the expected location.

Medium: No attempts are made to recover from failed nodes / node mobility; however, the system uses such a large number of nodes and they are likely to be reachable even sometime after the last update.

High: Active attempts made to mitigate failed nodes, node mobility and poor data accessibility.

The relative overheads and failure tolerance are expressed in the following table:

Table 5: Overhead comparisons

	Creation	**Maintenance**	**Update**	**Query**	**Failure tolerance**	**Hash function**
SOLS	*O(1)*	*O(v)*	*O(1)*	*O(1)*	High	Yes
Terminode	*O(1)*	*n/a*	*O(1)*	*O(1)*	Low	Yes
Horiz/Vert quorum	$O\left(\sqrt{n}\right)$	n/a	$O\left(\sqrt{n}\right)$	$O\left(\sqrt{n}\right)$	Low	No
DREAM	$O\left(n^2\right)$	n/a	$O\left(n^2\right)$	*O(1)*	Medium	No
Grid Location Service	$O(\log n)$	n/a	$O(\log n)$	$O(\log n)$	Low	No

As the table illustrates, SOLS and Terminodes are the only approaches which scale due to their fixed overheads for each node. Of the two schemes, SOLS provides fault tolerance and although incurring higher overhead due to maintenance, it is expected it will incur less overhead when other factors are taken into consideration; for example, Terminodes may fail when then the update frequency is insufficient, and so requires frequent updates even if the node is immobile whereas SOLS does not. Where SOLS incurs an overhead

of less than 10 packets per minute, each update to Terminode incurs an update overhead equal to the number of nodes in the region. In the scenarios simulated in this thesis the node density was approximately 6 per $100m^2$, which provides nearly complete connectivity (Ryu et al., 2004). As Terminodes is the only approach that scales, it is used for comparison throughout the rest of the chapter.

5.6 Implementation

The Terminode papers lack details on implementation of their home region and so this section will describe its implementation for comparison. A number of other techniques are used to optimise both algorithms performance such as location caching and they are described too.

5.6.1 Terminode's home-region

Although the authors of the Terminode's home-region infer the implementation of their system in their papers, they do not describe it explicitly. Therefore, the the implementation used for comparison is defined below.

Creation / Update

On creation of the network, each node chooses its home-region according to a globally known hash function. It then sends a packet to towards the home region and when a node inside the region receives the packet it will broadcast it. Any receiving node within the region that has not already seen the packet also broadcasts it. This ensures that all nodes inside the region receive a copy of the data and incurs an overhead equal to the number of nodes in the region.

To update the server, the same process is used but each update includes a unique sequential update number. This number is used by nodes receiving a broadcast to determine whether they have already received the update and if they have not then they will rebroadcast it. The packets used are equal to the data payload of the SOLS scheme.

Query

Nodes wishing to query the server simply send a query packet to the home-region, and on encountering the first node in the region with the data, then a reply is created with the information that node contains, and routed back to the querying node. The query packet is equal to that used in SOLS.

The home-region radius *R* is set at 100m, the same as the nodes' transmission radius. If the node density is six then one would expect six nodes to participate in the home-region.

5.6.2 Location caching

A location caching technique is used to optimise simulation of both Terminode and SOLS. If node A wishes to communicate with B, it first queries B's location server for its location. This location is then cached at A with an expiry time of the expected next update to the location server from B. A then sends packets to B and includes its own location in the packet header. B extracts A's location from the header and puts it in its location cache with an expiry equal to the A's server update frequency (this is globally known in simulation but maybe added to the packet). When B wishes to reply, it first checks its cache for A's location and if it has expired it queries A's location server. B then attaches its own location to the packet and sends it to A, who updates its location cache with the information.

This avoids the situation where nodes are querying location servers for every single packet they send. As the two nodes maybe in constant communication, attaching location information to packets allow the nodes to continually update one another to their location whilst still periodically updating the location server for others who may wish to communicate with it.

5.6.3 Transport of packets

As routing uses location information and this is different from any previously envisaged transport protocol, the transport protocol must be modified to include differing information. IP packets typically contain source and destination fields that denote the ID of the nodes concerned, in this scheme the packet will also include x and y co-ordinates

for the destination and source node. Here is the header of all transport packets (*P*) used in simulation:

$$P_{header} = \left(s_{ID}, s_x(t), s_y(t), d_{ID}, d_x(t), d_y(t), P_{ID} \right) \tag{25}$$

P is the packet itself with the header denoted by P_{header}. The header contains information (*ID*, *x*, *y*) about the source node $s \in N$, and the destination $d \in N$. In addition, a unique identification number is included P_{ID} that distinguishes the packet from any other in the simulation. In a TCP/IP packet this would typically be a sequence number along with source and destination fields but for simulation it is a unique number so that it is possible to isolate duplicate packets from inclusion in results.

Including this information in the header is necessary to allow forwarding nodes to make a decision on which neighbour the packet is to be sent to, rather than requiring each hop to discover the location information independently. This reduces location discovery overhead.

5.7 Results

Delivery success is the key indicator on the success of the system and this is primarily examined along with overhead (Owen and Adda, 2006a). As overhead is negligible, it will not have an effect on throughput and delay and so these variables are not examined. In these schemes, delay reflects the number of hops that query packets and data packets traverse which is not altered by either scheme. Use of multiple servers could reduce delays due to the increased likelihood of a server being closer to the querying node but will increase other overheads, and so is left for future work.

Three variables were isolated as most likely to have an effect on delivery success for both schemes. Speed, when coupled with update interval affects the accuracy of information stored at the location server. In addition, speed will move nodes participating in the location server away from the home location. Failure rate will result in the loss of nodes participating in the location server and high failure rates cause complete failure. Existing

approaches rely on frequent updates to handle most of these factors but when mobility is low these frequent updates incur unnecessary overhead; therefore, both schemes are also compared with varying update intervals.

5.7.1 Failure Tolerance

Most existing schemes, including Terminode, fail to address mitigation methods for fault tolerance. Therefore, it is important to examine the performance of the Terminode approach against SOLS with varying levels of failure rates. Both speed and update interval are also varied to determine their effect.

Figure 64 shows the effect of node failure on delivery success with varying update intervals. The choices for failure rate and node speed are arbitrary and have been chosen to reflect low and high failure/mobility but whilst still maintaining some survival. For all likely failure rates in real networks, SOLS outperforms Terminode by a significant margin; however, when failure rates are high then Terminode shows greater performance. Considering SOLS attempts to actively mitigate failures this is surprising. The reason is that Terminode effectively creates a new server with every update and one can see when the update frequency increases to 60s, then SOLS is only marginally outperformed by Terminode. If updates were more infrequent then Terminode would fail significantly earlier than SOLS.

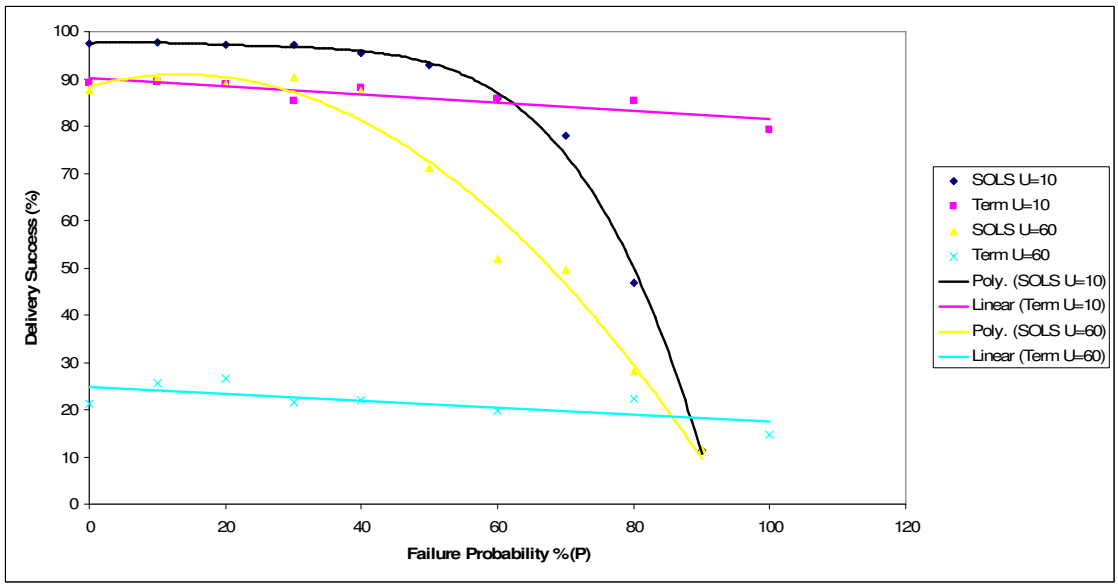

Figure 64: Delivery success measurements whilst varying failure probability (v=1m/s)

Figure 65 shows the overhead in packets per location server per minute for the above scenario. SOLS always has higher overhead because Terminode's overhead consists of only updates, whilst SOLS attempts to adapt to the changing environment. In either case, the overhead is negligible for modern wireless networks.

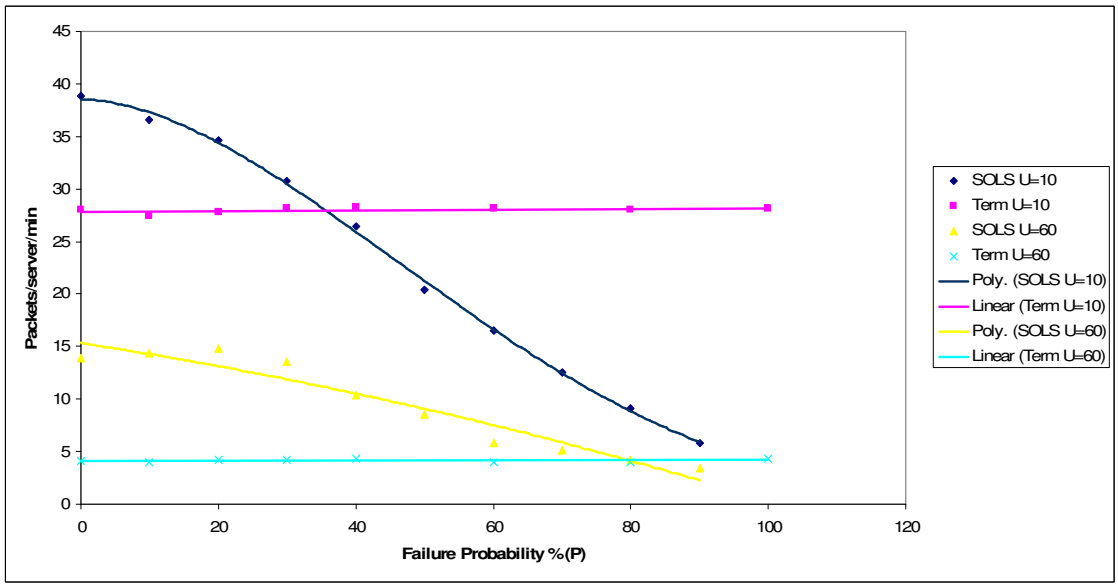

Figure 65: Overhead measurements whilst varying failure probability (v=1m/s)

Figure 66 shows delivery success for the above scenarios, but with a higher speed of 9m/s. Again for all likely failure rates SOLS outperforms Terminode, but this time by a much more significant margin. This is because of SOLS attempts to mitigate mobility.

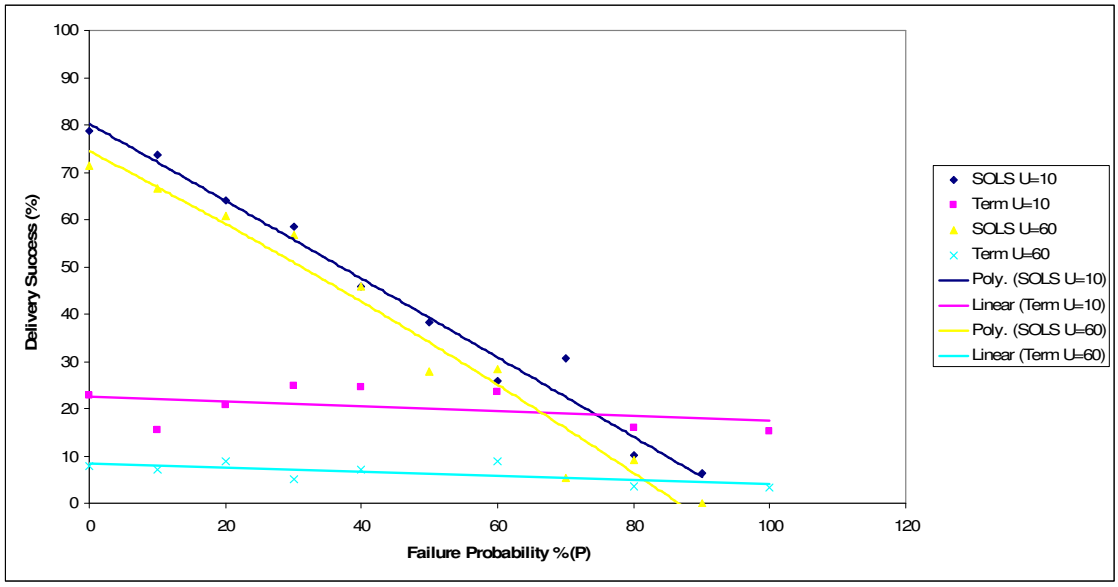

Figure 66: Delivery success measurements whilst varying failure probability (v=9m/s)

Figure 67 shows overhead per server per minute for the above scenario. SOLS incurs higher overhead than Terminode for likely failure rates, and this is again due to its attempts to mitigate mobility and failure in addition to update overhead; however, the overhead is much higher than the scenario with lower mobility and this is because GLS agents need to migrate more frequently.

