Direction Based Routing for Mobile Ad Hoc Networks

Andronikos Nedos

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

September 16, 2001

Declaration

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

Signed: ____________________

Andronikos Nedos

Date: September 16, 2001

Permission to lend and/or copy

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

Signed: ____________________

Andronikos Nedos

Date: September 16, 2001

Abstract

A Mobile Ad Hoc Network is comprised of nodes with wireless interfaces that are able to move arbitrarily, changing the network topology as they do so. Designing a routing protocol for such a network, where connections can be broken and re-established with different paths is challenging and a standard routing scheme has yet to emerge.

Routing protocols for Mobile Ad Hoc Networks have so far taken approaches similar to routing protocols used for traditional wired non-mobile networks. Variations of *link-state* or *source-routing* protocols or combinations of both have been proposed. Recently, a class of protocols that exploit the physical location of mobile nodes for more efficient routing of data has appeared. The subject of this thesis is related to such location-aware protocols.

We argue that by taking advantage of the location and the mobility path nodes follow, routing can be improved. We identify and explore a number of algorithms that can provide a metric for similarity of direction. One of these metrics is then implemented in a simulation environment (*ns*) using the Dynamic Source Routing (DSR) protocol. The protocol is then tested with mobility scenarios and communication patterns and results are extracted that compare its performance against the original DSR protocol. The three performance metrics that are evaluated from the simulation results include the longevity of routes, the routing efficiency and the channel usage of the protocol. Results have proved no significant performance gains of the Direction Based DSR against the standard DSR, but have shown scalability problems associated with DSR in densely populated networks.

Acknowledgements

I would like to thank my supervisor Mads Haahr for his invaluable help and support throughout the year. Many thanks also go to the M.Sc. NDS class for making this year really enjoyable. Thanks to mum and dad for the extra love and much needed cash.

Contents

1 Introduction **1**

1.1 Motivation . . . 1

1.2 Ad Hoc Networks . . . 2

1.2.1 Characteristics of Mobile Ad Hoc Networks . . . 4

1.2.2 Applications of Ad Hoc Networking . . . 7

1.3 Objectives . . . 8

1.4 Document Structure . . . 10

2 State of the Art **11**

2.1 Dynamic Source Routing . . . 13

2.1.1 Overview . . . 14

2.1.2 Route Discovery . . . 15

2.1.3 Route Maintenance . . . 18

2.1.4 Optimisation Techniques . . . 19

2.1.5 Summary . . . 20

2.2 Location Aided Routing . . . 21

2.2.1 Overview . . . 21

2.2.2 Summary . . . 23

2.3 Destination Sequence Distance Vector . . . 23

2.3.1 Protocol Overview . . . 24

2.3.2 Responding to Topology Changes . . . 25

2.4 Summary 25

3 Analysis and Design **27**

3.1 Background 28

3.1.1 Associated Concepts 28

3.2 Understanding Direction 31

3.3 Directional Routing 32

3.3.1 A Metric for Heading 33

3.3.2 A Metric for Distance 37

3.3.3 A Metric for Area 38

3.4 Selecting a Directional Metric 39

3.4.1 Choosing a Metric 40

3.4.2 Choosing a Protocol Type 40

3.4.3 Choosing the Protocol 42

3.5 Summary 44

4 Implementation **46**

4.1 The *ns* Network Simulator 47

4.1.1 Overview 47

4.1.2 Architecture 47

4.2 The CMU Monarch Extensions 50

4.3 The DSR Implementation 53

4.4 Extending DSR 55

4.5 Summary 58

5 Experiments and Evaluation **59**

5.1 Measuring Aims 60

5.2 Simulation Model 61

5.3 Performance Metrics 63

5.3.1 Number of Route Requests 64

5.3.2 Packet Delivery Ratio 65
5.3.3 Channel Usage 67
5.4 Summary 71

6 Conclusion **73**
6.1 Objectives Fulfilled 73
6.2 Future Work 74

List of Figures

1.1 A Simple Ad Hoc Network 4

2.1 Source Routing in Dynamic Source Routing 14

2.2 Route Request in Dynamic Source Routing 16

2.3 Reducing flooding during route discovery in LAR 22

3.1 Direction based routing in an Ad Hoc Network 29

3.2 Rate of angular change in a movement 34

3.3 Approximating direction by using a resultant vector 38

3.4 Choosing a Directional Metric 41

3.5 Appending vector information during route discovery. 43

4.1 C++ and OTcl usage in *ns* 48

4.2 Schematic of a mobile node (source: [MP]) 51

List of Tables

4.1 Optimisations used by the DSR implementation 54

5.1 Fixed simulation parameters 62

5.2 Total no. of route requests (10m/sec) 65

Chapter 1

Introduction

1.1 Motivation

Adaptation of technology to human needs has always been a great incentive for innovation in the fields of science and technology. This factor, together with the continuous growth in processing power and the miniaturisation of hardware has led to a tremendous rise in mobile computing. Users are no longer restricted by stationary computers in order to enjoy networked applications. By means of portable computers or Personal Digital Assistants (PDAs) equipped with wireless interfaces, they are allowed access to networked communication even while mobile.

Much of the shift towards mobile computing has to do with the fact that we want to retain a high degree of connectivity to the Internet. Indeed, the convenience that was offered originally with mobile phones and paging has been transforming (or converging) to keeping track of e-mails and scheduling appointments while outside office or house. Even more, a commercial PDA nowadays (like Compaq's IPAQ) can accept a wide range of expansion modules including wireless ethernet and Global Position System (GPS) cards. This gives rise to a whole new range of application domains that can be accessed while users are not restricted by fixed communication and the traditional bulkiness of personal computers. Consider for example that one might want to receive driving suggestions, based on information from GPS or even have location-based services as he or she roams around a new city they are not familiar with. This is a reality that is made possible today by small, inexpensive mobile devices and the affordable cost of having wireless communication links offering reasonable bandwidth.

As wireless devices of all types proliferate and their cost and size is reduced, users will demand access to information or application inter-connectivity even when they are outside the reach of the traditional Internet. That could be the case when either wireless Internet is not an option or because a failed component makes access to the Internet temporarily unavailable. As Perkins points out in [Per01, page 2] people using laptop computers at a conference in a hotel might wish to communicate in an ad hoc manner, without the mediation of routing across the global Internet. Yet today such obvious communications requirements cannot be easily met using Internet protocols. The theme of this thesis concerns networks that rise from the need to form this kind of spontaneous, short-lived, partially connected networks from mobile computers with wireless network interfaces.

1.2 Ad Hoc Networks

The Working Group on Mobile Ad Hoc Networks of the Internet Engineering Task Force (IETF MANET WG) defines ad hoc networks as:

> A "mobile ad hoc network" (MANET) is an autonomous system of mobile routers (and associated hosts) connected by wireless links—the union of which form an arbitrary graph. The routers are free to move randomly and organize themselves arbitrarily; thus, the network's wireless topology may change rapidly and unpredictably. Such a network may operate in a stand-alone fashion, or may be connected to the larger Internet.[1]

[1] `http://www.ietf.org/html.charters/manet-charter.html`

The above definition suggests a network that is characterised by certain unique properties unlike those found in common networks that were designed with a fixed infrastructure in mind like the Internet or Local Area Networks.[2]

As an example of a small ad hoc network consider figure 1.1. The network nodes, depicted by the solid line circles, are able to move relative to each other. As this is happening, their neighbouring nodes will change, establishing new links with nodes that are inside the coverage area of their wireless interface and breaking other links as nodes become distant from each other. In the above figure, node A is initially connected to node B. As it moves away it will establish new connections with nodes C and D. Thus, its physical location has changed but the node still wants to be an active network participant, exchanging data with the rest of the network and having its applications being agnostic of the node's mobility and changing neighbouring nodes.

In a communication scenario where node A transmits data to node C, initially the data sent will have to travel through node B, since *A's* communication range includes only B. This ability to relay data through intermediate nodes until they reach their destination is called *multi-hop.* Most store-and-forward packet switching networks like the Internet, work in a multi-hop fashion.[3] The problem with ad hoc networks lies in the fact that the network topology can change rather frequently, which results in established route paths becoming invalid after a certain (short) period of time. In the same example, when A migrates to its final position, data destined for C will no longer require a path through B but can be transmitted directly, since A is now a neighbouring node with C and D. For this reason, ad hoc networks present a totally different scenario and a whole new spectrum of design decisions and options when related to how non-mobile networks work. It is the reason that some of the routing protocols in ad hoc networks had to be re-designed

[2]Though current advancements in the Internet and LANs try to incorporate solutions to problems found in ad hoc networks (e.g. Mobile-IP for the Internet and wireless IEEE 803.11 LAN).

[3]In fact, it is the network layer Internet Protocol (IP) that is said to be multi-hop.

from the ground up rather than being simple modifications of non-mobile network protocols or required major modifications from existing protocols.

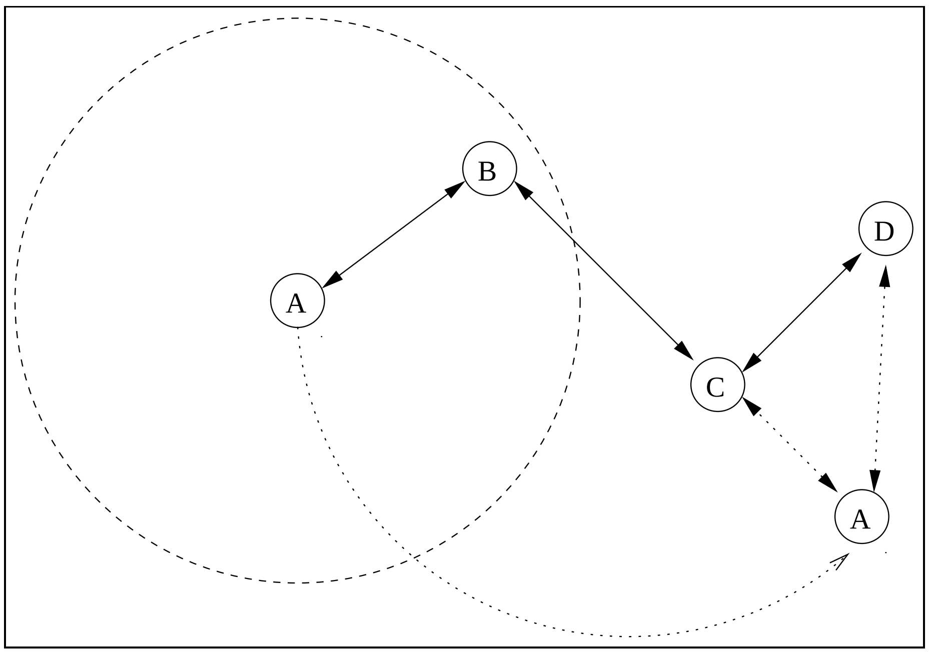

Figure 1.1: A Simple Ad Hoc Network

The primary focus of this thesis is on routing protocols for ad hoc networks. A discussion follows on some of the most salient characteristics and the application domains of ad hoc networks that can influence the design of routing algorithms. In fact, such is the very nature of the field that the characteristics described can dictate routing solutions that work well only in particular domains but fail when some conditions change. This and the lack of a standardised specification has contributed to a plethora of routing protocols with different architectures and properties that will be examined in more details in chapter 2.

1.2.1 Characteristics of Mobile Ad Hoc Networks

A MANET can be thought of as a collection of mobile platforms each combining the functionality of a router and a host. These mobile platforms, also called hosts, are attached to a number of wireless communication mediums and are free to move about arbitrarily. These hosts are equipped with wireless transmitters and receivers using antennas which may be omnidirectional (broadcast), highly directional (point-to-point), possible steerable or some combination thereof [CM99]. It is worth pointing out here, that the actual wireless medium is not important for the functionality of the upper network layers, since many of the same principles will apply regardless of the actual technology used. Examples include infra-red, wireless RF, bluetooth or even wired based transceivers used when emulating ad hoc networks.

Node mobility introduces certain scalability problems in ad hoc network protocols. When network topology changes frequently, control messages have to be sent between nodes so that new routes are found and propagated through the network. In networks where topology changes infrequently, it is reasonable to expect that when this happens there might be a short period where lots of control messages will propagate the network to distribute the new destination paths. But when a high rate of topology change is one of the characteristics of the network, the protocol designer should make provision for highly dynamic and fast adapting algorithms that minimise control messages and attempt to utilise long-lived routes to the maximum extent.

Minimising control messages is also essential because of the additional load that they place in the bandwidth constrained wireless links. Nowadays, there is at least an order of magnitude difference between wired and wireless data rates with the former having a standardised 100Mbit/sec in a local ethernet while the latter having, in the best case, a nominal bit rate of 10Mbit/sec.[4] The techniques used to reduce control messages must strike a balance between a minimum amount of messages by keeping network state information in each node and flooding the network each time the topology changes. The former has the side-effect that in an ever changing network topology stale routes will appear often, while when the latter is applied

[4]In fact due to interference and other problems end users usually end with a 1 to 2 Mbit/sec data rates.

to a network containing nodes with high rate of mobility, it might result in control messages consuming all the available bandwidth. This could also lead to very slow network convergence where nodes contain either incomplete or out-of-date views of the network topology.

In many kinds of ad hoc networks, mobile nodes usually rely on exhaustible means for providing energy, such as batteries. For these nodes, energy conservation suddenly becomes an important design decision. Nodes with low battery power may decide to enter a power saving mode when they have nothing to send or until another high priority event is generated. This behavior may effect the way the whole network is operating, since each node is responsible for forwarding other nodes' packets, apart from its own. If nodes decide to become "selfish" and break the collective and cooperative nature of ad hoc networking by not forwarding other nodes' data, the ad hoc architecture is endangered. A multitude of other problems and design trade-offs is concerned with power utilization in such networks and this particular area is becoming the focus of increased attention.