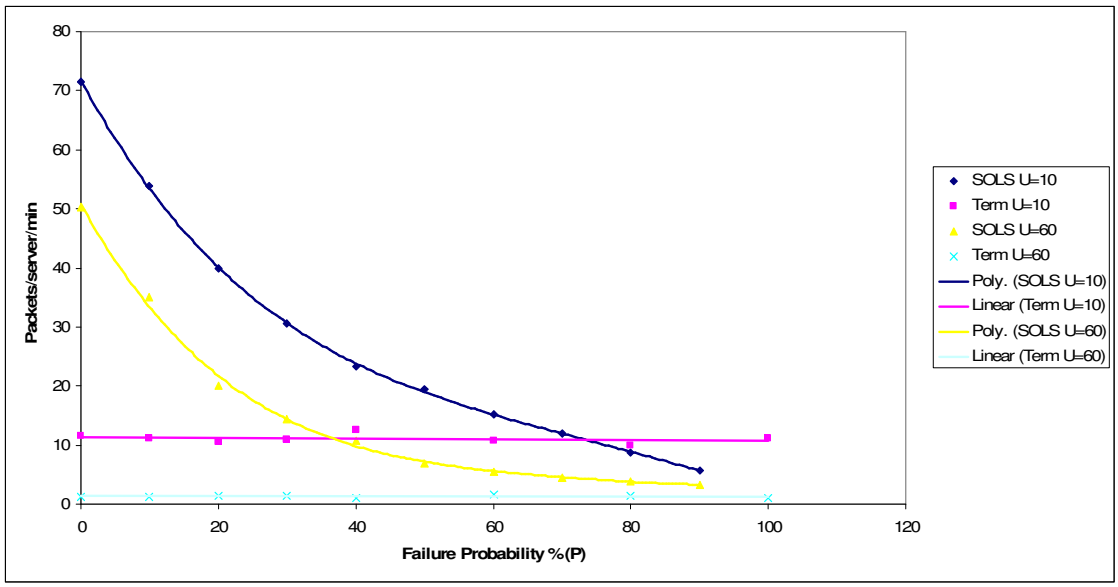

Figure 67: Overhead measurements whilst varying failure probability (v=9m/s)

5.7.2 Mobility Tolerance

The purpose of this section is to examine the mobility tolerance of Terminode and SOLS given that mobility is likely to be present in a wireless network. Figure 68 presents a comparison of the two schemes' delivery success with an update interval of 10 seconds. SOLS outperforms Terminode for all speeds examined by a significant margin. This is due to SOLS's attempts to migrate GLS agents back to the home location where they can be discovered; Terminode makes no such attempt.

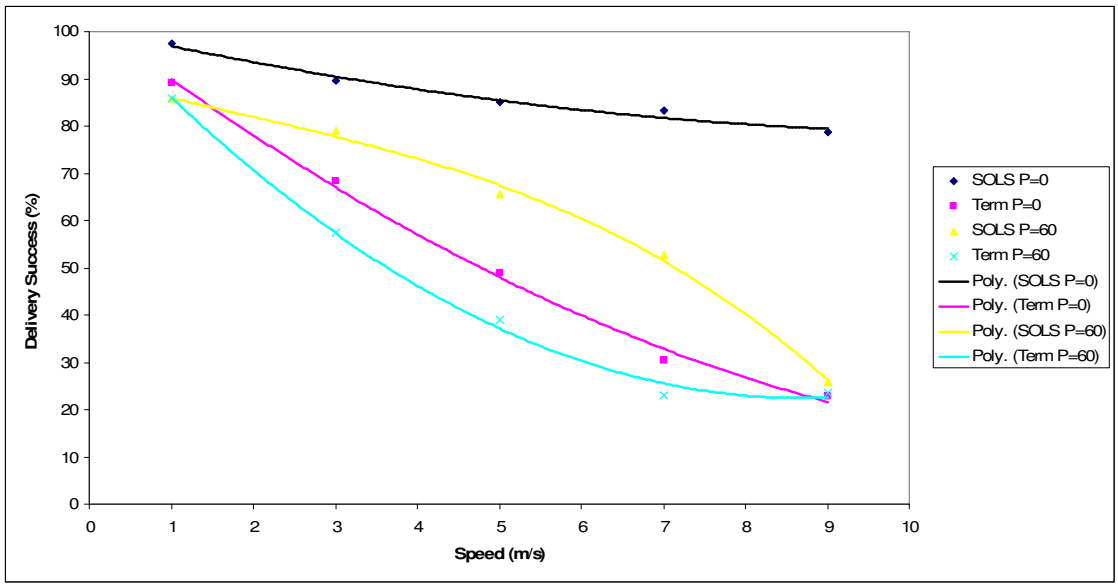

Figure 68: Delivery success measurements whilst varying speed (u = 10)

Figure 69 illustrates the overhead for the above scenario. Terminode has fixed overhead regardless of speed whilst SOLS' overhead increases with speed to mitigate mobility. SOLS' overhead with high failure rates is slightly below that of Terminode but this is because it is struggling to maintain the GLS with enough agents.

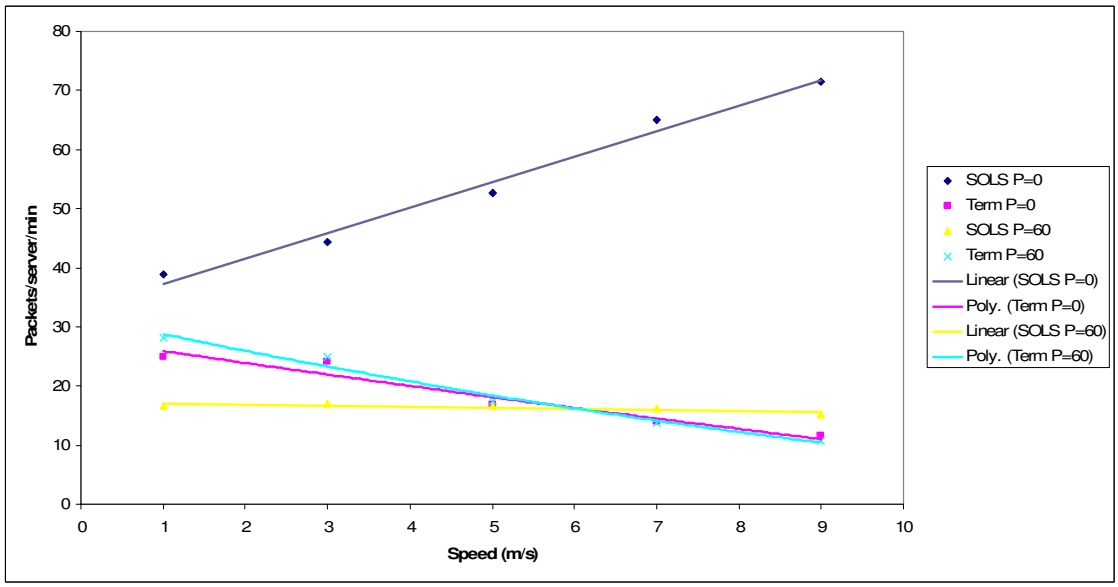

Figure 69: Overhead measurements whilst varying speed (u = 10)

Figure 70 and Figure 71 show delivery success and overhead respectively for the above scenario, but with update interval set to 60 seconds. Again SOLS outperforms Terminode at all speeds but also incurs higher overhead for reasons previously stated.

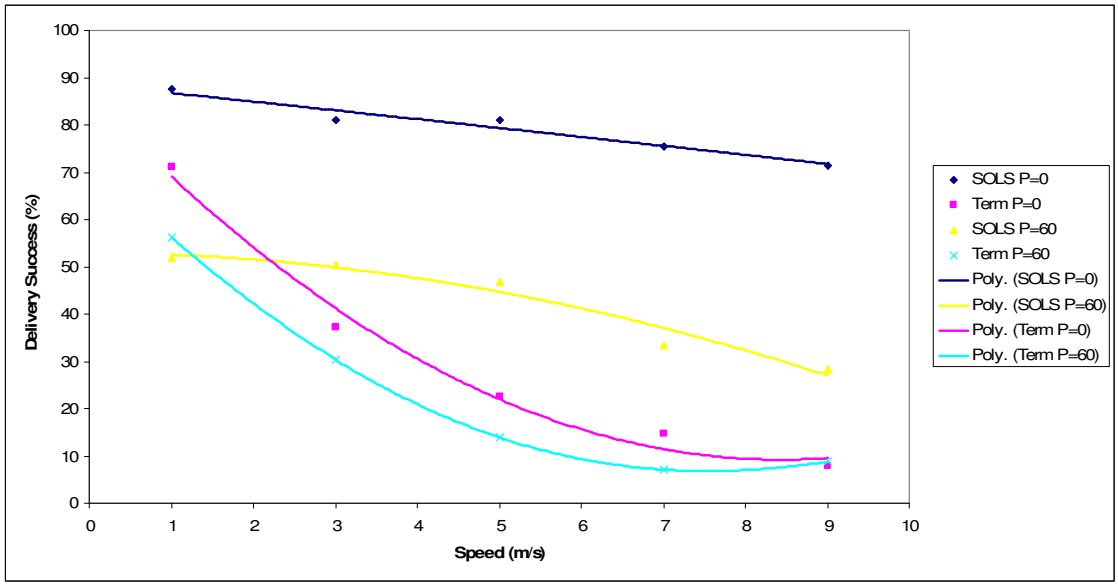

Figure 70: Delivery success measurements whilst varying speed (u = 60)

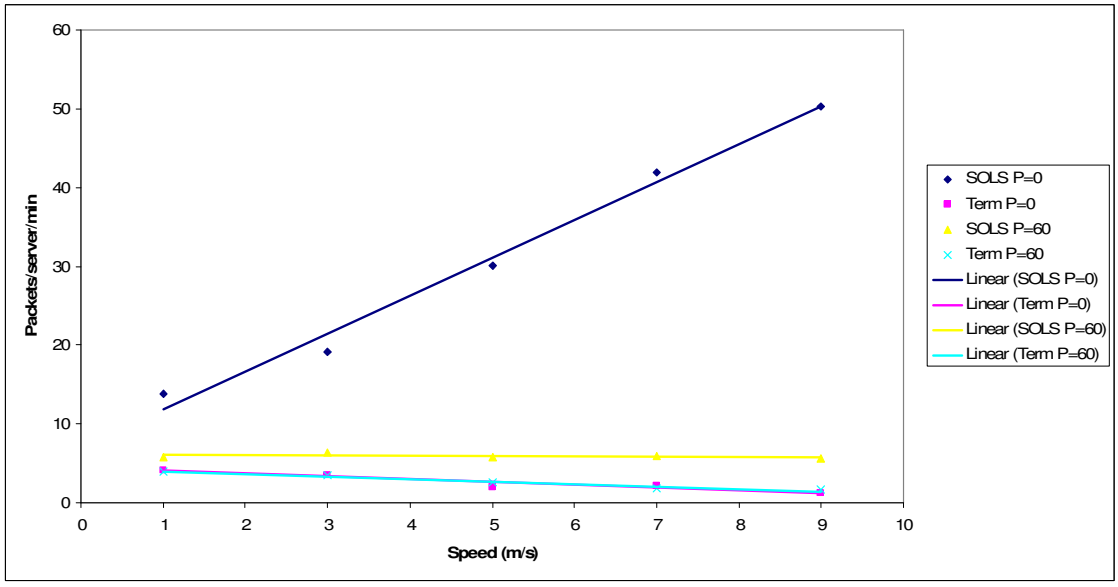

Figure 71: Overhead measurements whilst varying speed (u = 60)

5.7.3 Effect of update interval

As Terminode relies on frequent updates to maintain the location server it is important to vary the update interval to show how SOLS performs better.

Figure 72 and Figure 73 show the delivery success and overhead respectively, when update interval and failure probability are varied with nodes travelling at 1m/s. Delivery success of SOLS is above that of Terminode and drops off less quickly when update interval is high. With higher failure rates Terminode and SOLS perform similarly with Terminode slightly outperforming SOLS with higher update intervals. This is due to Terminode completely recreating the server at each interval.

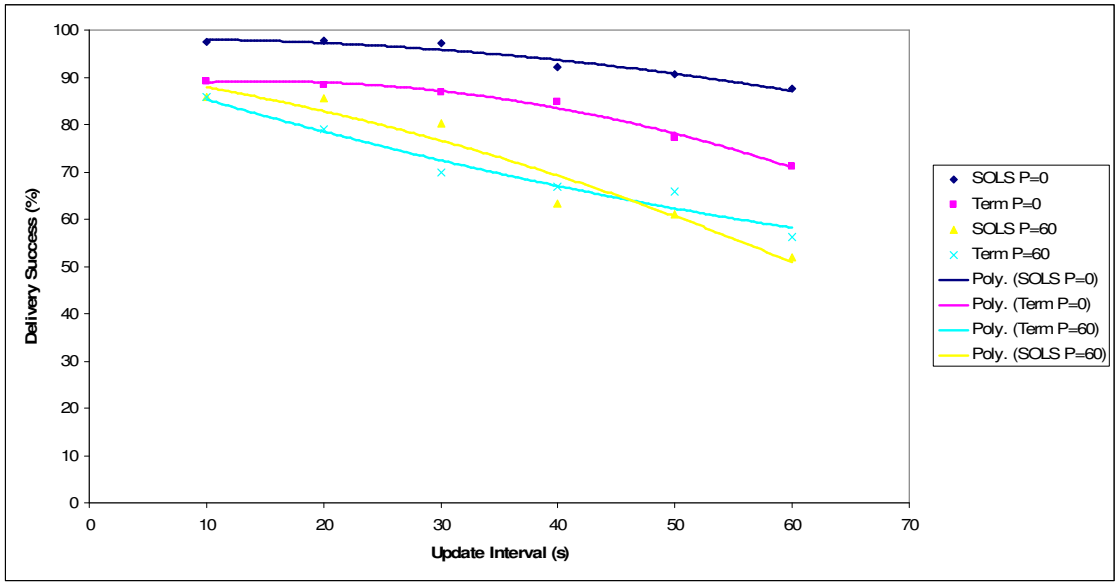

Figure 72: Delivery success measurements whilst varying update interval (v=1m/s)

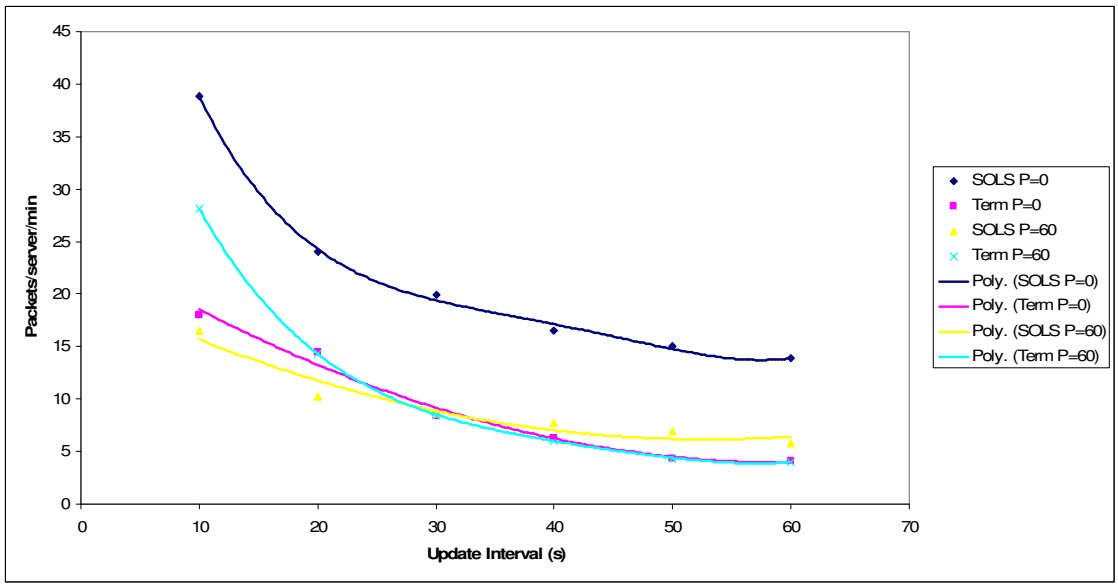

Figure 73: Overhead measurements whilst varying update interval (v=1m/s)

Figure 74 and Figure 75 show the delivery success and overhead respectively for the above scenario but where nodes are moving more quickly. In this scenario SOLS significantly outperforms Terminode in all scenarios. Terminode performs particularly poorly due to nodes quickly moving outside of the area shortly after an update, whereas SOLS actively mitigates this through migration. SOLS again incurs the highest overhead but when the update frequency is low, the overhead is similar to that of Terminode but the delivery success is higher.

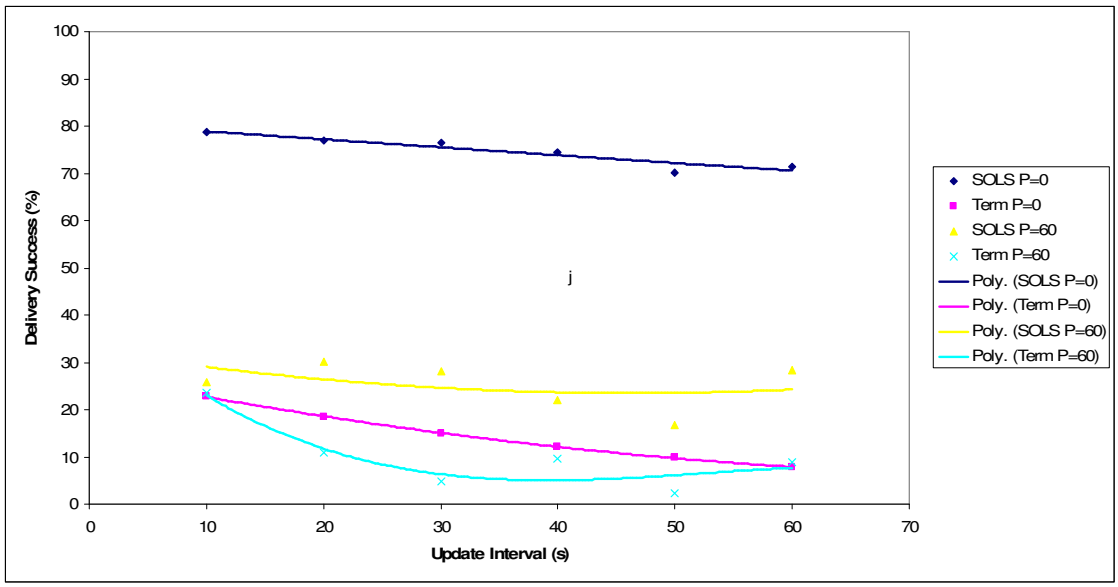

Figure 74: Delivery success measurements whilst varying update interval (v=9m/s)

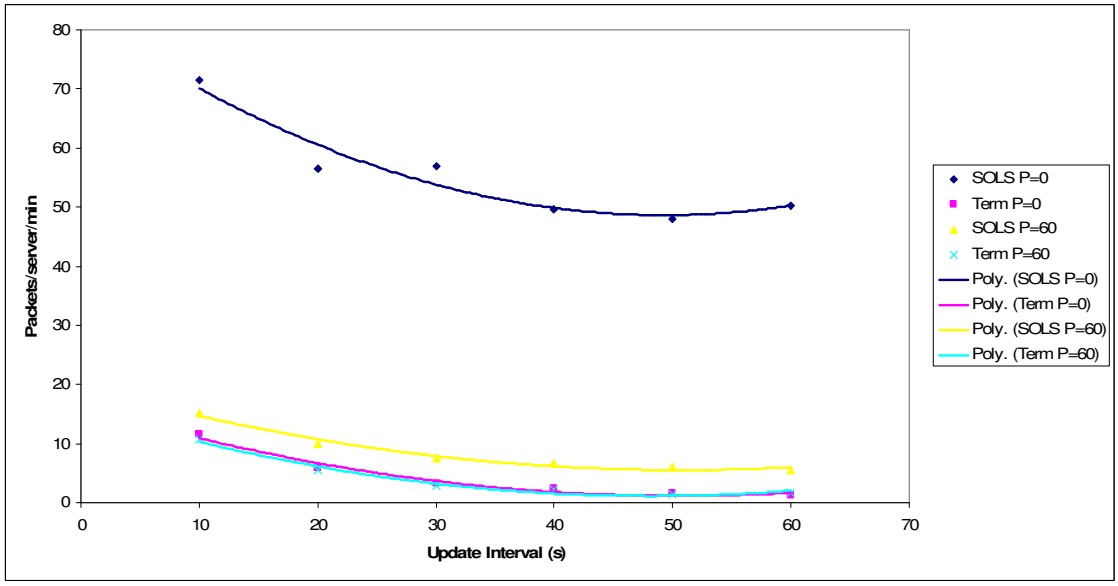

Figure 75: Overhead measurements whilst varying update interval (v=9m/s)

5.7.4 Voids

The ability of SOLS to handle networks with areas that are void of all nodes was explored in Section 5.4. In this section, Terminode and SOLS are compared in terms of delivery success when confronted with such a network. Just as in Figure 60, an area void of nodes is created of varying size in the centre of the network.

The scenario employs a static network with no mobility so that failures will be primarily due to inability to route around the void or due to location server discovery or deployment problems.

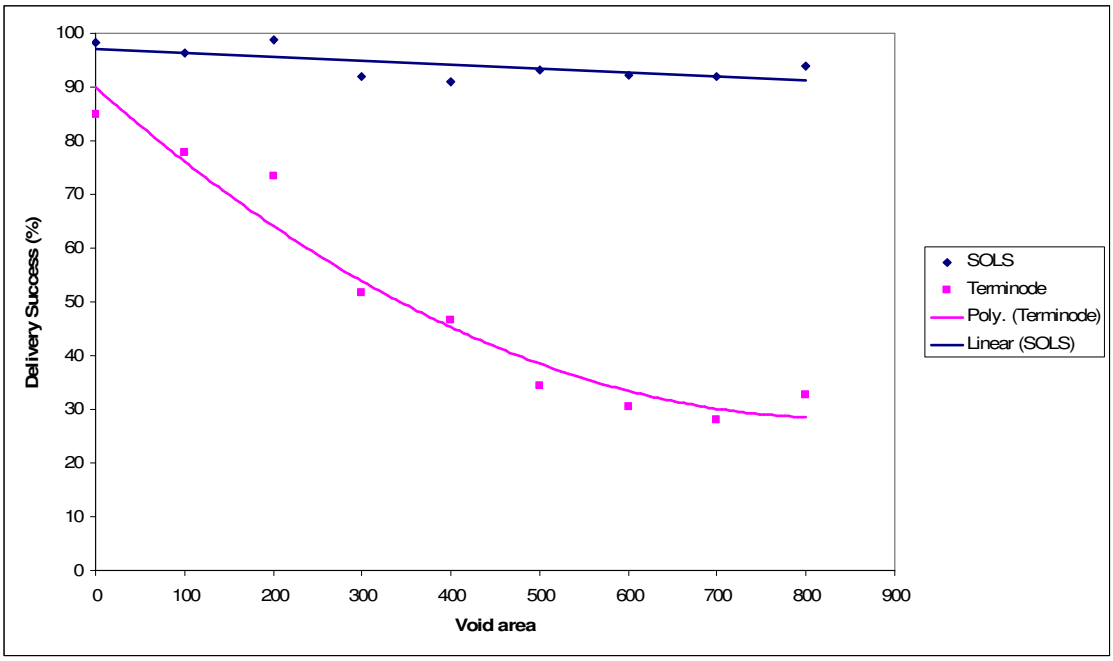

Figure 76: Comparison of ability of Terminode and SOLS to handle scenarios with voids

Figure 76 shows that the performance of Terminode drops of quickly as the void size increases. SOLS however maintains above 90% delivery success and the failures are due to the inability of the right-hand rule to route around the void. Even with very small void sizes (100x100m), Terminode's performance drops below 80%.

5.8 Conclusions

SOLS outperformed Terminode in most scenarios examined and in many case by a significant margin. Terminode had the advantage in that at each update interval the location server was created at full strength at the desired point; however, this benefit was mostly realised when the update frequency was high. When the update frequency was low and mobility or failure high SOLS excelled by a significant margin.

SOLS was rarely out-performed by Terminode and only in scenarios that are unlikely to be encountered such as exceptionally high failure rates. The overhead was consistently higher than Terminode but this is attributable to it having equivalent update cost with the addition of maintenance overhead.

Delivery success is the key indicator for a location server as it reflects the accuracy of the information returned, and the survival and accessibility of the server. SOLS attempts to mitigate both of these whereas Terminode relies purely on high update intervals to frequently renew the server. Failure rates simulated in the thesis are not likely to be encountered in the realistic scenarios envisaged and P=50 represents a single node failing every ten seconds of average. Below this failure rate SOLS always performed better in terms of delivery success than Terminode and because of this it can be considered to be the superior location server.

6 Implementation

6.1 Introduction

This section will examine the possibility of developing a large scale ad hoc network using the location server provided. It will examine how a large scale ad hoc network would be deployed and how the proposed service would be implemented with existing technologies.