Last but not least, we should examine briefly some security issues in ad hoc networks. As with any wireless communications, ad hoc networks can be highly vulnerable to security threats. On one hand, their distributed nature makes it difficult to implement any security scheme that relies on a central authority (e.g. for key distribution and management), while on the other hand there is an increased possibility for *eavesdropping*, *denial-of-service* and *man-in-the-middle* attacks. Add to this, the fact that the more secure an architecture is, the slower and more cumbersome it becomes and the whole issue of security in ad hoc networks becomes somewhat problematic. Nevertheless, the decentralised architecture provides a more resilient approach to single points of failures.

1.2.2 Applications of Ad Hoc Networking

The history behind the evolution of ad hoc networks is given in the beginning of chapter 2. In this section, an outline is given on some of the application domains that could suit the unique characteristics of ad hoc networks.

The two most cited applications for ad hoc networks are communications in situations where battlefield survivability counts or infrastructure is non-existent which is the case during disaster relief or rescue operations. Both of these applications rely on the decentralised and cooperative attributes of MANETs. However, a number of other applications can be envisioned.

Conferencing

Conferencing is perhaps the application domain that could produce the *killer app* for ad hoc networks. This scenario envisages a group of people gathering in the same area and exchanging shared information using the multi-hop characteristics of ad hoc networks. Currently, this is done by requiring everyone to connect to a central network which at times might be unavailable or the overhead might be to costly when all that is required is the sharing of small amount of data.

Ubiquitous Computing

If projections and estimations are correct then soon we could be living in a world where electronic devices can join spontaneously in established networks and exchange data with other devices in their close vicinity, in a transparent and simple fashion. *Bluetooth* is an emerging standard for realising such a vision and is backed by such companies as Ericsson Inc, IBM, Intel, Microsoft and Nokia. It is a short-range radio technology aimed at eliminating wires between electronic devices. Bluetooth allows up to eight devices to be connected into what is called a piconet. This could be a suitable technology that ad hoc networks could use for transferring information without utilising fixed infrastructure.

Data Gathering

Another application where ad hoc networks can prove useful is the collection of data from remote areas or terrains like air or sea. A network of sensors can contain small, inexpensive, short-range radio transmitters that can collectively gather and forward data towards a base station. Current alternatives include either storing data for later collection or using large and expensive satellite transmitters. But for cases like animal tracking where small transmitters are essential, ad hoc networks can prove to be a useful tool, provided there is a reasonable coverage of the specific area with tracking devices.

Another potential application domain is data gathering in urban environments through the use of vehicles. Cars for example, are much better suited as carriers of equipment, like tranceivers and batteries, which enable short to medium range communication. An urban ad hoc network can then be formed collecting and propagating up to date information about traffic characteristic in order to provide traffic information and perform congestion control, even allowing communication between passengers to take place.

1.3 Objectives

This section describes the objectives and the scope of the project as well as tasks that although related will not be addressed. These could be the aim of further research into the field.

1. The specification and design of a novel ad hoc routing algorithm that takes into account the location and direction of moving nodes. This is interpreted into a set of principles defining a routing metric that mobile nodes should apply whenever a routing decision is made.

2. The implementation of this routing algorithm in a simulation framework.

3. Verifying the correctness in the implementation of the simulated algorithm by providing a set of tests and interpreting the results. This is different from the evaluation of the performance of the algorithm in that we are trying to verify that the individual modules of the algorithm (cache module, state machine, etc) work as intended. Because of the difficulty of this task we could drop back into qualitative verification by source code review if time constrains are tight.

4. Evaluate the performance of the algorithm by constructing different mobility scenarios with varying parameters such as the number of nodes, their mobility patterns, velocity, transmission range etc. and test the protocol under these scenarios. Data generated by these test runs, would then be aggregated and suitable analysis should be conducted. It would also be interesting to apply this routing algorithm to the same kind of scenarios as other ad hoc routing protocols and make a comparative analysis in terms of efficiency, latency, overhead. A useful guide in this task could be the evaluation criteria, set by the IETF MANET WG in [CM99].

The issues that this thesis will *not* examine, research and deal with are:

1. The design of a fully fledged routing protocol. Certain work in this area can be reused and extended to meet the requirements of this thesis. The general format of routing packets should remain unchanged, as should the different timing and cache size values.

2. Interoperability with the Internet will not be considered and the algorithm should be assumed to run in layer 3 of the OSI/ISO model. Network addressing will be provided by the simulation environment.

1.4 Document Structure

This section outlines the chapters of the thesis, the contents of which are listed below.

Chapter 1: Introduction describes the motivation behind the Direction Based Routing project, makes an introduction to the concepts involved and states the objectives.

Chapter 2: State of the Art looks at relevant material in the area of ad hoc routing protocols.

Chapter 3: Analysis and Design introduces the ideas behind the Direction Based Routing project, identifies their properties, develops and discusses the relevent metrics. It also lays down the rules by which the protocol will operate.

Chapter 4: Implementation looks at the way the Direction Based Routing protocol was implemented in the simulation environment. There is an outline of the *ns* network simulator and the implementation details of the DSR protocol.

Chapter 5: Experiments and Evaluation describes the experiments performed to test the applicability of the new metric. An evaluation section interprets the results produced.

Chapter 6: Conclusion concludes the project.

Chapter 2

State of the Art

The initial success of packet switching technology during the 1960s, first demonstrated by the ARPAnet, gave rise to systems that could exploit this technology without the burden of a fixed wired infrastructure. Recognising the advantages of packet switching in a mobile wireless environment, in 1972 DARPA initiated a research effort to develop and demonstrate a packet radio network (PRNet). Although the initial PRNet protocols used a centralised control station, the core PRNet concept quickly evolved into a distributed architecture consisting of a network of broadcast radios with minimal central control, using multihop store-and-forward routing techniques to create complete end-user connectivity from incomplete radio connectivity [Per01, pages 33–34].

Over the years, commercial (and military) interest continued to grow for wireless mobile ad hoc networks as the underlying technology improved and the demand for new application paradigms involving ad hoc networks increased. With the introduction of the Internet Engineering Task Force MANET Working Group (IETF MANET WG) with both commercial and military participation, another step towards standardising a routing protocol among a dozen candidates was made.

Today, there are substantial number of algorithms available that exploit wildly different properties and characteristics of the mobile hosts and the network topology. However, most of these algorithms are still at an experimental stage with only few of them having actual implementations, while the majority relies on simulation results for further ramifications and improvements. It is fortunate that recent research has provided more comparative performance results between the existing algorithms giving more independent and objective views on the capabilities of each one [BMJ+98, DPR00].

In general, ad hoc routing systems can be classified into those that discover a path to the destination node at the time it is needed and those that try to maintain an up to date view of the changing network topology. Former systems are called *reactive* while latter protocols are characterised as *proactive*. Reactive protocols include the Dynamic Source Routing (DSR) [JM96] the Ad Hoc On Demand Distance Vector Routing (AODV) [PR99] and the Temporally-Ordered Routing Algorithm (TORA) [PC98]. Examples of proactive protocols include RIP (or distributed Bellman-Ford, DBF) [Mal94], OSPF [Moy94], and Destination Sequenced Distance Vector (DSDV) [PB94]. These protocols try to continuously maintain route entries to all destinations on the face of topology changes. The distinction between *reactive* and *proactive* protocols in ad hoc networks is similar as between having an IP source routing policy or the more widespread IP forwarding in the Internet. IP source routing allows IP packets to carry the sequence of routers and hosts they will be routed through, while with the IP forwarding approach, every router maintains link information and is responsible for choosing the next router.

The idea behind this thesis is closer to protocols that have appeared recently and make use of geographic position of network nodes to distribute messages and find routes. Routing schemes that are based on the location of nodes take advantage of the relation that exists in ad hoc networks between physical and network proximity.[1] Protocols that facilitate geographic forwarding of packets include LAR [YBKV98] and DREAM [BCSW99]. Exploiting location of nodes is an important aspect that is utilised in differing degrees for several other routing protocols. Su et al. have

[1]This suggests that nodes that are physical neighbours will probably take on the role of being network neighbours.

devised an algorithm for mobility prediction and routing [SLG00], Lin et al. use location information to provide loop-free routing [LS] and Morris et al. use a scalable distributed algorithm to keep track of nodes' positional information that can be used for routing purposes [LJD$^+$00].

The state of the art review is organised as follows. Section 2.1 describes the DSR protocol and some of its optimisations. Section 2.2 discusses LAR, a geographic routing protocol. Both of these protocols are *reactive*, with LAR being a further optimisation to the way DSR operates. Section 2.3 gives an overview of the salient characteristics of a *proactive* protocol, namely DSDV. The discussion will identify differences between the approaches and their respective advantages and disadvantages. All of the above protocols are chosen as representing the particular type and architecture which they belong and because of the increased attention they have received in the literature. Section 2.4 concludes with a summary of this chapter.

2.1 Dynamic Source Routing

Dynamic Source Routing is a routing protocol specifically designed for wireless mobile ad hoc networks. Its specification was published originally in [JM96], and subsequent changes to the protocol were made as Internet drafts with the latest being [JMHJ00].

As with any routing protocol, DSR was designed so that nodes would be allowed to dynamically discover and use routes to any other nodes in the ad hoc network.[2] It does so by using source routing. Source routing is a technique that allows nodes to communicate with each other by explicitly encoding the complete path to a destination in every packet's header. DSR, being a *reactive* protocol, tries to dynamically identify possible paths to a destination only when needed. This is in contrast to distance vector or link state protocols where they try to continuously monitor and

[2]Provided of course there is a sequence of nodes through which a packet can be forwarded to reach the destination.

update their list of paths towards possible destinations. Figure 2.1 shows an example of how source routing works. A packet is tagged with the complete path from source to destination node, with each intermediate node identifying its position in the header and then forwarding the packet to the next node in line.

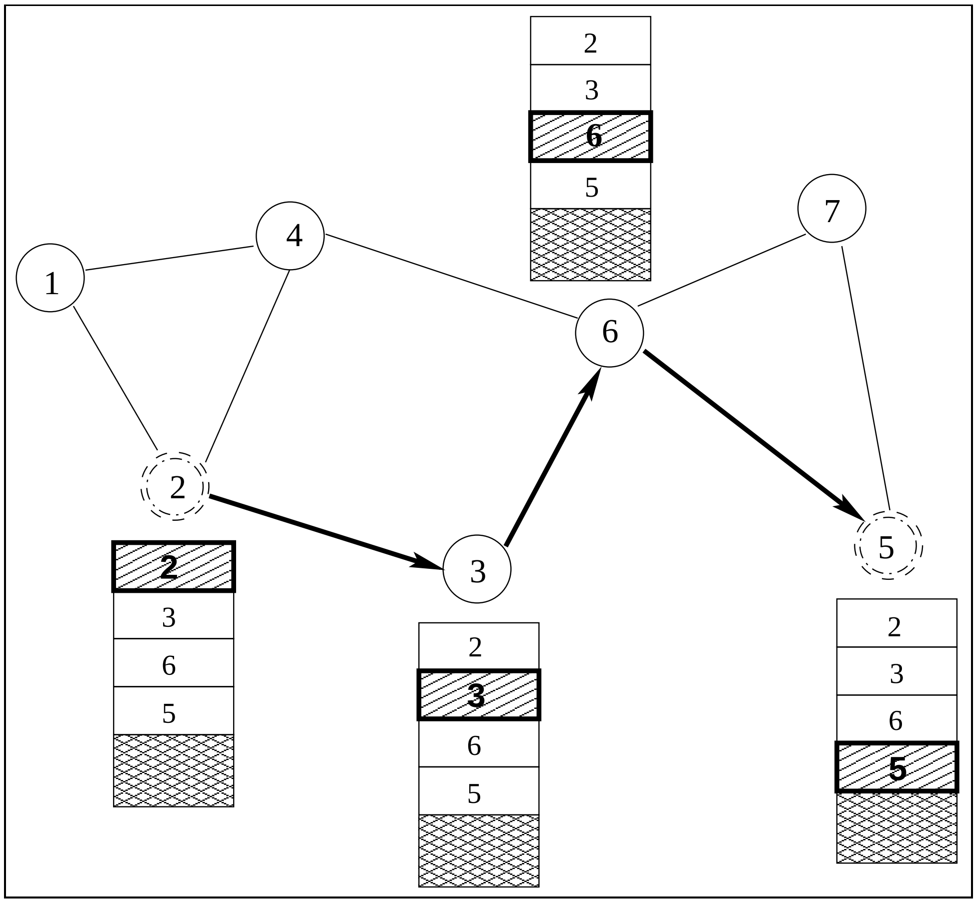

Figure 2.1: Source Routing in Dynamic Source Routing

2.1.1 Overview

The DSR protocol operates in three distinct modes.

- *Route Discovery*, in which any node that wants to find a route to a destination obtains a full path to it.

- *Route Maintenance*, is the mechanism by which a change in the network topology leading to a route breakage, will cause a new route to be used.
- *Data Relay*, the transmission of data from higher layer protocols to next hop neighbours, when a valid source route exists.

As stated previously, both route discovery and route maintenance operate on demand. Each node is also equipped with a *route cache*, for caching source routes that the node has learned. DSR in particular, makes aggressive use of caching techniques in order to improve efficiency. Some comparative results, when different caching techniques are used have been published in [HJ00].

2.1.2 Route Discovery

Whenever a source node transmits data to a destination node, it appends the full route of the intermediate nodes in the packet header. This route can either be extracted from the route cache maintained at the source node, or in case that the path to the destination is unknown, a *route discovery* sequence is initiated. In the latter case, the source node is called the *initiator* and the destination node the *target* of the route discovery.

Figure 2.2 illustrates an example where initiator node 2 commences a route discovery to find a path (or multiple paths) to destination node 5. To do that, node 2 transmits a ROUTE REQUEST packet which is received by all nodes currently in the wireless transmission range of 2. Each ROUTE REQUEST message contains the address of the node that initiated the request as well as the address of the destination node. It also contains a unique[3] *request ID* that is created by the initiator node. Furthermore, every node that receives the ROUTE REQUEST appends its own address in a record that is part of the packet header and lists the intermediate nodes that the ROUTE REQUEST has been through.

[3]Uniqueness is guaranteed in the scope of route discovery and source node

Any host that receives a ROUTE REQUEST attempts to identify if it is the destination of the request in which case the request is processed accordingly. If the host is not the destination, the request is either forwarded to the node's neighbours through broadcast or discarded if it has been seen before. It is worth mentioning at this point the technique that is employed by DSR to avoid loops. As noted, every ROUTE REQUEST contains the source, destination and request ID as part of its header. When a request is first seen by a host, it is cached for a certain period of time. Every subsequent request seen is compared against the source and the request ID fields of every cached copy. If it exists in the cache the packet is discarded, otherwise the relevant fields are stored in the route cache and the packet is further processed. This provides an inexpensive and simple way to avoid loops in the network, since every request originating from a certain source node will be

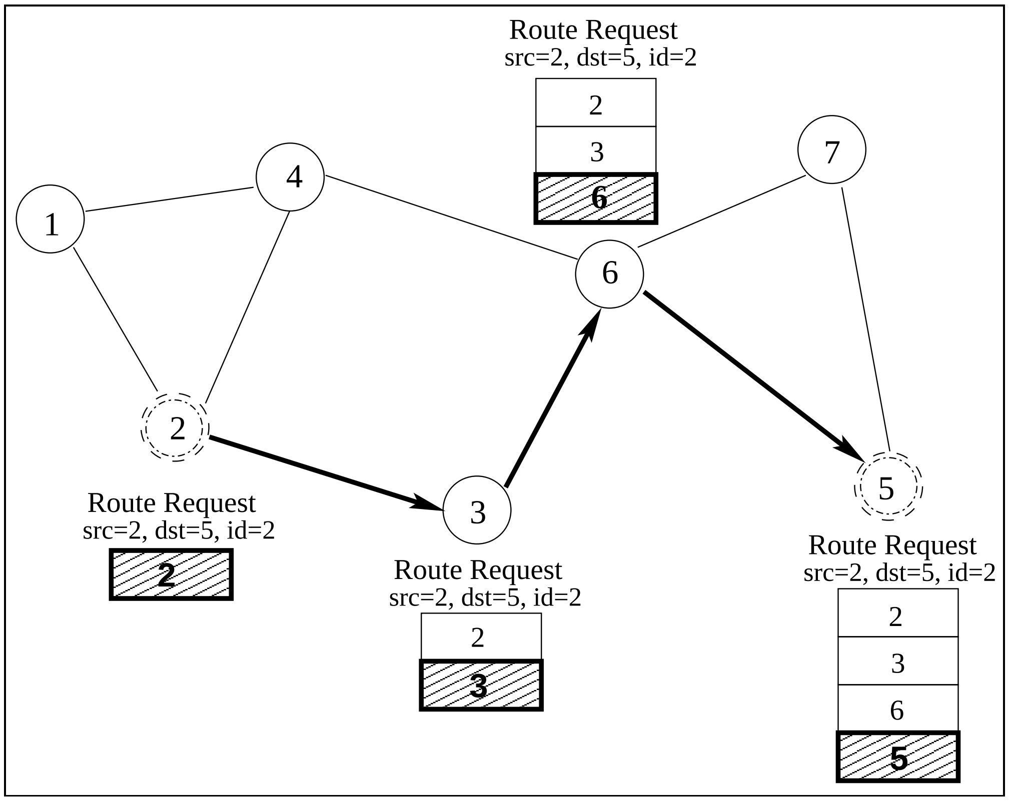

Figure 2.2: Route Request in Dynamic Source Routing

processed and forwarded only once by each subsequent node.

The request thus propagates through the ad hoc network until it reaches the target host, which then replies to the initiator. This happens by sending a ROUTE REPLY message to the source of the request that contains the fully ordered list of the hosts that the ROUTE REQUEST has been through. In the example of figure 2.2, ROUTE REPLY will contain the addresses of nodes [2,3,6,5]. When the ROUTE REPLY is received by the initiator node, the path to the destination is stored in its cache. Any packets that were buffered waiting for a valid route to the destination are tagged with the newly received path and are sent off. DSR employes several optimisations in many parts of its architecture and some of them are mentioned in section 2.1.4. Specifically to choosing a route for the source node during a ROUTE REPLY, the destination has several options depending on the nature of the ad hoc network.

- If a route to the initiator node exists in the target's cache, it can use this path to propagate the ROUTE REPLY.
- It can reverse the list of nodes the ROUTE REQUEST has stored in its record. That implies that nodes in the network have support for bi-directional links, something that is not always true in ad hoc networks due to interference, various obstacles and differing MAC architectures.
- In the case that bi-directional links are not supported, the destination node can piggyback the ROUTE REPLY in another ROUTE REQUEST message aimed at discovering a path to the initiator node.

Route discovery is probably the most expensive function in reactive protocols, like DSR. Especially when no technique is used to limit the scope of the route discovery and flooding is employed to identify destinations, it can cause traffic delays, increased latency, bandwidth consumption and create communication overhead. There are a number of ways to attack the above problems, each involving trade offs in some other areas. One can hope to decrease the effective zone of the route discovery thereby reducing traffic and minimising overhead, which is what the LAR protocol does. Other approaches like the ZRP [HP97] try to restrict route discovery to neighbouring nodes by creating a cluster environment and keeping some network state information in the node that acts as the cluster-head. Our approach aims at creating long-living routes, balancing in this way the effects of flooding during every route discovery.

2.1.3 Route Maintenance

Conventional routing protocols integrate route discovery with route maintenance by continuously sending periodic route updates. If the status of a link or router changes, the periodic updates will eventually reflect the changes to all other routers, presumably resulting in the computation of new routes. However, using route discovery, there are no periodic messages of any kind from any of the mobile hosts. Instead, while a route is in use, the route maintenance procedure monitors the operation of the route and informs the sender of any routing errors.

When a node along a routing path, detects a broken link, it returns a ROUTE ERROR message back to the originating node. The ROUTE ERROR message contains information to identify the link over which the error occurred. It contains the address of the node that identified the error as well as the address of the node that could not be reached. When the initiator node receives the ROUTE ERROR message it will purge from its route cache all the routes that contain the broken link. Any subsequent transmissions requiring this route will probably result in another route discovery (provided there is no alternative cached route to the destination).

The way that broken links are detected is usually done through receipt confirmation. Sometimes this is provided to DSR for free by layer 2 protocols (such as the link level acknowledgement defined by IEEE 802.11), otherwise DSR may explicitly request an acknowledgement by the next hop node by setting a flag in the packet's header. Finding a route to the originating node for a ROUTE ERROR message uses the same options that were mentioned for a ROUTE REPLY message in section 2.1.2. The reporting node can either decide to extract a route from its cache (provided there is one), reverse the route that exist in the packet it was trying to send or initiate a new ROUTE REQUEST to find a path to the source node.

2.1.4 Optimisation Techniques

DSR accepts a variety of optimisations, most of them having to do with aggressive use of the route cache. The following paragraphs mention some of these optimisations and give a brief overview for each one.

Promiscuous mode

DSR can "snoop" on any packet that it receives and extract information from the packet's header that it then processes and stores in its cache. It is not necessary that only originator nodes will cache routes from a ROUTE REPLY message, but any intermediate node that forwards the ROUTE REPLY can cache the route that is carried. This of course on the expense of processing power and memory consumption. The same idea can be applied for ROUTE ERROR packets making distribution of broken links faster.

Reply Using Cached Routes

Another form of optimisation is the ability of intermediate nodes to reply to ROUTE REQUEST messages by searching their cache for a route to the destination of the route discovery, instead of just broadcasting the request packet to their neighbours. In case a route exists to the destination of the ROUTE REQUEST, the cached path is appended to the path that is stored in the request's header and a ROUTE REPLY is sent to the initiator node by the intermediate node.

Preventing Route Reply Storms

A situation might occur in DSR, whereby neighbouring nodes can choose to respond with ROUTE REPLY messages simultaneously. This happens when nodes receive a ROUTE REQUEST and have the destination cached (possible because the destination is their neighbour). To avoid this behaviour, which can lead to wasting precious bandwidth, nodes snoop on neighbouring ROUTE REPLY messages and only send their own reply if they have a better (shorter) route to the destination.

Packet Salvaging

After sending a ROUTE ERROR message as part of route maintenance described in section 2.1.3, a node may attempt to *salvage* the data packet that caused the ROUTE ERROR rather than discard it. To salvage a packet, the node sending a ROUTE ERROR searches its own route cache for a route from itself to the destination of the packet causing the error. If such a route exists, the node replaces the source route of the salvaged packet with the one existing in its cache. To avoid potential routing loops, caused by nodes continuously salvaging the packet, a flag is set, so that further nodes will not attempt to salvage it again.

2.1.5 Summary

DSR is one of the few ad hoc routing protocols that apart from having undergone quite extensive simulation evaluation, has been implemented and tested in a real network [MBJ99]. DSR is essentially a protocol that uses flooding to discover routes to nodes only when needed. As such, no periodic transmissions are needed, but the full route to destination is included into every packet that is sent with DSR. This is one source of overhead that cannot be defeated by using intelligent caching techniques employed for reducing the amount of time needed for route discovery and propagation of broken links. Simulation results [BMJ$^+$98] have proven DSR to outperform 3 other protocols (DSDV, TORA, AODV). However, testing was done on network with 50 nodes by varying traffic and mobility patterns. It remains to be seen how DSR would operate in larger networks, where flooding the network for route discovery might create serious network congestion.

2.2 Location Aided Routing

2.2.1 Overview

This section describes another routing protocol which uses the physical location of mobile hosts in its route discovery operation. Protocols that make use of such location information are called *location-aware* and the protocol reviewed here is called Location Aided Routing (LAR) [YBKV98]. LAR attempts to reduce the route discovery overhead by limiting the physical area where nodes reside. The way that it tries to achieve that, is by defining an *Expected Zone* and a *Request Zone.*

Consider figure 2.3 for example, where host S wants to find a route to host D. Assuming node S knows that node D was at location L at time t_0 and that current time is t_1, then the *expected zone* for node D is an area where node S expects D to be found on time t_1. An approximation can be made if node S could know the speed by which node D is moving. Then it could calculate the expected zone as a circular region with radius $u(t_1 - t_0)$, centred at location L. In figure 2.3 the expected zone corresponds to the greyed out circle around node D. Naturally, if the location information at some previous moment in time is unknown, nothing can be assumed about the current position of any destination node. In this case LAR falls back to being a flooding protocol during route discovery (e.g. DSR).

The *request zone* is the core of the LAR protocol. When a node has stored location information about potential destination nodes, it defines a request zone during route discovery in which only nodes belonging physically in this zone will be allowed to forward the ROUTE REQUEST messages. To increase the probability that the route request will reach node D, the request zone should include the *expected* *zone.* LAR defines two schemes on how to construct a request zone. The first scheme is like the one defined in figure 2.3. Essentially, it is the smallest rectangle that includes current location of the source node and the expected zone of the destination node (the circular region defined above), such that the sides of the rectangle are parallel to the X and Y axis. In the example, the request zone under scheme 1 is the rectangle defined by the S,A,B and C vertices.

In LAR scheme 1, source S explicitly specifies the request zone in its route request message. In scheme 2, node S includes two pieces of information with its route request:

- Its distance to the destination node, denoted as $DIST_s$.
- The coordinates of the destination (X_d, Y_d).

Under this scheme, every node that receives the route request checks if its distance is less than the distance defined by the previous node. If it is, the node updates the distance stored in the route request message with its own distance from the destination and forwards the message to its neighbours. Otherwise, the message is discarded with no further processing.

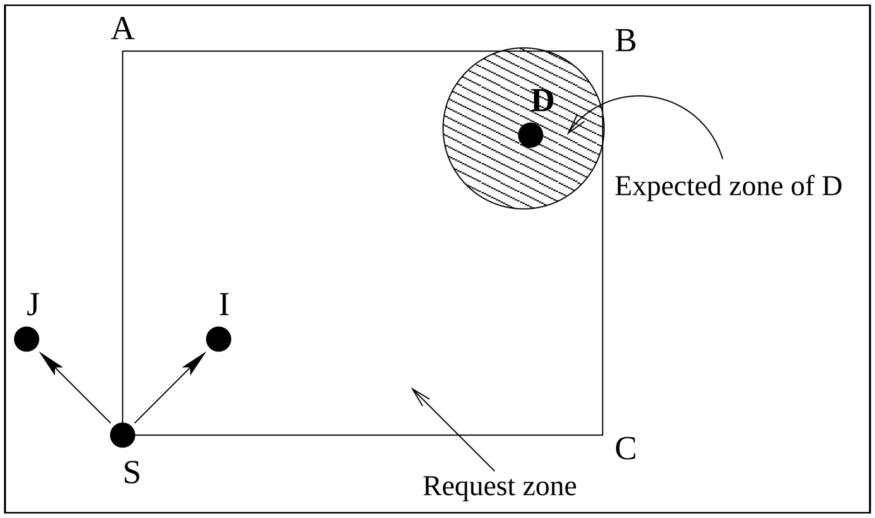

Figure 2.3: Reducing flooding during route discovery in LAR

2.2.2 Summary

LAR is a simple routing algorithm that relies on location information about nodes to reduce the number of routing messages required during route discovery. LAR can be seen as an optimisation to flooding protocols like DSR, improving their route discovery procedure. Some evaluation and testing in a simulation environment has been produced by the writers of LAR, showing LAR to outperform basic flooding protocols in both its schemes, with scheme 2 being slightly better in its efficiency than scheme 1.