We must agree upon what kind of ad hoc network we would be examining and the technology that we expect to deploy the work of this thesis in. There are a number of technologies currently used in the systems in which large scale ad hoc network technology could be useful, such as:

- Large sensor networks: Networks of small devices used for distribution sensor information, such as climate conditions. They typically use 802.11 wireless LAN technology, or Bluetooth (Sweeney, 2001).
- Cellular telephone network: Consist of cellular telephone towers, or base stations, that mobile telephones access using a radio transceiver. These towers then forward the calls to the tower nearest the destination, at which point the call is then transmitted to the destination phone. These technologies use GSM cellular networks in Europe and most countries around the world (Short, 1999).
- Wireless VoIP: Recently, Voice-over-IP has become quite popular with the commercialisation of consumer accessible software such as Skype (Khamsi, 2004). With the advent of Skype software, a number of manufacturers have started developing Wireless VoIP phones, that use a nearby wireless network to access the Internet and make telephone calls. These technologies use 802.11 wireless LAN technology.

Large scale ad hoc networks could be deployed in any of these to augment or replace the need for fixed infrastructure. Two of the examples above use 802.11 wireless LAN

technology which is cheap and accessible, and most importantly easier to modify. Take for example a PDA, one can access the wireless LAN interface and change the networking behaviour of the device; however, with GSM the software running of the phones is often proprietary and add-on applications are usually unable to access the underlying GSM receiver; although they can usually access the Bluetooth subsystem but its range is quite limited.

This chapter will focus primarily on implementing a large scale ad hoc network, and in particular the location server presented in this thesis, over a 802.11 wireless LAN, using PDAs or Laptops. It is assumed that the device has access to the Global Positioning System, which is not improbably given a number of manufacturers already make devices with this integrated, such as the O2 Orbit XDA and Mio A710 (partly due to the upsurge in satellite navigation usage).

Creating a system that is easily added to existing technology is important as requiring significant changes will hinder adoption. For example, if modification of an operating system kernel is required then co-operation from manufacturers and updating existing hardware is required, without which no implementations can be realised. Therefore, in this chapter, only techniques which can be implemented at the application layer will be examined where the application uses OS API calls that are available on operating systems such as Microsoft Windows XP/Vista, MacOS, GNU/Linux and UNIX *BSD variants. To achieve this it is assumed that an application will incorporate all of the functionality described and use standard API calls.

6.2 Beaconing

To perform geographical routing in a beacon scheme, one must regularly transmit its location to its neighbours in a broadcast packet. This permits neighbours within range to make local decisions about where to send packets.

The beacon packet contains the node's identifier and its location. The identifier can be any number that uniquely distinguishes the node from others, such as an IP address or

telephone number. The Internet Protocol is currently the most widely used transport protocol, due to its use in the internet. The current version (v4) is limited to 32-bits for the IP address and as such the remaining IP addresses are beginning to run out. The next incarnation is IPv6 which permits 128-bits for the address and is currently being rolled out across the Internet. As such, an IPv6 IP address is used to uniquely identify nodes in the beacon packet.

The co-ordinates need to be able to identify any geographical location to an accuracy of at least half the node's transmission range within the network area. In this thesis it has been assumed that a large scale network could be deployed across a city. Requiring the users to define the network area in advance defeats the objective of ad hoc networks and so it is therefore assumed that a node could occupy any location on the surface of the Earth.

The number of bits needed to identify a location with certain precision varies depending upon the node's latitude. At the equator, the more bits are needed than at the poles. The standard World Geodetic System 84 is used to represent the location which provides two values: latitude and longitude. Latitude describes the north and south axis which ranges from -90 to +90 degrees and longitude describes the east-west axis which varies from -180 to +180 degrees. The origin (0,0) is the point where the Greenwich meridian line crosses the equator. As one moves north or south from the equator, the circumference at that latitude decreases and so less precision is required to represent longitude.

A standard equation for calculating distance between two points across a sphere is shown below given the radius of the Earth (R), and the latitude and longitude of the two points A (φ_A, λ_B) and B (φ_A, λ_B).

$$d(\varphi_A, \lambda_A, \varphi_B, \lambda_B) = R\cos^{-1}\left(\sin\varphi_A \sin\varphi_B + \cos\varphi_A \cos\varphi_B \cos(\lambda_A - \lambda_B)\right) \quad (26)$$

Using this equation we can find the smallest value needed to represent the required precision for any latitude. By rearranging this equation, making λ_φ the subject, which is

the smallest value in longitudinal degrees required to represent a distance of d metres at a latitude of φ.

$$\lambda_{\varphi} = \cos^{-1}\left(\frac{\cos\frac{d}{R} - \sin^2\varphi}{\cos^2\varphi}\right) \tag{27}$$

The transmission range used in this thesis has been 100m and for MFR routing to succeed there needs to be a precision of 60m. Figure 77 indicates the smallest value of longitude that needs to be represented at each latitude to guarantee 60m accuracy.

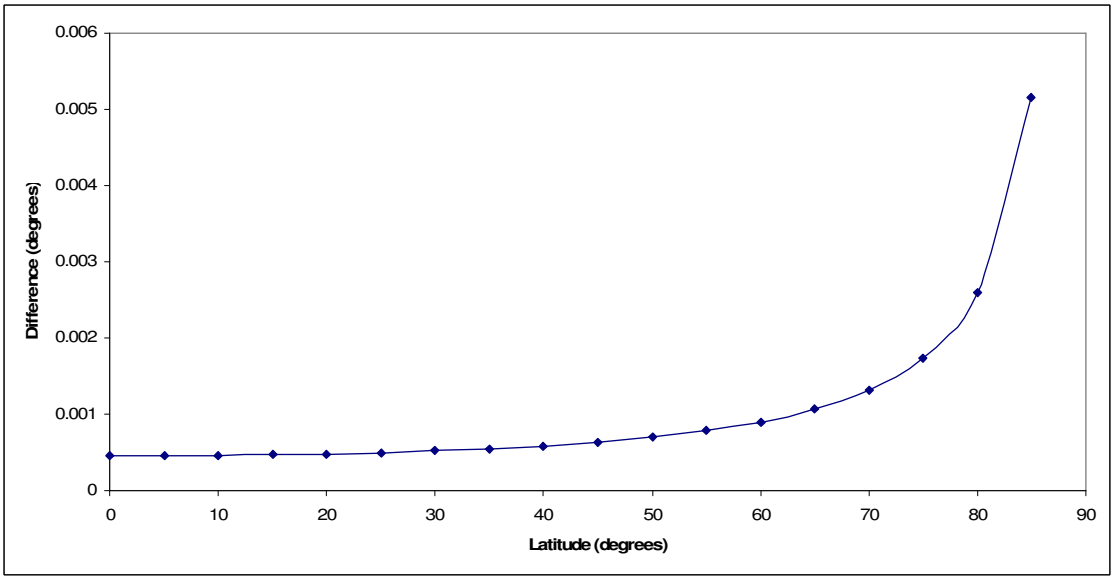

Figure 77: Precision required for 60m accuracy

The equator needs the most accuracy and this can be achieved by using fixed point numbers of 20bits in size. Figure 78 indicates the level of precision afforded by 20bits at varying latitudes, where the equator gives the least accuracy.

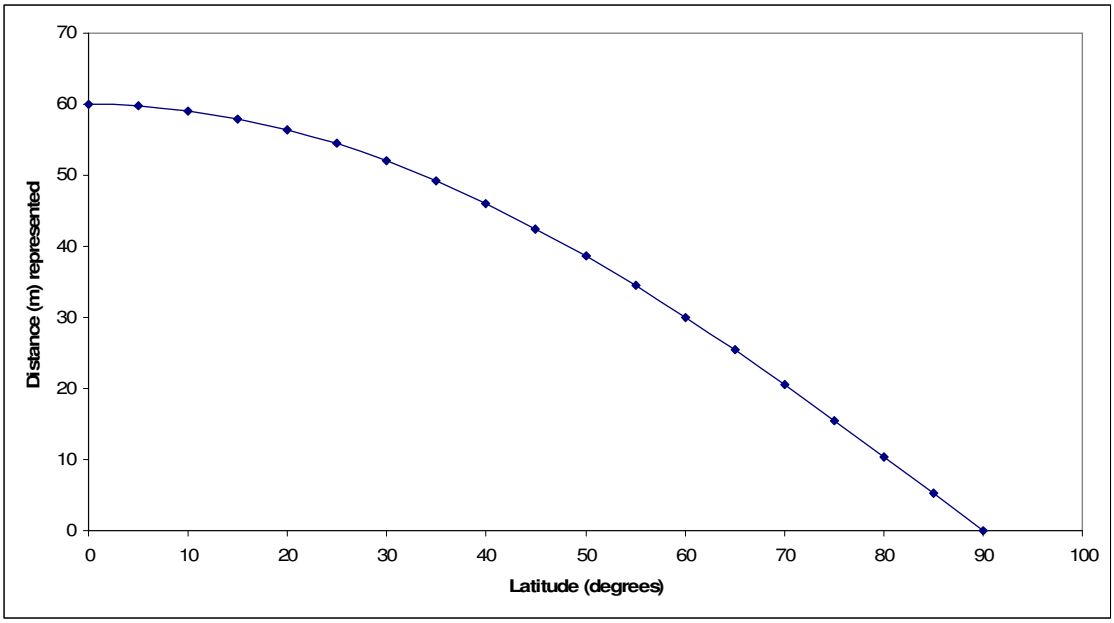

Figure 78: Accuracy provided by 20bit fixed point numbers

The latitudinal lines are equidistant apart regardless of location and so precision can be provided by 19bits, exactly half of that required by the longitudinal co-ordinate; however, to reduce computational complexity this is increased to 20bits to provide a set of co-ordinates 40bits in length that sit on a byte boundary.

Therefore, we can deduce the size of the packet to be 168 bits (21 bytes) as illustrated below with the list of agents added. The size of the agent ID list can be calculated using the packet size and deducting the size of the other fields.

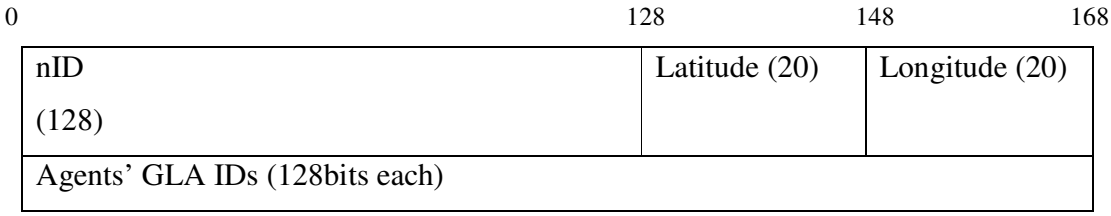

Figure 79: Beacon packet size

The easiest method to implement this beaconing is to simply have an application layer process that sends out periodic UDP broadcasts. This can therefore be implemented by

installation of a user application and requires no modification of the device's software communications stack.

6.3 Routing

Routing in large-scale ad hoc networks requires the use of geographical information to make decisions. The traditional implementation in TCP/IP is to use a routing table which maps IP address ranges to particular interfaces, or particular routing devices. An operating system kernel will maintain this table and on receiving a packet will lookup the destination in this table matching the destination IP address to the most suitable output interface. A network is defined by a range of IP addresses and the gateway is a remote routing device that knows a path to the network. When the network is attached to the local device then there is no gateway to connect to that network. The table below indicates a typical company intranet where the device is connected to a LAN, but has an entry for another of the company's subnets, routed through another device on the local interface.

Table 6: Example routing table

Network	Subnet mask	Gateway	Interface
192.168.0.0	255.255.255.0	-	Card 1
192.168.1.0	255.255.255.0	192.168.0.1	Card 1

The difficulty is in intercepting packets in the TCP/IP stack so that they can be buffered whilst a route or location is found for the destination. In addition, packets arriving at a node that does not know the next-hop will often be dropped. Linux allows the use of the netfilter interface to intercept packets and in Windows CE, a mobile operating system, the creation of a driver can intercept packets (West, 2003).

Each packet needs to carry extra information as described in 5.6.3 and this is achieved through the use of the IP options field, or by encapsulation of the packet.

6.3.1 SOLS

SOLS is an integral part of the routing protocol as it is used for discovering location information. The routing layer must be modified to support geographic routing and the interception of packets to unknown destinations as above; however, SOLS itself can be implemented at the application layer.

The advantage of implementing SOLS at the application layer is a generic application that can be easily ported to multiple platforms without relying on operating system specifics. If SOLS were to be implemented at the network layer then much of the code may need to be rewritten for the specific operating system rather than a generic Java implementation.

Sending packets from the application layer may require them to be sent using UDP which incurs extra overhead although is only a few bytes; however, many OSes may allow the sending of IP packets without UDP encapsulation from the application layer.

6.4 Conclusion

Implementation of SOLS and associated routing on most modern devices is possible with many providing driver hooks for packet interception. Some platforms may not support such integration into the network layer and will require manufacturer co-operation. Without this co-operation, it is still possible to implement SOLS routing at the application by requiring applications to use a proxy. Much of the SOLS algorithm can be implemented at the application layer to simplify multiple platform deployments.

Beacon packets were shown to be small and the beaconing scheme may also be implemented at the application to support easier deployment. For example, the beaconing application could be implemented in Java and would then require no modifications to be supported on platforms which have a Java runtime.

Overall implementation of SOLS can be easily achieved on most mainstream operating systems without difficulty.

7 Conclusions

The thesis proposed and examined a self organising location server (SOLS) based upon a novel concept. SOLS was shown to be fault and mobility tolerant unlike competing technologies, whilst still maintaining low overheads. When used as a location server, SOLS performed significantly better than the closest competitor in most scenarios and was able to provide delivery success where Terminode was not.

Implementation on many existing devices is entirely feasibly without modification of the existing operating source code. The use of an application layer program to either modify the routing table or to encapsulate the packet and forward it locally over say UDP is easily implemented on most mainstream operating systems.

Ad hoc networks will with little doubt become a leading technology in the next decade as more users demand ubiquitous internet access. In addition, wireless has the potential to significantly reduce cabling costs and is only limited by range and capacity. This work, along with others' on multi-hop ad hoc networks has addressed the issue of range and routing overhead, with the only challenge remaining being capacity.

The next section discusses future work for SOLS and for the field of large scale ad hoc networks.

7.1 Future work

7.1.1 SOLS specific

This subsection will discuss some improvements specific to SOLS. Firstly, SOLS employs only one GLA for each node wishing to store its location. Whilst this will work fine in smaller networks, when networks become larger and a node is potentially hundreds of kilometres from its GLA, then the update path will be long. To mitigate this, when a node moves a certain distance from its first GLA, it could create a second GLA that is closer and require the first GLA to refer nodes to the second. This would then result in the node having to send updates along a much shorter path.

SOLS consists on a one hop duplication scheme which provided good tolerance to failure in the simulations examined. However, in less dense networks a master may only have a few neighbours and so only create a few slaves increasing the likelihood of failure. Future work could examine the possibility of using a multiple hop duplication and monitoring scheme.

SOLS assumes that node failures cannot be predicted or known in advance as this is the worst case; however, many mobile phones and PDAs are able to determine time left until battery failure. This information could be used to hand-off agents to other nodes when a battery failure is eliminated thereby removing the need to have as many agents in the GLA. If this could be relied on all the time then only one agent would be needed, but as this is unlikely a minimum of two agents is prudent.

The adaptive failure threshold scheme described in section 4.4 assumed that all nodes would have the same failure rate. This is a reasonable expectation given that many phones and mobile devices have similar battery life and that the scheme would make no changes to the threshold when the failure rate is in terms of hours. however, for high failure systems it maybe necessary to investigate the situation where devices have vastly different failure rates.

The final results used an interval-based update scheme rather than the superior distance based scheme; however, because the speed was uniform amongst nodes the results are still applicable and transferable over to the distance based scheme. The situation that was not examined is where nodes are travelling at significantly different speeds and this is where the distance based scheme is expected to excel. Therefore, future work should examine the performance of the system with nodes having differing node speeds and using the distance based update scheme.

7.1.2 Internet augmentation

Nodes within an ad hoc network have limited communication bandwidth and whilst increasing the bandwidth available through new radio transmission techniques will offset these limitations, it will not cure the fundamental problem. Given a set of nodes distributed across a few square kilometres, who all communicate with other nodes at random, the nodes in the centre of the network will be responsible for forwarding more packets than those at the periphery. Assuming the nodes in use are mobile phones, the majority of phone users' usage is to those who are within proximity of the current city, although a small percentage of calls maybe long distance especially in the case where the user maybe away from his or her home city. Therefore, we can assume that given a country-wide network most of the communication would be localised. While the national or international traffic maybe small in comparison, with larger nodes in the network this begins to exhaust the capacity of nodes located centrally.

We are now beginning to examine the idea of replacing the cellular network with a large-scale ad hoc network; however, the increase in the number of homes who use wireless access point provides an excellent opportunity to make this a reality. Driving through a city one can pick up hundreds of wireless networks (Hal, 2004). We could use these wireless networks for routing of national and international traffic, or traffic that has to traverse a significant number of hops. Upon determining that a node is a certain distance away, one could opt to route traffic through the ad hoc network (if not within range) to the nearest access point, the traffic would then traverse the internet to the nearest access point to the destination, at which point it would exit the internet and back onto the ad hoc network to the destination. There are several problems that need to be examined for this to be a reality. These are how does one determine the nearest access point, and equally so, how does one determine the exit access point that is nearest to the destination. If these problems can be solved, possibly by use of an overlay network, then large-scale ad hoc networks can become a reality and ubiquitous internet accessibly almost anywhere would be possible.

7.2 Potential and current applications

The purpose of this section is to discuss potential applications of the work described in this thesis. Ad-hoc networks have wide applicability and were initially envisaged as being used in emergency disaster areas and military applications. In both of these scenarios ad hoc networks would be immensely useful due to the lack of communications infrastructure.

Where the work would be particularly important is when networks tend to stay in an area so that home locations can be permanently defined. Consider a village where Internet connectivity is not readily available to all homes but where a local company has obtained a high speed link through either fixed line or satellite. This company would wish to supply, for a fee, Internet access to villagers and installation of cabling would be prohibitively expensive. Initially one may consider that the company could host a location server for all nodes, but this will result in a single point of failure and high overhead if there are tens of thousands of villagers. Each villager would set up their own SOLS location server with their home location being their house. As many of the villagers may wish to talk/communicate with one another this avoids the need to route through the company and potentially avoids the fees. Villagers may then roam around the village and still be contactable by updating their home location regardless of whether they have a wireless device running inside their homes. The villagers then have an autonomous network that is independent of the company but where the company provides a service to the network users, and therefore also allows competition or expansion from other service providers. A similar scenario would work for university campuses, with staff's offices being their home location.

At the moment most customers pay a fee to a cellular telephone operator for their service and if a local cell tower fails then they lose the use of their phone. The use of the location server in this thesis, with a large ad hoc network, and augmentation from wireless access points could potentially replace a cellular network. Whilst undoubtedly there would be resistance from cellular operators, this would provide users with a resilient free alternative to making local calls. Telephone operators could change their business model

to augment the network, providing coverage and connectivity in rural areas and data services. With augmentation from the operators the network could become national or international and significantly reduce their costs in cellular infrastructure. Users would benefit from the free local calls whilst paying for national or calls to rural areas.

References

ABRAHAM, I., DOLEV, D. & MALKHI, D. (2004) LLS: a Locality Aware Location Service for Mobile Ad Hoc Networks. *Foundations of mobile computing; DIALM POMC 04.* Philadelphia, PA, Acm.

AMIR, Y. & WOOL, A. (1998) Optimal availability quorum systems: Theory and practice. *Information Processing Letters,* 65**,** 223-228.

BARBARA, D. & GARCIA-MOLINA, H. (1986) The vulnerability of vote assignments. *ACM Transactions on Computer Systems,* 4**,** 187-213.

BARBARA, D. & GARCIA-MOLINA, H. (1987) The reliability of vote mechanisms. *IEEE Transactions on Computers,* 36**,** 1197-1208.

BARR, R. (2004) An efficient, unifying approach to simulation using virtual machines., Cornell University.

BASAGNI, S., CHLAMTAC, I., SYROTIUK, V. R. & WOODWARD, B. A. (1998) A Distance Routing Effect Algorithm for Mobility (DREAM). *Mobile computing and networking; MobiCom `98.* Dallas, TX, New York.

BLAZEVIC, L., BUTTYAN, L., CAPKUN, S., GIORDANO, S., HUBAUX, J. P. & LE BOUDEC, J. Y. (2001) Self-Organization in Mobile Ad Hoc Networks: the Approach of Terminodes. *Ieee Communications Magazine,* 39**,** 166-175.

BLAZEVIC, L., LE BOUDEC, J. Y. & GIORDANO, S. (2002) A Scalable Routing Scheme for Self-Organized Terminode Network. IN ZNATI, T. & MCDONALD, B. (Eds.) *Communication networks and distributed systems modeling and simulation conference.* San Antonio, TX, Society for Computer Simulation International.

BOSE, P., MORIN, P., STOJMENOVIC, I. & URRUTIA, J. (2001) Routing with Guaranteed Delivery in Ad Hoc Wireless Networks. *Wireless Networks,* 7**,** 609-616.

CAMP, T., BOLENG, J. & WILCOX, L. (2002a) Location Information Services in Mobile Ad Hoc Networks. *IEEE International Conference on Communications.*

CAMP, T., BOLENG, J. & WILCOX, L. (2002b) Location Information Services in Mobile Ad Hoc Networks. *Ieee International Conference on Communications,* 5**,** 3318-3324.

CAMP, T., BOLENG, J., WILLIAMS, B., WILCOX, L. & NAVIDI, W. (2002c) Performance Comparison of Two Location Based Routing Protocols for Ad Hoc Networks. *Ieee Infocom,* 3**,** 1678-1687.

CAPKA, J. & BOUTABA, R. (2004) Mobility Prediction in Wireless Networks Using Neural Networks. IN VICENTE, J. & HUTCHISON, D. (Eds.) *Management of multimedia networks and services.* San Diego, CA, Berlin.

CARLSON, S. (2000) Simulating boids, floys and other artificial life. *Scientific American,* 283**,** 94-95.

CASTANEDA, R., DAS, S. R. & MARINA, M. K. (2002) Query Localization Techniques for On-Demand Routing Protocols in Ad Hoc Networks. *Wireless Networks,* 8**,** 137-152.

CHAWLA, M., GOEL, N., KALAICHELVAN, K., NAYAK, A. & STOJMENOVIC, I. (2006) Beaconless Position-based Routing with Guaranteed Delivery for Wireless Ad hoc and Sensor Networks. *ACTA Automatica Sinica,* 32.

CHENG, C. T., LEMBERG, H. L., PHILIP, S. J., VAN DEN BERG, E. & ZHANG, T. (2002) SLALoM: A Scalable Location Management Scheme for Large Mobile Ad Hoc Networks. *Wireless communications and networking conference.* Orlando, FL, Ieee.

CLAUSEN, T. & JACQUET, P. (2003) Optimized Link State Routing Protocol (OLSR). *Request for Comments: 3626.* Network Working Group.

DAS, S. M., PUCHA, H. & HU, Y. C. (2005) Performance Comparison of Scalable Location Services for Geographic Ad Hoc Routing. *Ieee Infocom,* 2**,** 1228-1239.

DATTA, S., STOJMENOVIC, I. & WU, J. (2001) Internal Node and Shortcut Based Routing with Guaranteed Delivery in Wireless Networks. IN TAKIZAWA, M. (Ed.) *International workshop on wireless networks and mobile computing; 21st International conference on distributed computing systems workshops.* Mesa, AZ, IEEE Computer Society.

DHILLON, S. S. & VAN MIEGHEM, P. (2007) Performance analysis of the AntNet algorithm. *Computer Networks,* 51**,** 2104-2125.

DI CARO, G. & DORIGO, M. (1999) AntNet: Distributed Stigmergetic Control for Communications Networks. *Vivek,* 12**,** 2-37.

DI CARO, G., DUCATELLE, F. & GAMBARDELLA, L. M. (2004) AntHocNet: An Ant-Based Hybrid Routing Algorithm for Mobile Ad Hoc Networks. *Lecture Notes in Computer Science***,** 461-470.

DOLEV, S., GILBERT, S., LAHIANI, L., LYNCH, N. & NOLTE, T. (2006) Timed Virtual Stationary Automata for Mobile Networks. *Lecture Notes in Computer Science***,** 130-145.

DOLEV, S., GILBERT, S., LYNCH, N. A., SHVARTSMAN, A. A. & WELCH, J. L. (2005a) GeoQuorums: implementing atomic memory in mobile ad hoc networks. *Distributed Computing,* 18**,** 125-155.

DOLEV, S., GILBERT, S., SCHILLER, E., SHVARTSMAN, A. & WELCH, J. (2005b) Autonomous Virtual Mobile Nodes. *Parallelism in algorithms and architectures; SPAA 2005.* Las Vegas, NV, ACM Press.

DUCATELLE, F., DI CARO, G. A. & GAMBARDELLA, L. M. (2006) An Analysis of the Different Components of the AntHocNet Routing Algorithm. *Lecture Notes in Computer Science***,** 37-48.

DYNALINK (2004) Wireless LAN Card Specifications.

EBERHART, R. C., SHI, Y. & KENNEDY, J. (2002) *Swarm Intelligence*, Morgan Kaufmann.

ERCIYES, K. & MARSHALL, G. (2004) A Cluster Based Hierarchical Routing Protocol for Mobile Networks. IN LAGANA, A. (Ed.) *International Conference on Computational Science and its Applications.* Assisi, Italy, Berlin.