Tseng et al in [YC], described GRID which is another location aware routing protocol. A comparative simulation analysis made in the same paper between GRID and between LAR and AODV, found GRID to outperform both LAR and AODV in areas like routing cost, delivery rate and maximum connection establishing time.

2.3 Destination Sequence Distance Vector

This section describes the third ad hoc routing protocol that is based on the *proactive* approach and is in fact an extension to the standard distance-vector algorithms. Although there are well known problems that are associated with this type of algorithm (i.e. *count-to-infinity*), DSDV tries to overcome them with techniques that shall be described below.

As described by Bertsekas et al. in [BG87], in traditional distance-vector algorithms, every node i maintains, for each destination x, a set of distances $\{d_{ij}(x)\}$ for each node j that is a neighbour of i. Node i treats neighbour k as a next hop for a packet destined for x if $d_{ik}(x)$ equals $min\{d_{ij}(x)\}$. The succession of next hops chosen in this manner leads to x along the shortest path. To keep the distance up to date, each node monitors the cost of its outgoing links and periodically broadcasts, to all of its neighbours, the current estimate of the shortest distance to every other node in the network.

2.3.1 Protocol Overview

In DSDV, packets are routed by using route tables stored at each node. Each route table maintains entries for every other node in the network. These entries include the address of the destination node as well as the address of the next hop that packets should take in order to reach the destination. Each such entry is also tagged with a sequence number that is generated from the destination node. In order to maintain consistency of the routing tables in every node of the network, periodic updates of the tables are generated when significant changes of the topology happen. Through these updates and each node's view of the neighbouring nodes, it can calculate the next hop for every destination taking into account relevant metrics. These metrics can be anything from number of hops to available bandwidth and link delay or anything else that depends on the application domain.

Since one of the characteristics of ad hoc networks is the frequently changing topology, the entries of the routing tables in DSDV tend to change quite often. This requires frequent updates to ensure that every mobile node can almost always locate every other mobile node in the network. The data that are broadcast by each mobile computer will contain a new sequence number and the following information for each new route:

- The address of the destination.
- The number of hops required to reach the destination.
- The sequence number of the information received regarding that destination, as originally stamped by the destination.

Tagging every update with a new sequence number from the transmitter, is done to distinguish between stale update packets and valid ones. The sequence number is a monotonically increasing number that uniquely identifies each update packet from a sender node. Therefore, as update packets propagate around the network and nodes receive update packets from the same originator, the most recent ones (with the highest sequence number) are always used for routing decisions. When the sequence number between an entry in the route table and an update packet is the same, then the one with the lowest metric is used.

2.3.2 Responding to Topology Changes

Mobile nodes cause broken links as they move from place to place. These broken links may be detected by the MAC layer or by the absence of broadcast packets from neighbouring nodes. Broken links are described in DSDV as having a metric of infinity (∞). When such a situation occurs, all routes through the broken link are tagged as having a ∞ metric and their sequence number is updated. Since this is something that could effect overall network performance, it must be propagated as soon as possible and indeed DSDV immediately broadcasts an update packet containing the broken routes. This is a special case where some node other than the originator of the update packet is allowed to change the sequence number. Sequence numbers generated to indicate an ∞ metric to a destination will be one greater than the last sequence number received from the destination. Subsequent nodes that receive these packets will trigger immediate broadcasts and eventually the result is going to be the removal of all the entries that use the broken link from the route tables across the network.

2.4 Summary

This chapter has reviewed three different routing protocols for mobile ad hoc networks. In fact, only DSR and DSDV can be considered protocols with the traditional sense of the word. They are defined rigorously, with detailed specification of their algorithms, the structures used and the protocol headers that they utilise. LAR on the other hand can be considered as an optimisation to the basic DSR protocol with the aim of reducing the overhead during route discovery. All of the three protocols have been subjected to testing through simulation, though the different scenarios under which the protocols were tested, could stress scalability issues more.

We have chosen to review a *reactive* and a *proactive* protocol, so that the analysis of mobility and direction in the following chapter will be put into the context of these protocols.

Chapter 3

Analysis and Design

This chapter discusses the idea of using mobility and direction as the basis for a routing algorithm in Mobile Ad Hoc Networks. What we seek to exploit, is the "movement experience" that nodes build up as they roam around the network. Provided that there is some way of obtaining nodes' position and then representing this movement (essentially a series of geographical coordinates through time) we will look at different algorithms that could then be used in the context of a protocol to improve routing. Our aim is to try and devise a *Directional Metric* through the evaluation and analysis of the salient properties that govern movement and then place that in the context of proactive and reactive routing schemes as defined in chapter 2.

Section 3.1 introduces the concept of Direction Based Routing and discusses relevant concepts and definitions associated with it. Section 3.2 deals with the concept of direction of a moving object through space and time and identifies problems and ideas that surround the term *direction*. Section 3.3 provides different metrics that can be used to describe the movement of a participating node. Section 3.4 looks at the different issues that are involved in integrating one of the metrics that were proposed in the previous sections to an actual protocol. Finally, section 3.5 summarises the ideas presented in this chapter.

3.1 Background

Some of the concepts that this thesis deals with, have already been introduced with the review of the LAR routing algorithm in section 2.2. Direction Based Routing extends the idea of exploiting location information a little further, so that hidden information or "patterns" regarding movement can be used to facilitate routing. When mentioning *mobility pattern* from hereafter, it will mean the arbitrary movement of an object in a two dimensional terrain, over a time period. Stationary objects can be considered as a degenerate case of a mobility pattern with zero velocity.

Understanding mobility patterns of moving nodes is difficult to do in a generic fashion. Some research has been done into producing mobility models suitable for simulating performance of ad hoc networks. Sánchez and Manzoni in [PMH] have defined mobility patterns as either *traces* or *synthetic*. Traces are patterns that are extracted from real life movement (e.g. a mobile phone carrier could provide the motion pattern of users in a certain area). However the use of traces raises problems. Traces tend to be domain specific and different traces can exhibit completely different motion patterns making it difficult to provide generic testing scenarios. Traces can also be difficult to create and obtain. Thus, sometime different types of modelling are required. In comparison synthetic models use certain algorithms to offer a realistic real life motion. Most protocols for ad hoc networks conduct their simulations using synthetic models.

3.1.1 Associated Concepts

An example illustrating the key concepts and the problems that are related to Direction Based Routing is shown in figure 3.1. Node **S** wants to transmit data to node **D**. Assuming that both intermediate nodes, **a** and **b** are inside the transmission range of **S**, then **S** can choose to transmit data destined for **D** through either **a** or **b**. In a case like this, the choice of an intermediate node can happen randomly or

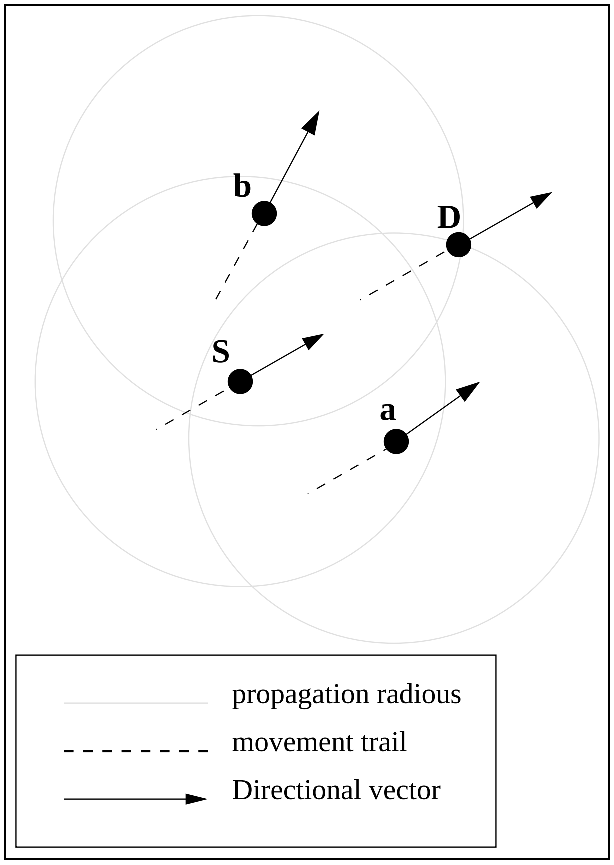

Figure 3.1: Direction based routing in an Ad Hoc Network

as a result of a decision metric.[1]

If there was a way to characterise the movement pattern of a node and assuming that the position of the destination node is known, choosing node **a** instead of **b** would provide the extra benefit of having source, intermediate and destination nodes travelling in the same general direction. That would mean that a route breakage between the three nodes will probably happen at a later stage than if selecting node **b**. This example is obviously simplified to fit the purposes of illustration. In particular, it does not take into account velocity of nodes, potential obstacles, overall pattern of movement through a sufficient long period of time, etc.

Providing a directional metric is something that could function in both *proactive*

[1]Like the distance to destination or quality of link.

and *reactive* routing environment. In the latter case that would be accomplished during a route discovery phase by reasoning that an intermediate node has better chances of staying in line with source and destination nodes. In the former, positional information can be exchanged as part of state messages that are sent frequently by proactive protocols in order to compute new routes in a changing topology. In both cases, and especially the proactive one, care should be taken so that the routing overhead does not increase beyond a certain point. Both possibilities will be investigated during the design phase.

Further analysis of patterns, might provide enough information so that the focus of the protocol might shift from maintaining long-lasting routes to reduced flooding during route discovery (only nodes that fit a certain movement pattern could be allowed to forward a route discovery). Another alternative would be for the source node to decide not to transmit the packet to any intermediate node, but to carry the packet towards the destination.

Some of these ideas could only be suited and applied in marginal application domains. For example, the idea of carrying packets towards a destination instead of transmitting them to neighbouring nodes, might suit extremely sparse environments where traffic delay is not important, but *eventually* reaching the destination is the primary objective. In such an environment, whenever a data-carrying node comes into close vicinity with other nodes, there could be a period of direction identification between all nodes. Then, depending on past and present direction, data would be relayed towards nodes that have the highest possibility of reaching particular geographic destination.

Our implementation and further experiments will try to prove if accounting for mobility information can help routing protocols provide a more efficient route discovery mechanism. Increasing the longevity of routes as well as reduced packet overhead will be our primary objectives.

3.2 Understanding Direction

A central point of this dissertation constitutes the concept of direction and heading of a moving node. In fact, in order to make an intelligent routing decision based on direction, one has to understand some basic properties of direction and define a metric that can be used when we wish to somehow quantify a mobility pattern. The rest of this document assumes that any node participating in an ad hoc network (or any moving object in fact) has the capability to know its location by taking samples of this data at certain time intervals. Every node can also transmit information regarding its coordinates when queried by other nodes.

As mentioned by Lin et al. in [LS] variations of directional routing algorithms have been recently proposed [BCSW99, YC, YBKV98, KSU99] that are shown to perform significantly better than algorithms and protocols that do not use topographical information for their routing purposes. Most of these protocols however, tend to use only positional information to perform routing more efficiently. Also, a closer look of the aforementioned protocols reveals that the term *direction* is used to denote movement towards a particular point or area. The actual direction of movement (e.g. North, South, East, West) or else the heading of the source, destination and intermediate nodes is generally not considered. In other words, when topographical information is taken into account for routing purposes, source and target nodes are pinpointed by their location but their mobility trace is not regarded.

When we try to adapt this directional knowledge for routing packets between moving nodes in a MANET, we are faced with some inherently complex and not well understood problems. On one hand, movement patterns and their "characteristics" are likely to affect what directional information we can extract and to what extent we can trust this information to be of value. An object moving in a straight line for a specified period of time presents less of a challenge for predicting its future move than an object moving in a Brownian-Random way. Moreover, it is important to identify certain attributes and characteristics that can be used to understand different types of movement. Simply exchanging information about a node's current heading unlikely to be sufficient. It could be the case that during this data exchange the node made a spontaneous move towards a totally different direction than the one it will continue to follow in the future. Capturing and transmitting this will result in routing decisions that are not accurate since they are based on a transient manoeuvre of the node and it is not expressive enough to rely on.

What is needed is a set of metrics that can quantify the mobility pattern of a node and give an estimate of its past and future route. Knowledge of this past route is essential if we want to avoid relaying packets to nodes that are for example moving in loops and thus are unable to transfer packets very far. It also provides us with some data so that we can reason about the node's future route.

3.3 Directional Routing

Devising an algorithm that gives a good approximation of an object's directional behaviour in every possible mobility pattern is beyond the scope of this thesis. Instead, we will try to identify certain properties that can be parameterised depending on the application domain. Three qualities of movement are identified and analysed in the following sections. They result from the general intuition regarding movement in that it has a *heading*, it traverses a certain *distance* from a starting point and also covers a certain *area*. The movement of every object can be described in a four-dimensional plane as a series of positional and time coordinates. This is how we will approximate movement in our analysis and simulation. Every movement also has a starting and an ending point as well a starting and an ending time, except for objects moving ad infinitum.

Between any two given points we can represent movement using vectors. The heading of the vector represents the movement's direction while the vector's magnitude represent the distance travelled. In the following paragraphs we propose metrics to approximate the mobility properties mentioned above by quantifying the most important characteristics. For each one of the three properties, we will discuss possible algorithms and choose the one which is going to be used in the design and implementation of the Direction Based Routing protocol.

3.3.1 A Metric for Heading

The aim here is to identify an algorithm that could approximate a node's future position provided that we have enough information about its past movement trace. It should be noted that if a node's mobility pattern is truly random, then trying to identify where the node is going to be next carries no real significance. However, common sense may suggest that in a suitable time period, certain types of movement may disclose patterns[DTMB99].

Time is an important factor in prediction, especially when trying to approximate something as volatile as movement. It can influence a predictive algorithm for direction in cases like,

- Where the input to the algorithm consists of the node's past trajectory. In this case we need to be aware of the time where we took the samples of the node's position.
- How long a predictive result is valid for.