FEHNKER, A. & GAO, P. (2006) Formal Verification and Simulation for Performance Analysis for Probabilistic Broadcast Protocols. *Lecture Notes in Computer Science*, 128-141.

FRANKLIN, S. & GRAESSER, A. (1997) Is it an Agent, or Just a Program?: A Taxonomy for Autonomous Agents. *Lecture Notes in Computer Science*, 21-36.

FUßLER, H., WIDMER, J., KASEMANN, M., MAUVE, M. & HARTENSTEIN, H. (2001) Beaconless Position-Based Routing for Mobile Ad-Hoc Networks. REIHE INFORMATIK.

GAO, J., BUIBAS, L. J., HERSHBERGER, J., ZHANG, L. & ZHU, A. (2001) Geometric Spanner for Routing in Mobile Networks. *Mobile ad hoc networking & computing*. Long Beach, CA, Ieee.

GATES, B. (1995) *The Road Ahead*, Penguin.

GAVALAS, D., PANTZIOU, G., KONSTANTOPOULOS, C. & MAMALIS, B. (2006) Stable and Energy Efficient Clustering of Wireless Ad-Hoc Networks with LIDAR Algorithm. IN CUENCA CASTILLO, P. A. & OROZCO-BARBOSA, L. (Eds.) *Personal wireless communications; PWC 2006.* Albacete, Spain, Berlin.

GEDIK, B. & LIU, L. (2005) Location Privacy in Mobile Systems: A Personalized Anonymization Model. *International Conference on Distributed Computing Systems,* 25, 620-632.

GIORDANO, S. & HAMDI, M. (1999) Mobility Management: The Virtual Home Region. EPFL, Lausanne, Switzerland.

GUNES, M., SORGES, U. & BOUAZIZI, I. (2002) ARA - The Ant-Colony Based Routing Algorithm for MANETs. IN OLARIU, S. (Ed.) *International conference on parallel processing.* Vancouver, Canada, IEEE Computer Society.

HAAS, Z. (1997) A new routing protocol for reconfigurable wireless networks. *IEEE International Conference of Universal Perional Communications.*

HAL, B. (2004) Wireless infidelity I: war driving. *Commun. ACM,* 47, 21-26.

HEDRICK, C. (1988) Routing Information Protocol. *Request for Comments 1088.*

HOU, T.-C. & LI, V. (1986) Transmission range control in multihop packet radio networks. *IEEE Transactions on Communications,* 34, 38-44.

IETF (1997) DHCP: Dynamic Host Configuration Protocol (RFC 2131).

ISKANDER, M. F. & YUN, Z. (2002) Propagation Prediction Models for Wireless Communication Systems. *Ieee Transactions on Microwave Theory and Techniques,* 50, 662-673.

JADBABAIE, A. (2004) On Geographic Routing without Location Information. *Decision and control; CDC 43.* Nassau, Bahamas, Ieee.

JAIN, R., PURI, A. & SENGUPTA, R. (2001) Geographical Routing Using Partial Information for Wireless Ad Hoc Networks. *Ieee Personal Communications,* 8, 48-57.

JOHANSSON, P., LARSSON, T., HEDMAN, N., MIELCZAREK, B. & MDEGERMARK, M. (1999) Scenario-based Performance Analysis of Routing Protocols for Mobile Ad-hoc Networks. *5th annual ACM/IEEE international conference of Mobile computing and networking.* ACM Press.

JOHNSON, S. (2001) *Emergence: The Connected Lives of Ants, Brains, Cities and Software*, Scribner.

KARP, B. & KUNG, H. T. (2000) GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. *Mobile computing and networking; MobiCom 2000.* Boston, MA, New York.

KARUMANCHI, G., MURALIDHARAN, S. & PRAKASH, R. (1999) Information Dissemination in Partitionable Mobile Ad Hoc Networks. *Reliable distributed systems.* Lausanne, Switzerland, IEEE Computer Society.

KHAMSI, R. (2004) Skype Beyond the Hype. *Technology Review,* 107**,** 44-47.

KRISHNAMACHARI, B., WICKER, S. B., BEJAR, R. & PEARLMAN, M. (2003) Critical Density Thresholds in Distributed Wireless Networks. *Kluwer International Series in Engineering and Computer Science***,** 279-296.

KUHN, F., WATTENHOFER, R., ZHANG, Y. & ZOLLINGER, A. (2003a) Geometric Ad-Hoc Routing: Of Theory and Practice. *Principles of distributed computing; PODC 2003.* Boston, MA, ACM Press.

KUHN, F., WATTENHOFER, R. & ZOLLINGER, A. (2003b) Worst-Case Optimal and Average-Case Efficient Geometric Ad-Hoc Routing. *Mobile ad hoc networking & computing; MobiHoc 2003.* Annapolis, MD, Acm.

KWON, S., PARK, H. & LEE, K. (2005) A Novel Mobility Prediction Algorithm Based on User Movement History in Wireless Networks. *Lecture Notes in Computer Science***,** 419-428.

LAMPORT, L., SHOSTAK, R. & PEASE, M. (1982) Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems,* 4.

LEE, S. J., BELDING-ROYER, E. M. & PERKINS, C. E. (2003) Scalability study of the ad hoc on-demand distance vector routing protocol. *International Journal of Network Management,* 13**,** 97-114.

LI, J., JANNOTTI, J., DE COUTO, D. S. J., KARGER, D. R. & MORRIS, R. (2000) A Scalable Location Service for Geographic Ad Hoc Routing. *6th ACM International Conference on Mobile Computing and Networking (MobiCom).*

LIU, D., STOJMENOVIC, I. & JIA, X. (2006) A Scalable Quorum Based Location Service in Ad Hoc and Sensor Networks. *IEEE International Conference on Mobile Ad-hoc and Sensor Systems MASS.* Vancouver.

LOCHERT, C., HARTENSTEIN, H., TIAN, J., FUßLER, H., HERMANN, D. & MAUVE, M. (2003) A routing strategy for vehicular ad hoc networks in city environments. *IEEE Intelligent Vehicles Symposium***,** 156-161.

LUO, J., EUGSTER, P. T. & HUBAUX, J. P. (2004) Pilot: Probabilistic Lightweight Group Communication System for Ad Hoc Networks. *Ieee Transactions on Mobile Computing,* 3**,** 164-179.

MACGOUGAN, G., LACHAPELLE, G., KLUKAS, R., SIU, K., GARIN, L., SHEWFELT, J. & COX, G. (2002) Performance analysis of a stand-alone high-sensitivity receiver. *GPS Solutions,* 6**,** 179-195.

MARSHALL, G. & ERCIYES, K. (2005) Implementation of a Cluster Based Routing Protocol for Mobile Networks. *Lecture Notes in Computer Science***,** 388-395.

MAUVE, M., WIDMER, J. & HARTENSTEIN, H. (2001) A Survey on Position-Based Routing in Mobile Ad Hoc Networks. *Ieee Network,* 15**,** 30-39.

MCQUILLAN, J. M., RICHER, I. & ROSEN, E. C. (1980) The Net Routing Algorithm for the ARPANet. *IEEE Transactions of Communications,* 28**,** 711-719.

MOROWITZ, H. J. (2003) *Emergences : Twenty-Eight Steps from the Beginning to the Human Spirit*, Oxford University Press.

NAOR, M. & WOOL, A. (1998) The load, capacity and availability of quorum systems. *SIAM Journal of Computing,* 27**,** 423-447.

NAOUMOV, V. & GROSS, T. (2003) Simulation of Large Ad Hoc Networks. *Modeling analysis and simulation of wireless and mobile systems; ACM MSWIM 2003.* San Diego, CA, Acm.

NICULESCU, D. & NATH, B. (2001) Ad Hoc Positioning System (APS). *Globecom,* 5**,** 2926-2931.

NICULESCU, D. & NATH, B. (2004) Error Characteristics of Ad Hoc Positioning Systems (APS). *Mobile ad hoc networking and computing; Mobihoc 2004.* Tokyo, Acm.

ONS (2002) General Household Survey. Office of National Statistics.

OWEN, G. H. & ADDA, M. (2005) Feasibility of geographically static data storage in ad-hoc networks. IN IADAT (Ed.) *2nd International Conference on Telecommunications and Computer Networks.* Portsmouth.

OWEN, G. H. & ADDA, M. (2006a) Geographically static quorums in ad-hoc networks and their performance as location servers. IN IADAT (Ed.) *3rd International conference on Telecommunications and Computer networks.* Portsmouth, UK.

OWEN, G. H. & ADDA, M. (2006b) Quorum based geographically static data storage in ad-hoc networks. *International Network Conference.* Plymouth, UK.

OWEN, G. H. & ADDA, M. (2006c) Self organizing quorum systems for ad hoc networks. IN IASTED (Ed.) *International Conference on Communication, Network, and Information Security.* MIT Faculty Club, Cambridge, Massachusetts, USA, IASTED.

OWEN, G. H. & ADDA, M. (2006d) Simulation of quorum systems in ad hoc networks. *3rd International Conference on Artificial Intelligence in Engineering and Technology.* Malaysia.

OWEN, G. H. & ADDA, M. (2007) A Fault and Mobility Tolerant Location Server for Large-scale Ad-hoc networks. *IST Mobile and Wireless Communications Summit.* Budapest, Hungary.

PARKINSON, B. W. & SPILKER, J. J. (1996) *Global Positioning System: theory and applications*, Aiaa.

PELEG, D. & WOOL, A. (1995) The availability of quorum systems. *Information and Computation,* 123**,** 210-233.

PERKINS, C. & ROYER, E. M. (1999) Ad-hoc on-demand distance vector routing. *2nd IEEE workshop on Mobile Computing Systems and Applications.*

PERKINS, C. E., ROYER, E. M., DAS, S. R. & MARINA, M. K. (2001) Performance Comparison of Two On-Demand Routing Protocols for Ad Hoc Networks. *Ieee Personal Communications,* 8**,** 16-29.

PHILIP, S. J., GHOSH, J., KHEDEKAR, S. & QIAO, C. (2004) Scalability Analysis of Location Management Protocols for Mobile Ad Hoc Networks. *Wireless communications and networking conference; Broadband wireless: the time is now.* Atlanta, GA, Institute of Electrical and Electronics Engineers.

PHILIP, S. J. & QIAO, C. (2003) ELF: Efficient Location Forwarding for Ad Hoc Networks. *Globecom,* 2**,** 913-918.

POP, P. C., PINTEA, C. M. & SITAR, C. P. (2007) An Ant-Based Heuristic for the Railway Traveling Salesman Problem. IN GIACOBINI, M. (Ed.) *Applications of evolutionary computing; EvoWorkshops 2007: EvoCOMNET, EvoFIN, EvoIASP, EvoINTERACTION, EvoMUSART, EvoSTOC and EvoTRANSLOG.* Valencia, Spain, Berlin.

PRIYANTHA, N., CHAKRABORTY, A. & BALAKRISHNAN, H. (2000) The Cricket Location-Support System. *Mobile computing and networking; MobiCom 2000.* Boston, MA, New York.

PURIS, A., BELLO, R., MARTINEZ, Y. & NOWE, A. (2007) Two-Stage Ant Colony Optimization for Solving the Traveling Salesman Problem. *Lecture Notes in Computer Science***,** 307-316.

RAJAGOPALAN, S. & SHEN, C. C. (2006) ANSI: A swarm intelligence-based unicast routing protocol for hybrid ad hoc networks. *Journal of Systems Architecture,* 52**,** 485-504.

RAO, A., RATNASAMY, S., PAPADIMITRIOU, C., SHENKER, S. & STOICA, I. (2003) Geographic Routing without Location Information. *Mobile computing and networking; MobiCom 2003.* San Diego, CA, New York.

REYNOLDS, C. W. (1987) Flocks, Herds, and Schools: A Distributed Behavioral Model, in Computer Graphics. *SIGGRAPH.*

REZAIIFAR, R. & MAKOWSKI, A. M. (1997) From Optimal Search Theory to Sequential Paging in Cellular Networks. *Ieee Journal on Selected Areas in Communications Sac,* 15**,** 1253-1264.

ROJAS, A., BRANCH, P. & ARMITAGE, G. (2005) Experimental Validation of the Random Waypoint Mobility Model through a Real World Mobility Trace for Large Geographical Areas. IN BOUKERCHE, A., LEUNG, V., CHIASSERINI, C. F. & SRINIVASAN, V. (Eds.) *ACM Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems; ACM MSWiM 2005.* Montreal, Quebec, Canada, New York N.Y.:.

ROYER, E. M. & TOH, C.-K. (1999) A review of current routing protocols for ad-hoc mobile wireless networks. *IEEE Personal Communications Magazine***,** 46-55.

RYU, J. P., KIM, M. S., HWANG, S. H. & HAN, K. J. (2004) An Adaptive Probabilistic Broadcast Scheme for Ad-Hoc Networks. IN MAMMERI, Z. & LORENZ, P. (Eds.) *High speed networks and multimedia communications: 7th IEEE International Conference, HSNMC 2004, Toulouse, France, June 30-July 2, 2004, proceedings /.* Toulouse, France, Berlin.

SEET, B.-C., PAN, Y., HSU, W.-J. & LAU, C.-T. (2005) Multi-Home Region Location Service for Wireless Ad Hoc Networks: An Adaptive Demand-Driven Approach. *Second Annual Conference on Wireless On-demand Network Systems and Services, 2005. WONS 2005.*

SHORT, M. (1999) The success of GSM: How the GSM industry needs to progress in the run-up to the introduction of Third Generation services. *Mobile Communications International***,** 86-87.

SIVAVAKEESAR, S. & PAVLOU, G. (2004) Cluster-based Location-Services for Scalable Ad Hoc Network Routing. IN BELDING-ROYER, E. M., AL AGHA, K. & PUJOLLE, G. (Eds.) *IFIP TC6/WG6.8 Conference on Mobile and Wireless Communication Networks; (MWCN 2004).* Paris, France, New York.

SMITH, A., BALAKRISHNAN, H., GORACZKO, M. & PRIYANTHA, N. (2004) Tracking Moving Devices with the Cricket Location System. *Mobile systems, applications and services; MobiSys 2004.* Boston, Mass, Acm.

STOJMENOVIC, I. (1999) A scalable quorum based location update scheme for routing in ad hoc wireless networks. University of Ottawa.

SWEENEY, D. (2001) An Introduction to Bluetooth: A Standard for Short-range Wireless Networking. IN MUKUND, P. R., CHICKANOSKY, J. & KRISHNAMURTHY, R. K. (Eds.) *Annual IEEE international ASIC/SOC conference.* Arlington, VA, Ieee.

TEKINER, F. (2004) The Antnet Routing Algorithm - A Modified Version. IN DLAY, S. S. (Ed.) *Communication systems, networks and digital signal processing; CSNDSP 2004.* Newcastle, School of Electrical Electronic and Computer Engineering The University of Newcastle upon Tyne.

THOMAS, R., GILBERT, H. & MAZZIOTTO, G. (1988) Influence of the Moving of the Mobile Stations on the performance of a Radio Cellular Netwrk. *Third Nordic Semnar.*

THOPPIAN, M. R. & PRAKASH, R. (2006) A Distributed Protocol for Dynamic Address Assignment in Mobile Ad Hoc Networks. *Ieee Transactions on Mobile Computing,* 5**,** 4-19.

TSENG, Y. C., LIAO, W. H. & WU, S. L. (2002a) Mobile ad hoc networks and routing protocols. IN STOJMENOVIC, I. (Ed.) *Handbook of Wireless Networks and Mobile Computing.* John Wiley & Sons.

TSENG, Y. C., NI, S. Y., CHEN, Y. S. & SHEU, J. P. (2002b) The Broadcast Storm Problem in a Mobile Ad Hoc Network. *Wireless Networks,* 8**,** 153-168.

TUMMOLINI, L. & CASTELFRANCHI, C. (2007) Trace Signals: The Meanings of Stigmergy. *Lecture Notes in Computer Science***,** 141-156.

WEST, D. (2003) An Implementation and Evaluation of the Ad-Hoc On-Demand Distance Vector Routing Protocol for Windows CE. *Computer Science.* University of Dublin.

WOLFGANG, K., HOLGER, F., LER, J, RG, W. & MARTIN, M. (2004) Hierarchical location service for mobile ad-hoc networks. *SIGMOBILE Mob. Comput. Commun. Rev.,* 8**,** 47-58.

WOO, S. C. M. & SINGH, S. (2001) Scalable Routing Protocol for Ad Hoc Networks. *Wireless Networks,* 7**,** 513-530.

WU, X. (2005) VPDS: Virtual Home Region Based Distributed Position Service in Mobile Ad Hoc Networks. *25th IEEE international conference on distributed computing systems: ICDCS 2005.* Columbus, OH, Ieee.

YOON, J., LIU, M. & NOBLE, B. (2003) Random Waypoint Considered Harmful. *Computer communications; IEEE INFOCOM 2003.* San Francisco, CA, Ieee.

YUAN, P., CHEN, H. & WANG, G. X. (2006) Particle Swarm Optimization for Routing Design in Ad hoc Networks. *Minimicro Systems,* 27**,** 1193-1196.

YUSIONG, J. P. T. & NAVAL, P. C. (2006) Training Neural Networks Using Multiobjective Particle Swarm Optimization. *Lecture Notes in Computer Science***,** 879-888.

ZENG, X., BAGRODIA, R. & GERLA, M. (1998) GloMoSim: a library for parallel simulation of large-scale wireless networks. *12th Workshop of Parallel and Distributed Simulations - PADS '98.* UCLA Parallel Computing Laboratory.

ZHANG, X. H. & XU, W. B. (2006) QoS Based Routing in Wireless Sensor Network with Particle Swarm Optimization. IN SHI, Z. & SADANANDA, R. (Eds.) *Agent computing and multi-agent systems; PRIMA 2006.* Guilin, China, Berlin.

ZHAO, Y. (2002) Standardization of Mobile Phone Positioning for 3g Systems. *Ieee Communications Magazine,* 40**,** 108-117.

ZHENG, X., GE, L., GUO, W. & LIU, R. (2007) A cross-layer design and ant-colony optimization based load-balancing routing protocol for ad-hoc networks. *Frontiers of Electrical and Electronic Engineering in China,* 2**,** 219-229.

ZHOU, B., LEE, Y. Z., GERLA, M. & DE RANGO, F. (2006) Geo-LANMAR: a scalable routing protocol for ad hoc networks with group motion. *Wireless Communications and Mobile Computing,* 6**,** 989-1002.

8 Appendices

8.1 Terminology and notation

This subchapter will describe the terminology and notation used throughout the thesis. A number of other terms and notation will be used but these will be described when the terms are introduced.

Time: Time is represented by t and a number of variables will be denoted as a function of time to indicate that their value changes throughout the simulation.

Node: A node is a network device equipped with a wireless radio. The node is carried by a human being and is therefore mobile. The set of all nodes is designated by N, and a single node is designated as $n \in N$. A node has a unique identifier n_{ID}, and a co-ordinate (n_x, n_y).

Neighbour: A neighbour is described from the point of view of a node. A neighbour of a node is any other node which is in communication range.

Agent: An Agent is a software program, with a data payload, that can act independently and move from node to node. The agent contains data on the location of the node which created it and can be queried by any other node.

GLA: A Group of Location Agents that all contain information on the same node. The ID of a GLA is that of the node which created the first member.

<mylist>: A list of items.

|A|: The number of items in set A.

8.2 Simulation environment

8.2.1 Introduction

Simulations are used throughout the thesis to analyse the performance of various aspects of the proposal and to determine the best practice before moving forward. Simulation is widely used in the ad-hoc networks research community due to the high complexity of modelling such scenarios mathematically (Naoumov and Gross, 2003). Additionally it is prohibitively expensive to provide real world analysis and so simulation is a feasible alternative.

When designing a simulation environment one must consider the real world expected application of any technique. In this thesis, it is assumed that the technique will be used in a large-scale consumer network such as replacing the cellular network or large ad hoc WiFi networks in cities. Therefore, most nodes will be phones, PDAs and laptops carried in pockets of their owners and so will be mobile. If we were to consider a network in a city it could feasibly contain millions of nodes, which is not possible to simulate due to memory and computation limitations.

8.2.2 Simulator

A number of simulators exist for wireless networks such as DARPA ns-2, Jist/SWANS (Barr, 2004) and Glomosim (Zeng et al., 1998). Each provides an approximation of a wireless ad hoc network simulating radio frequency propagation, MAC protocols, node mobility, etc. Initially, much of the algorithm was developed in Glomosim due to its ability to process larger numbers of nodes than similar products. Unfortunately, due to lack of garbage collection the use of the simulater involved significant development times. Therefore, Jist/SWANS, a Java-based simulator with automatic garbage collection was used for collection of all the results in the thesis; however, the results were compared with output from the existing Glomosim modifications to confirm the validity of the simulation.

The ad-hoc network simulators require significant development time for algorithms due to the need to use simulator API calls and incorporation into the simulator methodology. Therefore, their use for initial prototyping was not desirable due to the considerable development time. As a result, an in-house simulator was developed to quickly prototype ideas and give indication as to their prospects. This simulator was a discrete time step Java simulator that did not consider RF interference or MAC layer protocols. Each node was represented by a Java class and the simulator cycled repeatedly through all nodes asking them to perform any tasks they had. The simulator calculated radio communication probability and approximated mobility by the use of the Random Waypoint Model. This simulator was used for prototyping and all results presented in the thesis use Jist/SWANS so as to more closely approximate reality. The results produced by the in-house simulator served as indication only to the expected outcome of the final simulations.

Jist/SWANS is a Java simulator designed specifically for fast simulation times of larger networks. Earlier simulators such as Glomosim and ns-2 are slow when simulating networks of several hundred nodes and due to the lack of garbage collection memory leaks are frequent problem in slowing development (Zeng et al., 1998). Jist/SWANS implements a realistic MAC layer and predictive radio propagation based upon the two-ray model (Iskander and Yun, 2002). The simulator therefore provides a close approximation to a real world ad hoc network and is appropriate for use in this thesis.

The algorithm is implemented at the network layer and the code introduced into Jist/SWANS entirely replaces any other routing algorithm. As discussed in earlier sections, certain parts of the technique could be implemented at the application layer although this is immaterial in simulation.

8.2.3 Simulation Parameters

As the technique proposed is not influenced by network size, simulation of small numbers of nodes will be adequate. As such, a network of 200 nodes in a one kilometre-squared area is used, approximating that of a town centre.

Each node moves and so one must determine how the node moves so that it maybe simulated. The Random Waypoint Model (Johnson, 2001) is almost uniformly used in the research literature due to its simplicity and approximation of human movement. Although there are a number of more complex models, each is specific to a certain situation and as it is impossible to determine exactly the movement of nodes, the random waypoint model is a close approximation (Rojas et al., 2005). In a town centre, the node speeds will consist of those walking (1-5km/hr), cycling (20-30km/hr) and travelling by motor vehicle (30-40km/hr). Typically in the literature a speed between 1m/s and 10m/s is chosen and this suits the needs of this thesis covering all three of the possible scenarios (2-36km/hr). When the speed is random, it is important that the minimum is not set to zero as all nodes will converge to slow long walks (Yoon et al., 2003) and therefore the minimum speed is set at one metre per second rather than zero.

Finally, one needs to discuss the radio range of nodes as this is one of the key factors in ad hoc network performance. Typically WiFi cards available today (Sept 2007) can offer up to 350m outdoors and 100m indoors (Dynalink, 2004). Using a radio range of 100m provides near complete connectivity with the network of the parameters described above (Krishnamachari et al., 2003), with the exception of the occasional node who is travelling near the periphery. Therefore, choosing 100m provides connectivity and is a realistic expectation of real-world scenarios.

Table 7: Simulation parameters

Area	1000 x 1000 m	**Simulation time**	3 minutes
Transmission Radius	100m	**Bidirectional flows (allocated randomly)**	5

Number of nodes	200	**Simulation repeated for verification**	10 times
Mobility Model	Random Waypoint *v = 1 – 10m/s;* *Pause time = 0s.*	**Number of agents used for SOLS (master + slaves)**	6
Propagation model	Two-ray	**Terminode *R* value**	100m
Beacon rate	1 second	**Location Server Update interval**	10 seconds

A number of other parameters in Table 7 will be described when used. Whilst a number of these parameters will be varied between simulations to determine their effect, if a parameter is not specified it is assumed to be set as described above.

8.2.4 Presentation of results

Most of the results presented in this thesis are presented in scatter graph form with lines of best fit to demonstrate trends. Ten simulations are performed of any combination of parameters to ensure validity of the results. The result of these simulations is then presented on the graph as one point. A line of best-fit consists of several points and so each line will be a result of several tens of simulations with each set of ten having different parameters.

When measuring against time it is often possible to obtain enough data from a single simulation to plot a line; however, in all the scenarios in this thesis no measure is taken against time. It is not possible to vary the failure or speed of nodes during a single simulation without providing inaccurate results or significantly increased simulation complexity. These are the reasons that each possible set of parameters is simulated as a one simulation and repeated for validity.

Lines of best-fit are either Linear or Polynomial (with varying orders) depending on the one with the highest R^2 value. Each graph indicates the type of best-fit used in the legend.

8.3 Case for beaconing

8.3.1 Introduction

The purpose of this subsection is to outline beaconing and to justify it over beaconless schemes. Beaconing is a process whereby nodes periodically broadcast information about themselves to their neighbours. In geographical routing, as a minimum this contains the node identifier and its geographical or relative position. By beaconing in this fashion nodes can make decisions on where to route packets using information on the whereabouts of their neighbours; however, the rate at which nodes beacon can affect the network performance and also the accuracy of information stored at neighbours.

$$B_{n,t} = (n_{ID}, n_x(t), n_y(t)) \tag{28}$$

When forwarding a packet there are several delays involved between sender and receiver. Assuming the path is known and there is no discovery delay the delay from sender to receiver can be described as follows. The first is the processing delay where each node in the path is deciding where to forward the packet and is typically of the order of microseconds, and here is considered negligible. The second and most significant delay is the retransmission delay whereby the time taken to transmit the packet is added at each hop. Finally, the propagation delay is the delay incurred in transmitting the packet through the air or medium. This is usually near the speed of light.