Different application domains can have surprisingly different requirements for taking positional samples and for valid time frames of prediction results. For our discussion, we assume that coordinates are measured at almost identical discreet time intervals and the result is valid for the same period of time.

Solution 1: Rate of Angular Change

Let us assume that we have a movement pattern like the one depicted in figure 3.2. We measure the coordinates of the node in points A,B,C,... at times t_1,t_2,t_3,...,t_n.

One way to express the change of direction is to use an equation like the following,

$$a_1 = \frac{|\phi_2 - \phi_1|}{t_2 - t_1} \tag{3.1}$$

or in the more general case,

$$a_t = \frac{|\Delta\phi|}{\Delta t} \tag{3.2}$$

where a_t is the *rate of angular change* expressed in degrees per second. Since the result is an absolute value, we can only expect this metric to be useful when we want to know how volatile a movement is over a period of time. This will give us an average of a node's change of direction between two different time measurements. A comparison of the *rate of angular change* between nodes will not tell whether the nodes are heading towards the same destination, but express the tendency they have to follow the same pattern of changes in their headings. Furthermore, it will not give us any estimation for any future heading, but could prove useful for analysing past changes of direction.

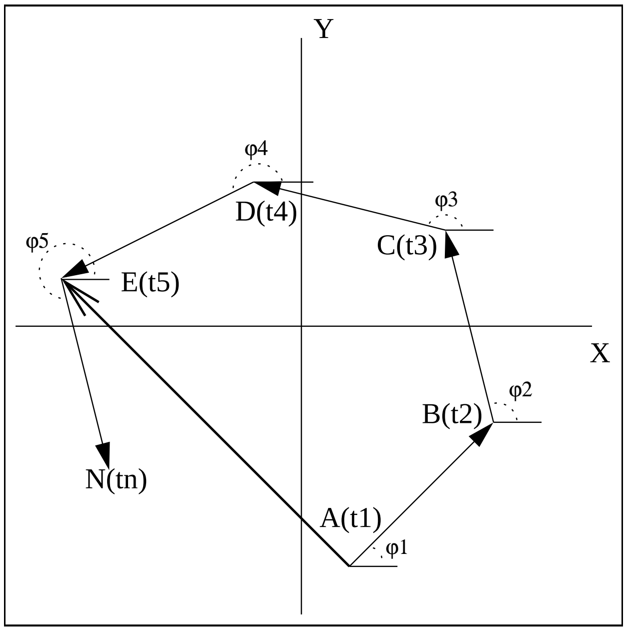

Figure 3.2: Rate of angular change in a movement

Solution 2: Weighted Average

An alternative way to try and predict changes in the node's direction would be to take an average of the past n headings. The formula should be parameterised in a suitable way so that it can be ratified for different application domains. That can be achieved by assigning different weights to past and current headings, which would depend on the significance we would like to attribute to the current direction compared to the past route. It could also be a way (albeit a simple one) to dynamically adapt the prediction functionality when mobility conditions change. Such a metric of direction can then be modelled as,

$$h = \frac{\sum_{i=1}^{n} W(i)\phi_i}{n} \tag{3.3}$$

where h would be the result in either degrees or radiants, W(i) a weight function with $\sum_{i=1}^{n} W(i) = 1$ and $\phi_1,\ldots,\phi_n$ the angles of the corresponding directional vectors.

In figure 3.2 for example, the approximation to the direction of the $\overrightarrow{EN}$ vector corresponding to angle ϕ_5 can be calculated by substitution to the previous equation like,

$$\phi_5 = \frac{0.5\sum_{i=1}^{3} \phi_i + 0.5\phi 4}{4} \tag{3.4}$$

giving an equal weight to past and current headings. A problem with this equation is the fact that in order for a comparison to take place we need to ensure that an identical number of positional samples was used for every node. Otherwise, comparison will produce skewed results because if n is not equal for every node, it will give a different weight to the measured angles.

Solution 3: Time Dependent Weighted Average

One of the shortcomings of the previous solution is that it does not take into consideration the time a node has been following a certain path. That means that angles will be given the same weight regardless of the amount of time. This could be remedied if the function assumed that positional coordinates are taken at identical timing periods, which again prevents the algorithm from adapting to changing parameters (e.g. changes in speed could result in changes in the period that positional samples are taken).

A better solution would be to integrate the rate of change function with the weighted average, giving an equation like the following,

$$h = \frac{1}{t_n - t_1} \sum_{i=1}^{n-1} \phi_i (t_{i+1} - t_i) W(i) \tag{3.5}$$

where h, W(i) and ϕ_1,...,ϕ_n are the same parameters defined in the equation 3.3, while $(t_{i+1} - t_i)$ is the time the node traced the path defined by angle ϕ_i.

Critique

Solution 1 described above is not attractive because it is not sufficiently expressive. It could be used to predict whether we should anticipate small or large directional changes but it cannot actually give an estimate position. Unlike solutions 2 and 3, solution 1 does not take detailed past positions into account, only the starting point and the last recorded position. Solution 1 is also an absolute metric which makes it considerable harder to compare headings between nodes. Solution 2 on the other hand, provides a good metric when comparison is needed but does not encapsulate the notion of time in the calculations. This might result in inconsistent values coming from the fact that we assume each heading to be followed for an equal amount of time. The fact that it also produces an average by division of the number of sample positions creates another problem, since this needs to be the same when nodes compare results. Solution 3 can be a good candidate for implementation, being the only one that can give a good estimate taking as input timing requirements and directional changes.

The problem with solution 3 arises when a value is required as a comparative metric. If we want to measure similarity in direction between nodes, then an extra step will be required after calculating equation 3.5. Unlike metrics like bandwidth, delay, distance, etc. for which values can be compared on a minimum, maximum function, conversion from different values of the aforementioned function would require another function to normalise the results and return a *similarity* metric (e.g. if results are within a certain threshold value then assign a certain number).

3.3.2 A Metric for Distance

As was indicated in section 3.3 another important factor for constructing a relevant routing metric is the distance travelled by a node during a given period of time. What this distance tells about the behaviour of a node, is things like,

1. The *activity* of a node. This is a property closely related to velocity. Assuming identical time periods of measurements, we can distinguish between the degree of activity of nodes depending on the distance covered.

2. By comparing the total distance travelled between two points with the magnitude of a vector connecting the starting and ending points we can reason about the quality of movement.

Point 2 is especially important for providing a good comparative metric. We can formulate it in an equation like the following,

$$d = \frac{\|\vec{R}\|}{\sum_{k=1}^{n} \|\vec{r}_k\|} \qquad \text{where } \|\vec{r}_k\|, \|\vec{R}\| \in \mathbb{R}^2 \tag{3.6}$$

and $\|\vec{r}_k\|$ is the distance vector between two positions, while $\|\vec{R}\|$ is the resultant between points 1 and n.

What this equation expresses, is how well the resultant vector approximates the constituent vectors. In other words, it says how determined was a node in keeping a steady course during the movement. We will name this factor the *determination factor*, where a large fraction would mean that the movement is more or less close to the resultant vector while a small value would suggest a movement that fluctuates around the resultant. The movement depicted in figure 3.2 between points A and B for example, would present a larger fraction than a movement between points A and E in figure 3.3.

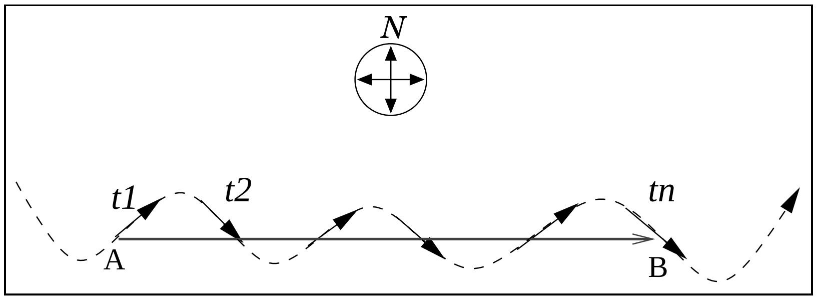

Figure 3.3: Approximating direction by using a resultant vector

This *determination factor* presents a metric whereby the quality of a movement can be characterised. It is also a time independent function, though again measurement intervals are of importance and specific to application domains that can influence the quality of the routing decisions made.

3.3.3 A Metric for Area

Complementary information to distance is also the area that a node is covering when moving. This could be a further aid in order to quantify proximity to a target node and for making a more accurate prediction of how a node's mobility pattern has evolved in the past. For random movement patterns developed for studying ad hoc networks in simulated environments, this might provide only limited knowledge but it could prove very useful when ad hoc networks perform under real life scenarios.

Using routing algorithms that rely on an area coverage is an approach that is discussed by Navas et al. in [NI97]. They have devised an algorithm for routing in fixed networks that is based on geographical areas. A router is responsible for a certain area that is defined by bounding polygons and a packet finds its destination by polygon intersection between the packet's coordinates and the area that the router is responsible for. The same mechanism can also be employed but for a different purpose in Direction Based Routing.

Bounding Polygons

Having constructed a bounding polygon from the trajectory of a node, allows some assumptions to be made on the suitability of the node to carry or transmit a packet to the desired geographical location. In general, a bounding polygon that covers a large area means frequent changes of direction. This can provide useful input which can be used in conjunction with the distance metric as described in section 3.3.2 to provide more accurate approximations of mobility.

However, further analysis should be conducted on the merits and the actual benefits that are provided by accounting covered area in the calculation of the directional metric versus the computational cost in calculating bounding polygons. If the bounding polygon can be obtained in a computationally inexpensive way it could then be used in addition with the distance metric. Another interesting characteristic of bounding polygons, is the dependence of the calculated area on the number of vertices (in our case location measurement points). Ideally, we would like changes in the number of location points measured not to affect, or affect proportionally the area that is calculated around these points.

This is an area that has not been adequately researched, as it was considered peripheral to the project's aims. It is a field though that worths exploring for future work in location aware and direction based routing.

3.4 Selecting a Directional Metric

The decisions that have to be made concern the type of directional metric that will be used, the type of protocol the metric will be fitted in (either *proactive* or *reactive*) and, after choosing the type, the exact protocol that will be used. Hopefully the above choices will direct us into a clean and straightforward design on how the metric will function with the protocol and how it will be calculated with minimal overhead. Although some of the choices are interdependent, this leads to a nice separation of sections, with each of the following discussing one of the above items.

3.4.1 Choosing a Metric

For this thesis, the selection of an appropriate metric depends on several factors. Time constrains mandate that a relatively straightforward metric be chosen for implementation, without sacrificing the effectiveness of the algorithm and the efficiency of the protocol. A metric that can fit into a preexisting protocol would be desirable, since the design and implementation of a fully fledged ad hoc routing protocol is outside the scope of this dissertation. Regarding algorithmic effectiveness, it is desirable that the metric leads directly to a quantity that can be used as a comparative value during route selection.

For the above reasons a decision was made to use a variation of the distance metric described in section 3.3.2. As mentioned in the beginning of this chapter, the goal is to find a metric that describes similarity in the direction of moving nodes. By knowing the directional vectors of the moving nodes, we can divide the magnitude of the overall resultant of these vectors with the sum of the magnitude of each individual vector. In this way we get a fraction that as it approaches 1 it means greater similarity in direction. Figure 3.4 helps to clarify this point. Vectors $\vec{r}_1$, $\vec{r}_2$, $\vec{r}_3$, $\vec{r}_4$ represent the movement of individual nodes, while vector $\vec{R}$ is the resultant. By dividing the magnitude of $\vec{R}$ with the sum of the magnitudes of the constituent vectors, a fraction close to 1 will be calculated, since all of the individual vectors face the same direction and belong to the same quadrant. The mathematical notation of this description is the following equation,

$$d = \frac{\|\vec{R}\|}{\sum_{k=1}^{n} \|\vec{r}_k\|} \qquad \text{where } \|\vec{r}_k\|, \|\vec{R}\| \in \mathbb{R}^2 \tag{3.7}$$

with $n = 4$ for the example in figure 3.4.

3.4.2 Choosing a Protocol Type

The decision for the selection of a protocol type was interleaved with the decision for the metric in section 3.4.1. Although the introduction mentioned that the metric should work in both types of protocols, early on it was made clear that such

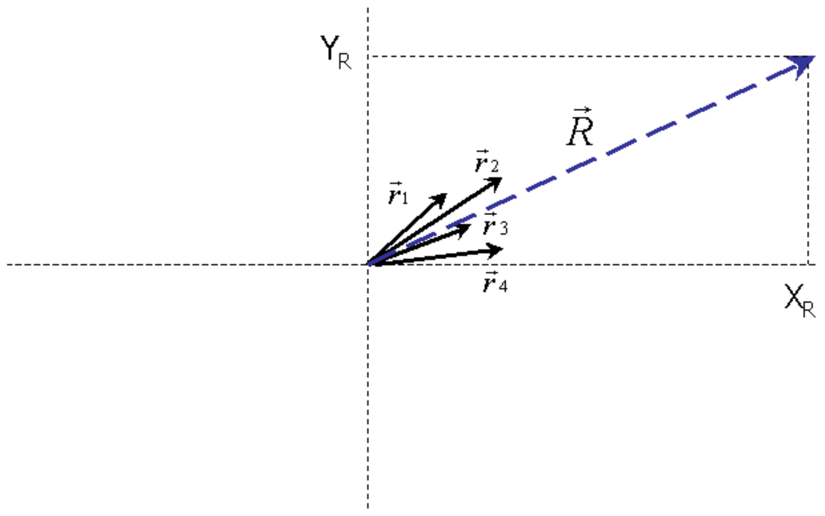

Figure 3.4: Choosing a Directional Metric

a metric would cooperate better with a *reactive* algorithm than a *proactive* one. The most important reason for this, is the tendency of proactive protocols to hide the complexity that the mobility of nodes introduces under the frequent exchange of messages. Clearly, seeking to maximise the performance of routing by taking advantage of mobility patterns under a protocol that tries to hide this factor is difficult. Even if this is disregarded, the fact that *proactive* protocols operate on a next-hop fashion, makes it difficult to gather, analyse and extract results on nodes' directional similarity. We found no suitable place to merge the vectors of the source, intermediate and destination nodes so that a metric could be calculated. The possibility that such a metric can be used with proactive protocols remains an open issue and could be the subject of further research.

Looking at on demand protocols, a simple algorithm was realised that could provide the ability for nodes to make routing decisions based on the directional metric. The protocol would perform the following steps,

1. Issue a discovery to find a destination.

2. Have each node encapsulate its directional vector together with its address in the discovery message.

3. Destination node receives the discovery message, appends its own vector information and sends back a reply.

4. Source node receives (potentially) multiple replies from the destination with vector information included and stores them in its cache.

5. When a packet is ready to be sent to a known destination, calculate the metric for each route in the cache and choose the one with the highest fraction.

Figure 3.5 helps illustrate the process of obtaining vector information during the route discovery procedure. The notation `<node number>|dx/dy` implements step 2 that when each node adds its address in the ROUTE REQUEST message, it also includes its vector information in the *vector representation* format (this format is further explained in section 4.4).

3.4.3 Choosing the Protocol

As mentioned already in chapter 2, reactive protocols include but are not limited to the Dynamic Source Routing (DSR) protocol, the Ad hoc On demand Distance Vector (AODV) and the Zone Routing Protocol (ZRP). Between them, ZRP is really a hybrid approach, dividing the network area in to zones, with intra-zone routing taking place in a proactive fashion while inter-zone routing is done reactively. Although its use as a base protocol has not been exhaustively researched, we think that the same problems will be encountered as when using a purely proactive protocol.

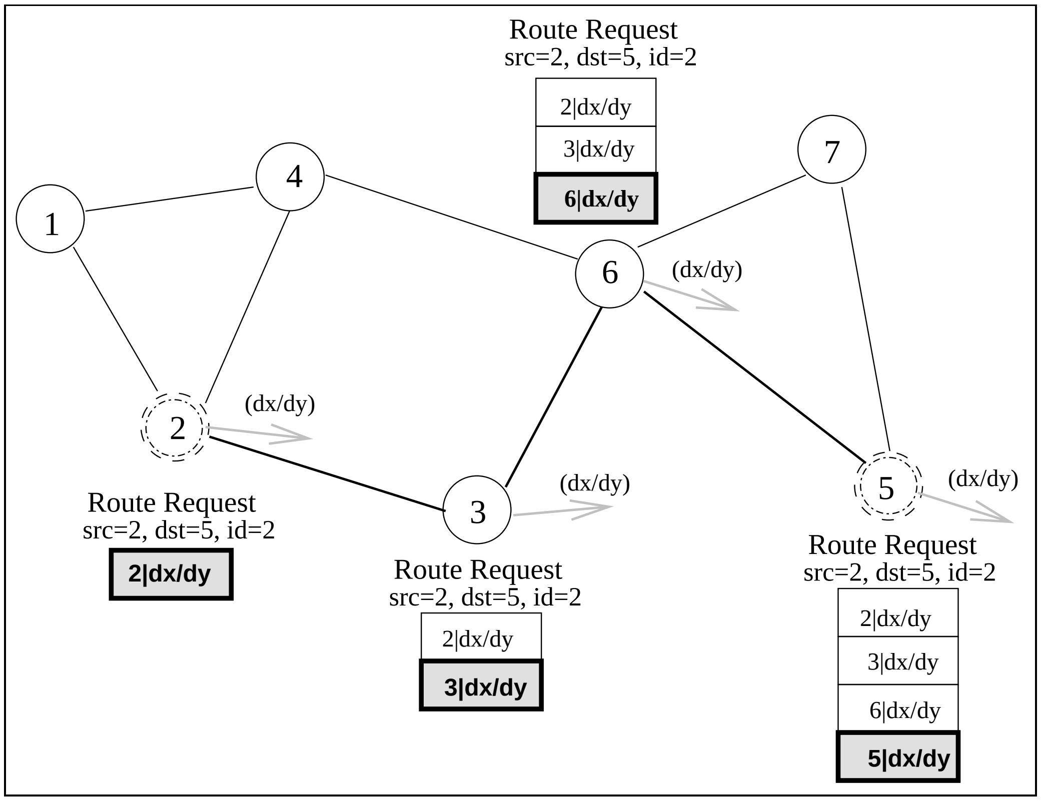

Figure 3.5: Appending vector information during route discovery.

The choice between DSR and AODV was based on several factors. Although both protocols operate on-demand the source routing architecture of DSR provided an extra benefit. The metric would have to be advised only in the source node before a route is tagged into the outgoing packet. In the case of AODV, an on-demand, next-hop protocol, the metric would have to be calculated in each node. The fact that AODV employes routing tables for next-hop routing is also a problematic factor, because of the reasons that were mentioned before.

Apart from an implementation in a simulation environment, DSR also features a real implementation under the FreeBSD Unix operating system. This is an extra motivation for the use of DSR, since it means that apart from a simulation, we could have a code base for a future real implementation of the Direction Based Routing algorithm using wireless LAN cards and GPS modules. The author does not know of any existing or future plans for a real implementation of AODV.

The benefits of AODV include native support for multicast and broadcast as well as simulation results that show that AODV scales gracefully for large population of nodes. Tests conducted with the AODV protocol, employed 500 nodes, while its authors Perkins et. al. claims that results with AODV in node sizes of up to 10,000 nodes have shown good delay and packet delivery measurements [Per01, page 207]. Multicast and broadcast support are outside the scope of this thesis, but scalability is an important characteristic, especially since tests under DSR have only been conducted with no more than 50 nodes. Still, the merits of DSR outweigh those of AODV.

3.5 Summary

In this chapter we identified techniques and metrics that can be used to characterise the quality of a movement pattern in order to facilitate more efficient routing in mobile ad hoc networks. The analysis and the discussion was not tied up to specific to routing algorithms, but was put in the more general context of *movement approximation and prediction.* This gave more flexibility over the design, where a base protocol, a metric and a generic algorithm were selected.

Three properties of a movement were identified and alternative ways of representation were suggested. The decisions made in the previous sections are outlined below.

Heading

Based on feedback from the angular value of previous distance vectors, an algorithm was devised that could be used for predicting and comparing directions in a generic way in an ad hoc network. Three solutions were proposed; a rate of change function, a weighted average of direction and a time dependent weighted average. Equation 3.5 was found to be more expressive because it encapsulates timing and angular parameters giving a more detailed metric. Computational cost is a remaining issue and could be dealt with in future work.

Distance

Whilst the heading metric can be used to approximate the future position of a node, the distance metric expresses the quality of the node's past movement. When this is used with multiple nodes, it can provide a metric for measuring similarity in direction. This *determination factor* will be used in the implementation to show the resemblance in the movement of nodes that participate in a source route towards a destination.

Area

The use of a metric regarding the area covered by the route a node traverses, can be used either complementary to the distance metric or as an optimisation in certain application domains. Bounding polygons offer a solution for defining and calculating an area given the coordinates of its vertices but it remains to be seen whether their use is justified for routing decision.

Chapter 4

Implementation

This chapter describes the implementation of the ideas discussed already in chapter 3. Having provided an analysis of the aims, as well as the algorithms that will be used, we test these in a simulation environment. As was mentioned earlier, after studying both *proactive* and *reactive* protocols, a decision was made to use Dynamic Source Routing as the basis in which to fit the algorithm. The *ns* [MF] network simulator was chosen for having an existing mature implementation of the DSR protocol and an overall modular architecture using languages (C++, TCL) that the author is fairly well accustomed to.

Section 4.1 provides details into the workings of the *ns* simulator. In section 4.2 we describe the changes made by the CMU Monarch group to add wireless mobile capabilities to *ns*. Section 4.3 overviews the implementation of the DSR protocol. Section 4.4 presents the extensions that were necessary in order for the DSR protocol to accommodate the Direction Based Routing algorithm. Finally, section 4.5 concludes the implementation chapter.

4.1 The *ns* Network Simulator

4.1.1 Overview

The *ns* network simulator is a *packet-level*, *discrete event* simulator. It is an open source software product, mainly used by the research and academic community to validate research in most areas of networking. It supports both wired and wireless networks from the Data Link to the Application layer (Layers 2 to 7 of the OSI protocol stack). It also provides the framework and the implementation of unicast and multicast algorithms for routing, TCP (Transmission Control Protocol), UDP (Unix Datagram Protocol), CBR (Constant Bit Rate) and congestion control for the transport layer as well as web caching and multimedia algorithms at the application level. The simulation engine is also available in a multitude of architectures and operating systems including flavors of BSD (Open-BSD, Free-BSD), Linux, Solaris, HP, SGI and Windows 95/98/NT.

4.1.2 Architecture

The overview that follows will concentrate on the latest version of the simulator, now called *ns-2*. From here after, whenever we refer to *ns* we will mean version 2. *ns* is written in an object oriented fashion for maximum scalability and extensibility. It offers a modular approach which splits the functionality of objects in two languages, C++ and OTcl.[1] The duality of the language choice offers a design where separation exists between control operations on the state of the simulation and the low level data manipulation. OTcl, a scripting language, is used when the parameters of the simulation need to change (e.g. varying some protocol parameters) without the extra burden of recompiling the program. On the other hand, efficient packet and data manipulation needs a system language, which is why C++ was chosen for. In general, things that have to do with configuration parameters and setup are written

[1]OTcl is an object oriented extension to Tcl from David Wetherall at MIT/LCS.

to be manipulated in OTcl while lower level algorithms and protocols are written in C++ for maximum performance.

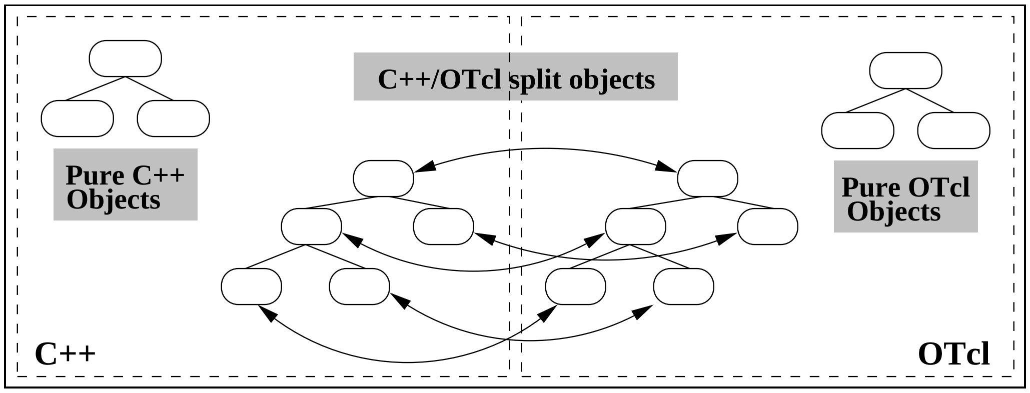

Figure 4.1: C++ and OTcl usage in *ns*

Figure 4.1 shows how the two languages are bound together in the simulator to create a reusable and extensible framework without compromising efficiency. Although there are two different object models, one in C++ and another one in OTcl, most of the C++ object hierarchy has a corresponding OTcl hierarchy which allows the objects to change behaviour at run-time through the OTcl interpreter. C++ variables can be bound into OTcl, thereby enabling the protocol designer to quickly change aspects of some algorithm and re-run the simulation.

Components

Here we will describe some of the most important components that make up the simulation environment and can be used to extend it. Although most of these components correspond to a C++/OTcl class, there are some exceptions where a component's functionality is provided by a class hierarchy.

- **Classifier**. When a node in *ns* receives a packet, it tries to examine the packet's fields (e.g. destination address and port number) and then map these values to an outgoing interface object that is the next downstream recipient of this packet. This function is performed by a *classifier* object. A *classifier* object acts as a filter and tries to match each packet against some logical criteria, retrieve a simulator object based on the matched results (e.g. the routing layer) and let this object deal with the packet. Classifiers include unicast, multicast, flow and address.

- **Agents**. An *agent* object is the 'core' of *ns.* They represent endpoints where packets are usually constructed or consumed and are mainly used for the implementation of the different protocols. The *agent* object has an implementation partly in C++ and partly in OTcl. Examples of *agents* already implemented in *ns* include variations of the TCP protocol (Reno, Vegas), the UDP, mobile ad hoc routing protocols, etc.

- **Trace**. *Traces* are special objects that are used to store information about each packet that is sent, received or dropped during run-time, so that these data can then be analysed. This information can be displayed during the execution but is usually written to a file and post-processed after the simulation has finished executing.

- **Queues**. *Queues* are used for buffer management and packet scheduling.

- **Mathematical Support**. This includes support for a random number generator, calculation of integrals and samples objects that support the computation of mean and variance statistics for a given sample.

ns also includes support (as a separate program) for visualising the results of a network simulation. The program is called *nam* (network animator) and is written in Tcl/Tk. It is fed results produced by the *trace* component and is able to visualise both wired and wireless networks showing nodes, links and the flow of packets with parameterised level of detail.

4.2 The CMU Monarch Extensions

This section describes some of the modifications and additions made to the *ns* simulation by the CMU Monarch Project.[2] This happened in order to support a number of ad hoc routing protocols including DSR. The description here is deemed necessary because *ns* did not support wireless networks and ad hoc routing protocols before the contribution of the Monarch project and because it will provide the base for the discussion on the design and implementation of the routing algorithm of this thesis. For a more detailed description on the changes, see [MP].