The transmission time, *t(S,D),* between two nodes, *S* and *D*, in an ad-hoc network can be calculated as a function of the number of hops (*h*), the packet size (*L*), the transmission rate (*R*), the processing delay at each hop (*p*), the distance travelled (*d*) and the speed of light (*c*) as follows:

$$t(S,D) = h\left(\frac{L}{R} + p\right) + \frac{d}{c} \tag{29}$$

The above scheme requires that we know the path and in geographical forwarding that each hop knows the location of is neighbours, through a mechanism such as beaconing. Beaconing can be undesirable as it requires that nodes periodically transmit packets, which while not having a significant effect on throughput when the beacon rate is low, does prevent nodes entering sleep mode to conserve power. It is worth considering that beaconing need not be performed by the main processing unit it could be implemented in hardware thereby negating the sleep disadvantage. Figure 80 illustrates the effect beaconing has on throughput on a single link between nodes, with a link capacity of 2Mbps and a beacon packet size of 100bytes, with both nodes beaconing. The graph shows negligible effect, but if the two schemes are to be compared accurately then the reduction in link capacity must be established. Averaging the reduction up until the link reaches saturation results in a reduction in capacity of 0.096kbps.

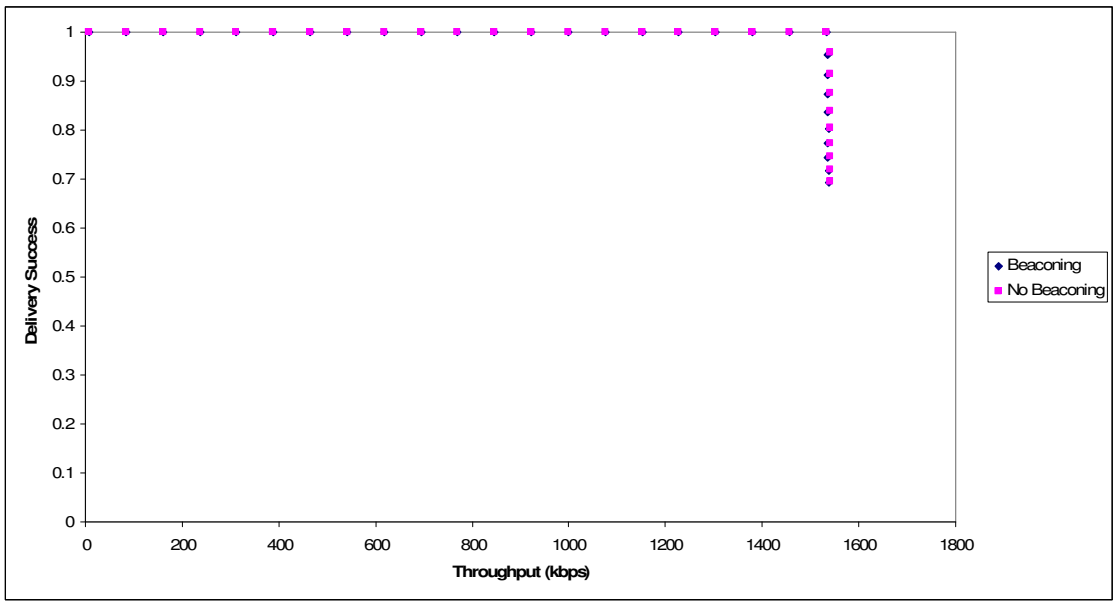

Figure 80: Effect of beaconing on a single link

Beaconless schemes (Fußler et al., 2001) were proposed to eliminate the need for nodes to beacon thereby allowing them to enter sleep mode. The proposed solution is that a

node wishing to forward a packet simply broadcasts it. Each node receiving the packet estimates its distance to the destination node of the packet. This delay is such that closer nodes to the destination choose smaller delays, whilst those that are further away choose longer delays. After the delay the node broadcasts a packet indicating that it will forward the packet. The nodes whose delays were longer will receive the packet and this will cause them to discard the original data packet. The net effect is that the node closest to the destination will broadcast a forward inhibit message first and then proceed to forward the message; therefore sending the packet as close to the destination at each hop as possible. The disadvantage of this technique, other than incurring extra delay, is that not all potential forwarding nodes will receive the forward inhibit packet, and so a number of duplicate packets may be forwarded at each hop in the network. This in turn can increase the load of the network and in the original work it was indicated that each packet could have a number of duplicates reaching the destination.

This extra delay is indicated by adding it to the delay as a processing delay where D_x/D_y are the location of the destination, n_x/n_y are the location of the neighbour and T is the maximum delay such that $d(n,D) \in [0,T]$. It is assumed the maximum delay is set at 45ms in accordance with the experiments carried out by Fußler.

$$d(n,D) = T\left(1 - \frac{\sqrt{(D_x - n_x)^2 + (D_y - n_y)^2}}{\sqrt{D_x{}^2 + D_y{}^2}}\right) \tag{30}$$

The node density will have an effect on the delay at each hop as in more dense networks there is a higher probability of a node being closer to the extreme ranges of the broadcasting node's transmitter. This node, being closer, will draw a smaller delay than a more distant node.

8.3.2 Comparison of beaconing and beaconless

Fußler carried out experiments to measure the average delay in the beaconless scheme experienced at each hop. When the node density is four, a delay of $0.45T$ was found, and

when the node density is 16 nodes, a delay of $0.22T$ is found. A link capacity is chosen of 2Mbps with the appropriate reduction in capacity included for the beaconing scheme. Nodes have a distance of 50m between them and signals travel at the speed of light.

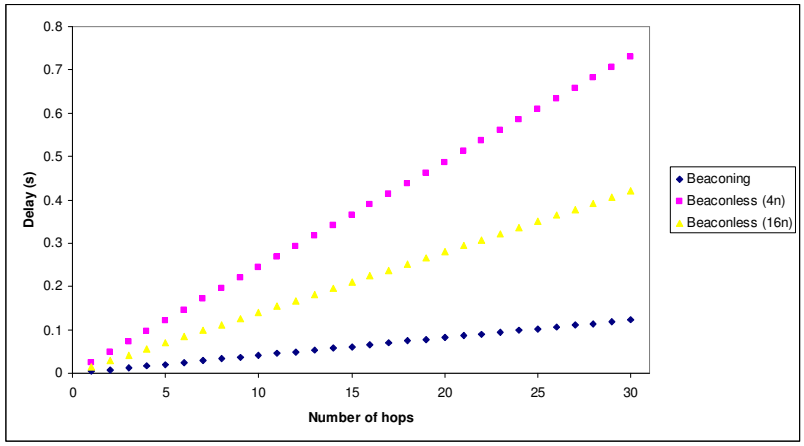

Figure 81: Delay in beaconing and beaconless routing schemes

Figure 81 shows a theoretical comparison of a beaconing scheme according to the equation and the beaconless scheme with varying node densities described in the earlier equation. As one can see the delay is significantly less than that of the beaconless scheme which causes delays that would be noticeable using video or voice after only 10-20 hops, far less than those expected in large networks.

If using a beaconless scheme is not acceptable for large scale ad hoc networks, then one must examine how often beaconing will be required to provide high delivery success. An examination is performed in Jist/SWANS with a network of 200 nodes in a 1000x1000m area, and a transmission radius of 100m. The node speed is varied so that the effect this has on the required beacon rate can be examined. When nodes are moving more quickly it is important nodes beacon more frequently to allow their neighbours to make sensible routing decisions.

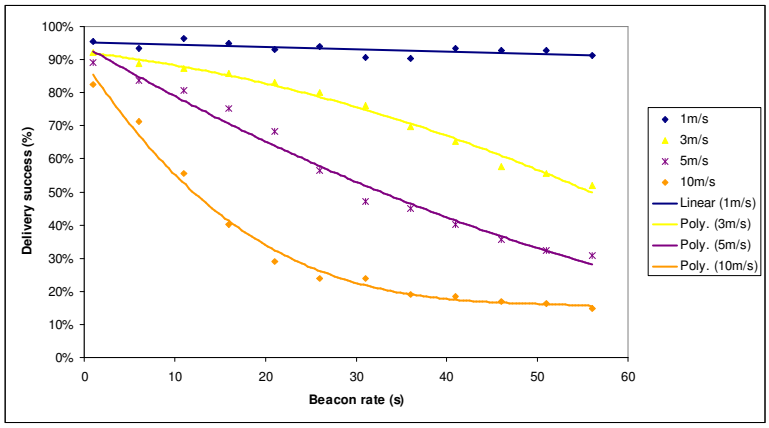

Figure 82: MFR routing success vs. beacon rate

Figure 82 shows the results on delivery success when the beacon rate was varied. With low mobility beacons at a rate as low as one per minute suffice but when node speeds become similar to that of city traffic at 10m/s, then beaconing at one packet per second becomes inadequate to maintain greater than 90% delivery success.

It is important to determine the effect that beacon rate will have on the capacity of the network. To test this scenario a simulation of a network of 200 nodes moving according to the Random Waypoint Model (*v=1* to *10m/s*; *0* pause time). Each node is assigned another at random to send constant bit-rate traffic to and the aggregate traffic of all flows is indicated in the graph.

Figure 83 shows the results obtained from simulation and the beacon rates have been chosen arbitrarily to ease implementation and have no special meaning. Beacon rates as low as 51.2 second had no measurable effect on throughput and decreasing the rate to 0.4 seconds had only a marginal effect. Increasing the rate by an order of magnitude to 0.04 seconds had a significant effect on throughput reducing packets received to 70% at low aggregate throughputs. Therefore, values above 0.4 seconds can be considered to have negligible effect on throughput; however, in spite of this due to simulation complexity, only beacon rates of 1 second and above are examined in this thesis. As a result of this, delivery successes shown in the thesis can be considered a small underestimate of those that could be expected with higher rates.

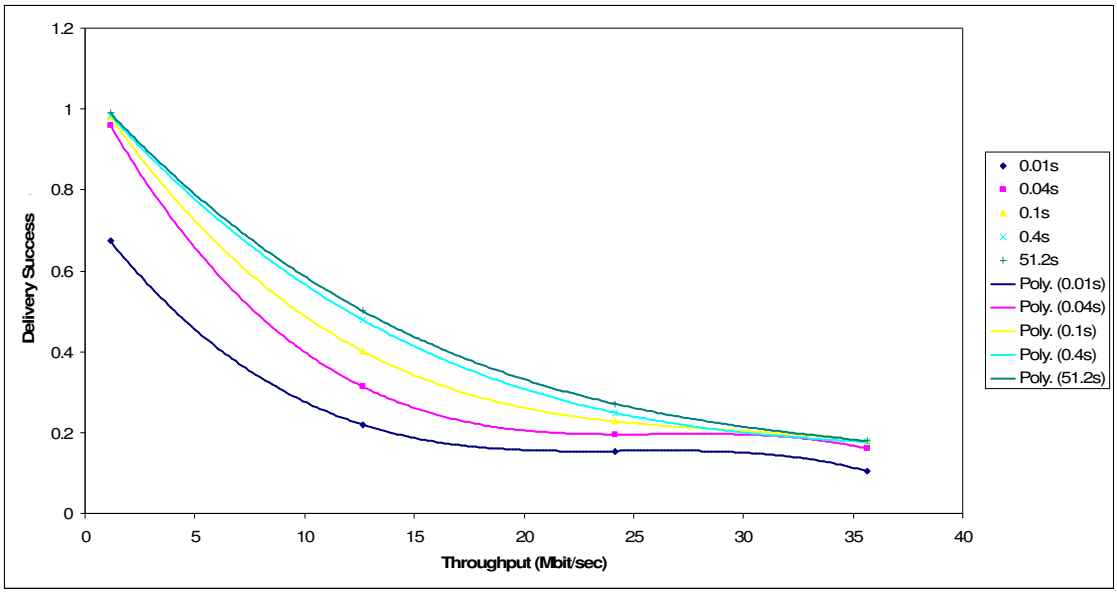

Figure 83: Effect of beacon rate on throughput

8.3.3 Conclusions

Typically hops in a large network will be long and include tens, hundreds or even thousands of hops. Inevitable longer paths will incur more delay but it is important that delay for average hops is kept to a minimum or below that which is likely to have a noticeable effect on video and voice applications. Equally important is that nodes must conserve their battery life otherwise a loss of a number of nodes can result in poor routing success.

Beaconing can be performed without switching the entire system into sleep mode by implementing a distinct hardware entity to perform the function. Another concern is that beaconing could impair throughput but as shown in Figure 82 the frequency of beacons is of the order of seconds, not the milliseconds required to impair throughput.

Based upon the evidence presented in this subsection it is concluded that beaconing is essential in ad hoc networks if delays are to be kept to an acceptable minimum. The reduction in throughput is negligible at low beacon rates.