The CMU Monarch project contributed changes to the *ns* in the physical, link and routing layers. With these changes it is possible to construct simulations for wireless LANs or mobile ad hoc networks. Specifically the components that were implemented are the following:

1. A Radio Propagation Model, Omni-Directional antennas and a shared media network interface that approximates Lucent's WaveLAN direct-sequence spread-spectrum (DSSS) radio interface at the physical layer.

2. At the link layer the specification of the IEEE 802.11 Distributed Coordination Function (DCF) was implemented.

3. Four ad hoc routing protocols were implemented at the routing layer (DSR, DSDV, TORA, AODV).

Another important component added by the CMU extensions is the `MobileNode` super class. It inherits some of its functionality from the basic `Node` class, which represents network nodes in *ns*. The `MobileNode` class is responsible for keeping track of its position and velocity as it roams in the network. A layout of the functionality associated with a mobile node in *ns* is given in figure 4.2.

[2] `http://www.monarch.cs.rice.edu`

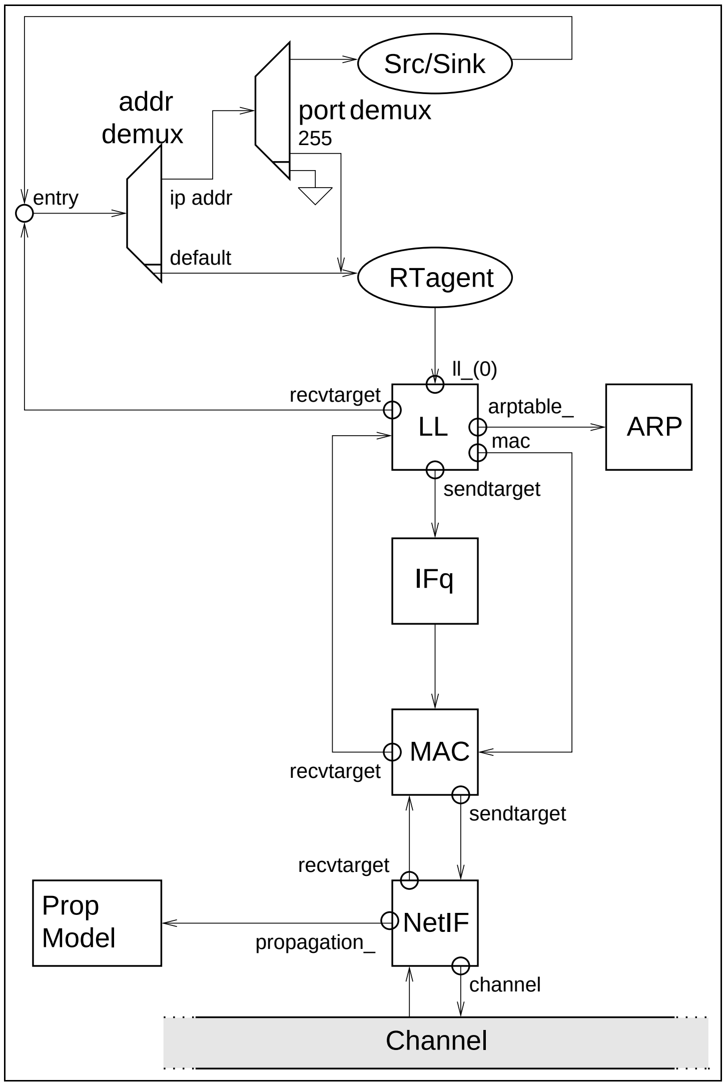

Figure 4.2: Schematic of a mobile node (source: [MP])

Incoming Packets

Starting with the lower layers of the figure, each *mobile node* is attached to a *channel*, which simulates the medium that transfers data between nodes. All *mobile nodes* participating in a wireless network are connected to this channel by means of a *network interface*. Each packet that is sent to the network, is distributed by the *channel* entity to all *network interfaces*. These interfaces then use a *radio propagation model* which takes into account signal strength and distance between source and destination nodes to determine if the target node is able to receive the packet. Once a packet is received by the *network interface*, is is passed to the MAC layer which will test whether the packet was received collision and error free. The *link layer* then takes control of the packet and hands it off to an entry point. There, an *address demultiplexer* will decide if the packet's destination is the current node, in which case the packet is received by the application layer, otherwise the packet is sent to the *routing agent* which will decide and tag the packet's next hop towards the destination.

Outgoing Packets

Packets that are generated from the application layer are passed down to the *address demultiplexer*. The address demultiplexer will check the packet's destination address and if it is destined for the same node, it will pass the packet to the port demultiplexer which has the task of assigning packets to different agents listening at different ports. Most packets generated by a source node will be destined for a different node, in which case the address demultiplexer hands them down to the routing agent. The routing agent will tag the packet with the next hop towards the destination and send the packet to the *link layer* module. Another contribution from the CMU Monarch project was an ARP component that the *link layer* will use when the address of the packet is an IP address. After the ARP request has been completed the packet is pushed in an *interface queue* (IFq), which gives priority to control packets over normal ones. The Media Access Control (MAC) object pulls packets from the *interface queue* and sends them to the *network interface* object which stamps each packet with properties like signal power and node position and then passes them to the channel object.

4.3 The DSR Implementation

As already mentioned in section 2.1, DSR is a simple ad hoc routing protocol that employes quite a few optimisations, described in section 2.1.4, to reduce route discovery delay and channel efficiency. Table 4.1 shows what optimisations are supported by the current implementation of the DSR and which one of these had to be disabled for the implementation of the Direction Based Routing algorithm. An explanation of why some of these had to be disabled will be given on the next section.

The basic architecture of the `MobileNode` super class in *ns* had to be changed to accommodate some of the optimisations. The `MobileNode` class incorporates both the address demultiplexer and the routing agent into a single object (the `DSRAgent`). This is done in order to support some extra functionality, whereby the node could extract or insert extra routing information in data packets that would otherwise not flow through the routing agent. The implementation of DSR supported by *ns* is a fully functional implementation based on the published specification that tries to follow closely the architecture and the structures that would be used in a real implementation.

Cache Structure

The DSR simulated implementation uses the same cache structure and request table as the real implementation that runs on FreeBSD. The route cache is populated by ROUTE REPLY messages that the node receives and also by listening to traffic generated by neighbouring nodes and then extracting from this traffic paths to destinations. For this reason, DSR uses a primary and a secondary cache. Routes that are learned as responses to ROUTE REQUEST messages are automatically inserted into the primary cache, while routes learned while the node is in promiscuous mode are inserted into the secondary cache. The implementation uses a set of rules to upgrade routes from the secondary to primary cache and to replace entries when the cache becomes full in FIFO order.

Variables	*Description*	*DB-DSR*
`dsragent_snoop_forwarded_errors`	snoops forwarded errors	`Enabled`
`dsragent_snoop_source_routes`	snoop on source routes	`Disabled`
`dsragent_reply_only_to_first_rtreq`	reply only to first Route Request	`Disabled`
`dsragent_propagate_last_error`	propagate error messages on the next Route Request	`Enabled`
`dsragent_send_grat_replies`	send gratuitous replies to affect route shortening	`Enabled`
`dsragent_salvage_with_cache`	when a route breaks consult the cache for another route	`Enabled`
`dsragent_use_tap`	use promiscuous mode	`Disabled`
`dsragent_reply_from_cache_on_propagating`	answer from the cache before propagating Route Request	`Disabled`
`dsragent_ring_zero_search`	send a non-propagating Route Request first	`Disabled`

Table 4.1: Optimisations used by the DSR implementation

The cache structure that DSR utilises is a *path cache*. That means that every entry in the cache list will be occupied by a full path with the first element being the caching node. Implementing a *path cache* is simple and straightforward with an easy algorithm for navigating and searching each entry in the list. On the other hand, it does not offer great flexibility in cross-connecting path information in different entries. A cache structure like *link cache*, where paths to destinations appear as a unified graph, offer a better chance of obtaining alternative paths to a destination but with the price of increased overhead during search and more complex structures.

Request Table

The request table structure is used in DSR to avoid the problem of propagating multiple ROUTE REQUEST messages. Every time that a new ROUTE REQUEST message is received by a node, the address of the source of the message as well as the `sequence_number` are added to a table-like structure. Every subsequent ROUTE REQUEST is compared against the entries of this table and if its source address and `sequence_number` matches an entry in the cache, it is dropped. That helps avoid potential routing loops, since ROUTE REQUESTS will never be propagated twice.

4.4 Extending DSR

We will describe here the changes that were made to the core DSR implementation in *ns*, in order to accommodate a new metric for route selection. The steps of the algorithm were briefly outlined in section 3.4.2. The areas that were modified are outlined below.

- The DSR Routing Agent.
- The DSR Packet Header.
- The Route Request Mode.

- The Route Reply Mode.
- The Cache Module.

For the implementation, the latest available version of *ns-2* was chosen, `ns-2.1b8a`.

DSR Routing Agent

The routing agent in *ns* corresponds to what would normally be the component that incorporates the logic and the functional aspects in a real implementation of a protocol. An extra timer was added to the routing agent, so that periodically the coordinates of the node's position are stored in a circular buffer. The period for taking the positional samples was chosen as a uniform distribution between 0.75 and 1.5 seconds.

A modification was made so that during ROUTE REQUEST and ROUTE REPLY messages, each node would append its movement vector together with its address to the packet's header. This vector is essentially the distance between the last and the most recent entry in the circular buffer. The *vector representation*[3] was chosen for storing vectors in structures. It is a simple representation that requires storing the difference in the x,y coordinates between starting and ending point. This is not a positional representation but allows the extraction of the vector's magnitude and angle and is also convenient for calculating the resultant between vectors since it only involves the addition of corresponding coordinates.

It is the responsibility of the destination node of the ROUTE REQUEST message, to copy the addresses and the vectors of all the nodes to the ROUTE REPLY message. It also appends its own address and vector representation and sends the reply message back to the originator node. DSR has an optimisation, whereby a node can reply to a route discovery by searching its cache for routes to this destination. This was disabled in our implementation, as indicated in table 4.1, so that the

[3]Mathematically is represented with $\binom{x2-x1}{y2-y1}$ where $\binom{x1}{y1}$ is the start point and $\binom{x2}{y2}$ is the end point.

vector information obtained by the originating nodes would always be up to date.

When a ROUTE REPLY arrives at the source node, the path to the destination is stored in the route cache. This path is composed by a series of addresses along with vector information. In dense networks, we should expect multiple ROUTE REPLY messages to arrive to the originating node, each containing different paths to the destination. Our metric applies when for a given destination more than one possible paths to the destination exist. For every packet sent, the algorithm follows these steps,

1. Initialise hops to destination to some maximum and metric to zero.
2. Calculate the number of hops for a given path.
3. Calculate equation 3.7 where k is the number of hops from source to destination inclusive.
4. If the number of hops is less than the number stored, overwrite and tag this path as good.
5. If the number of hops is equal to the number stored, compare calculated metric with the one stored.
6. If the calculated metric is bigger then tag this path as good.
7. Repeat for each path to destination in route cache.
8. The path chosen will be the one with the smallest number of hops and highest similarity metric.

It is obvious from the above analysis that the primary metric is the number of hops towards a destination, while the directional metric is a secondary metric that takes effect when paths with the same number of hops to destination are encountered. Again, depending on the density of the network there are likely to be quite a lot of paths containing the same number of hops towards a particular destination.

DSR Packet Header

The DSR packet header was modified so that during the route discovery and route reply procedure, the header would be able to carry not only the IP or MAC addresses of the nodes that it is going through, but also the directional vector of the node. Assuming single precision floating representation, the overhead of this is eight bytes (four bytes per coordinate) per node that participates in the path. This is not a big overhead, even for a real implementation since it only occurs during the route discovery and the route reply phases of the protocol operation. Other than that no other changes were made to the DSR header.

4.5 Summary

In this chapter we have looked at the implementation details of the *ns* simulation environment and the way that DSR is implemented inside this framework. There is also a description of the changes that were necessary to be made to DSR in order to support the Direction Based Routing metric. The next chapter is going to deal with a comparative performance evaluation between the modifications presented in this thesis and the original DSR protocol as implemented by the CMU Monarch project.

Chapter 5

Experiments and Evaluation

This chapter describes the simulation model and the performance results produced, in order to evaluate the efficiency of the algorithm described in this thesis. Since Direction Based Routing is based on the DSR protocol it is natural to compare the modified DSR against the standard implementation.

When simulating mobile ad hoc networks, *ns* requires information about the *network topology* the description of a *mobility scenario* and a *communication scenario*. A mobility scenario describes the mobility pattern that nodes follow during the simulation period. The communication scenario refers to the application layer protocol used as well as the number of connections between nodes and the amount, size and rate of packets sent during a connection between a source and a destination.

Although a detailed description of the simulation model is in section 5.2, the rationale for choosing mobility scenarios is introduced here. The vast majority of performance results published in the area of MANET routing protocols use simulations with mobility scenarios based on the *random-waypoint* pattern. This is a simple pattern in which nodes are assigned in a random fashion a destination as soon as they reach their current one. In the case of *ns* this does not happen during simulation time, but rather such a pattern needs to be "fed" into the simulation engine before the execution. Each simulation can be run with several variations of the basic random-waypoint scenario, changing for example the velocity of the nodes, the pause time before they start moving towards a target, whether or not the nodes bounce off the edges of the specified area in the simulation, etc. The fundamental principle, however, remains the random factor which makes it difficult to make any kind of prediction about movement patterns by studying the past trace of a node.

More mobility scenarios are available that implement slightly more complex movement patterns, but the choice is for the random-waypoint pattern mainly because it is widely used. A secondary consideration is the fact that testing a routing algorithm that accounts for mobility with "ordered" patterns, might provide an unfair bias. This might have been one of the choices that led to the less than anticipated results in the simulation.

Section 5.1 gives a description of the aims of the evaluation and explains why particular metrics were selected. Section 5.2 discusses the simulation model and section 5.3 performs a statistical analysis and presents and evaluates the results on the metrics used. Finally, in section 5.4 a summary is given.

5.1 Measuring Aims

Section 3.1.1 listed one of the primary objectives for the algorithm as increasing the longevity of routes, thus producing less overhead in the form of route discovery traffic and therefore improving overall network throughput. The primary method to measure route longevity is to measure the average period of time for which routes are valid. Especially in DSR the lifetime of a route could be measured from the first data packet sent after the path to a destination has been inserted into the cache, until a ROUTE ERROR message removes this path from the cache. However, this approach was problematic in its implementation due to the way *ns* produces traces of operations on packets through the different network layers. It would take a complicated filtering mechanism to understand when the first packet after a ROUTE REPLY message has actually left the interface queue where it was buffered and got transmitted through the MAC layer. The same difficulty was experienced when trying to identify the last packet transmitted before an error message erases a specific path.

Instead, a more simplified, metric was used; the number of total ROUTE REQUEST messages per simulated run. This is the number of request messages that are actually broadcasted by the originator node. It does not include any discovery messages dropped or any of them that are forwarded from intermediate nodes. This metric is coarse-grained since it expresses a breakage on a route (which could consists of multiple alternative paths) rather than a path. Future work should concentrate on providing this type of analysis on a per path basis. Between two protocols operating in a source route on-demand way the one with the least amount of request messages would be the one that has chosen and maintained routes the longest.

Next metric evaluates the overall throughput of the protocol by means of packets received per data packet transmitted. Increased route lifetimes would result in fewer packets lost, thereby improving protocol efficiency. The final metric is channel usage which shows the number of control bytes transmitted per data byte delivered. It expresses the cost of the routing protocol in terms of bandwidth spent on management.

5.2 Simulation Model

The simulation uses an area of $1500m \times 300m$, while varying the number of nodes between 25 and 100. Each mobile host randomly starts its journey from a random location to a random destination with varying speed. Once a destination is reached, another destination is picked at random fashion. The mobile hosts move continuously with no pause time in between choosing waypoints. All simulations run for 900 seconds (simulated) for each scenario. The gathered data represents the average of five runs with identical traffic models, but different randomly generated mobility scenarios. Both DSR and the Direction Based DSR are run with same mobility and communication scenarios. Table 5.1 shows the fixed parameters that

Parameter	**Value**
Network Area	$1500m \times 300m$
Transmission Radius	$250m$
Packet Size	512 bytes
Application Output Rate	4 Packets/sec
Maximum Number of Packets	10000
Pause Time	0 sec
Number of Connections	20
Transmission Rate	2.5 Mbps
Application Protocol	CBR

Table 5.1: Fixed simulation parameters

are used during the simulations. In each run, 20 pairs of sources and destinations are selected randomly. Data packets are generated on a fixed rate of four packet per second until either the simulation ends or the maximum number of packets is exceeded. The application data packets are Constant Bit Rate (CBR) packets. Use of Transmission Control Protocol (TCP) or other higher level protocols is avoided in order not to skew the results with protocol timers used by such protocols.

A total of 60 mobility scenarios are generated with each one of them used twice for DSR and DB-DSR. The experiments are run with the number of nodes in the network set to 25, 50, 75 and 100 respectively. The speed with which these nodes are moving in the area is altered between 10, 20, 30 m/sec. When nodes reach the edges of the simulated area they bounce back with a 90 degrees angle. It has to be noted here that these conditions are "stress testing" both the algorithms and *ns*. Especially the scenario with the 100 nodes travelling at 30 m/sec (108 Km/h) is not the average speed that you travel inside a city. Regarding *ns*, there has only been one previous simulation experiment that tested two routing protocols with 100 nodes, though they increased the perimeter of the area to $2200m \times 600m$ and the nodes were travelling at speeds uniformly distributed between 0–20 m/sec [DPR00]. Where possible the simulation parameters have remained unchanged, with the same values as the one used during performance evaluation of the DSR from the Monarch group.

5.3 Performance Metrics

Three parameters are measured,

Number of Route Requests. How many ROUTE REQUEST messages have originated from source nodes.

Packet Delivery Ratio. The number of data packets delivered to destinations divided by the number of data packets generated by the CBR sources.

Channel Usage. The number of control bytes transmitted per data bytes delivered. Control bytes account for control messages (ROUTE REQUEST, ROUTE REPLY, ROUTE ERROR) as well as the routing overhead in Packet Data Units. Data bytes account for the number of bytes in a packet excluding the header.

For the purposes of evaluating route lifetimes the first metric is the most important one, since it shows the number of times that routes were broken and attempted re-established by the source node. This metric provides an indication whether the original aims of this algorithm are satisfied. The second and third metrics measure the gain in routing efficiency. Packet delivery ratio is important because it describes the throughput that the network can support and is therefore an indication of the loss rate that will be seen by higher level protocols. Channel usage presents a metric of how optimal the use of the physical channel is from the routing protocol. Long source routes for example will create long packet headers that will be appended to the data packets generated and produce a low ratio of routing efficiency.

5.3.1 Number of Route Requests

Figures 5.1, 5.2 and 5.3 show the results of the first set of experiments comparing the number of route requests to the number of nodes and the speed they roam. A total set of 15 experiments were run, 5 times for each speed selected. The results presented in each figure is an average of the numbers extracted from each of the five runs. As mentioned above, we expected DB-DSR to have had fewer route request messages therefore indicating that longer route lifetimes were established. The speed was varied to observe how that would affect the establishment of routes.

The results generated show that DSR and DB-DSR create about the same number of request messages with some small variations. The biggest difference that favours DSR is in the case of 100 nodes moving with 20 m/sec(figure 5.2) which is in the area of 10.2%. In contrast DB-DSR shows fewer request messages in the case of 25 nodes moving at 30 m/sec (figure 5.3). This difference is in the range of 3.1%. What is surprising in both protocols, is the number of route request messages that are generated when 100 mobile hosts roam the network. Although both of the protocols seem to scale gracefully as density increases from 25 to 50 and then to 75 nodes, in the 100 nodes case there is an apparent deterioration of connectivity. Until a certain threshold, the effect of network density is that more ROUTE REPLY messages will arrive from the destination to the source with more alternative paths added to the route cache. In this way, connections that would otherwise terminate due to ROUTE ERROR messages, can be redirected through different paths increasing route lifetimes and the efficiency of the protocol. In the 100 nodes case, perhaps the density of the network and the broadcast nature of ROUTE REQUESTS led to network congestion (a phenomenon called "broadcast storm effect" [SYNYCTYSCJPS99] in MANETs) that caused nodes to drop incoming packets. Because of the similarity of the simulation data, a statistical analysis was performed on the results that constitute figure 5.1. In practice, a *difference of 2 means test* was conducted on the data of table 5.2. This is expressed in the

	DSR(25n)	DB-DSR	DSR(50n)	DB-DSR	DSR(75n)	DB-DSR	DSR(100n)	DB-DSR
Run 1	1094	1126	593	653	292	285	1208	1320
Run 2	725	742	990	1046	349	307	1797	1829
Run 3	921	876	604	575	390	416	1103	1099
Run 4	699	698	661	699	548	583	1021	1060
Run 5	660	737	910	1024	522	629	2316	2190
t	0.315046342		0.831841177		0.619714342		0.071515342	

Table 5.2: Total no. of route requests (10m/sec)

following equation,

$$t = \frac{\overline{x}_1 - \overline{x}_2}{\sqrt{\frac{\sigma_1^2 - \sigma_2^2}{n}}} \tag{5.1}$$

where $\overline{x}_1$, $\overline{x}_2$ constitutes the mean and σ_1, σ_2 the standard deviation. When t is less that 3, it is interpreted as the absence of a statistical significant difference between the two process means. The last row of the table represents value t of the above equation, which being a very small value leads to the conclusion that the differences between the two protocols is likely to be coming from random factors. One way to overcome this result is to provide a larger sample of data, but tight time constrains as well as the time and resource intensive[1] running of *ns* have prevented from gathering more data.

5.3.2 Packet Delivery Ratio

The packet delivery ratio as a function of the number of nodes and the mobility speed is shown in figures 5.4, 5.5 and 5.6. As with the number of route requests, the pattern in the values is the same as for the packet delivery ratio. Besides, the two results are intermingled, since more route requests, means more data packets had been dropped at some intermediate node resulting in the overall packet delivery

[1]Scenario generation and simulation running took four high-end machines two days to finish.

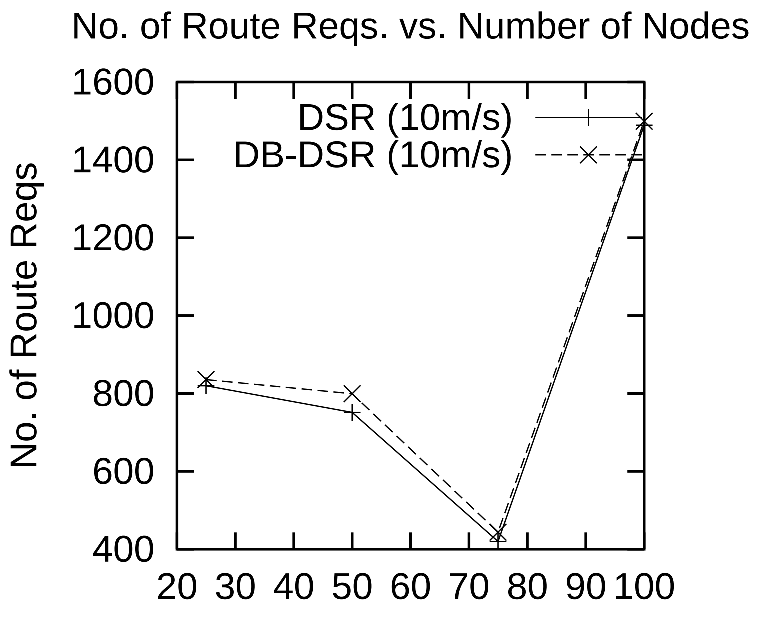

Figure 5.1: No. of Nodes (10m/sec)

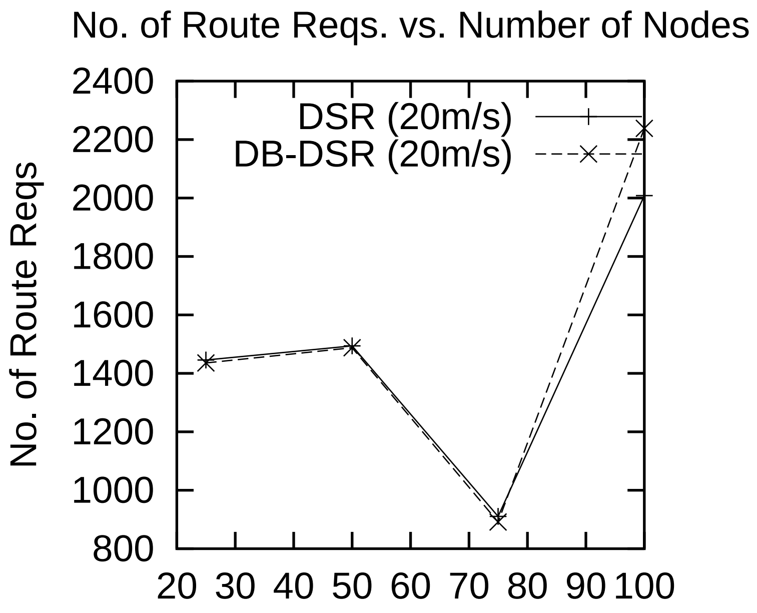

Figure 5.2: No. of Nodes (20m/sec)

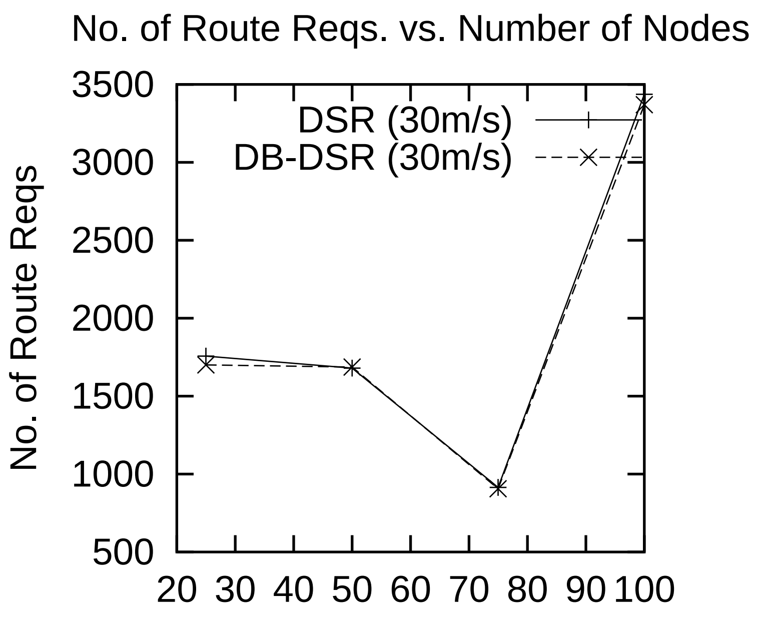

Figure 5.3: No. of Nodes (30m/sec)

ratio dropping. As speed increases, the routing effectiveness of DSR and DB-DSR degrades. When the node population increases until the 75 nodes threshold, the ratio remains approximately the same, gaining a bit when node mobility is 10 and 30 m/sec.

As nodes move faster, link connectivity changes more often, resulting in more control messages sent between nodes. When the network population is 100 nodes, this situation creates an enormous amount of packets, contributing to collisions, congestion, contention and packet drops. Again, both protocols seem to perform well and have a high delivery rate up to a certain density beyond which scalability is poor.

5.3.3 Channel Usage

This section compares the bandwidth requirements incurred by the two routing protocols. We observe the total routing cost experienced per delivered data packet. This includes all control messages (route request, route reply and route error) as well

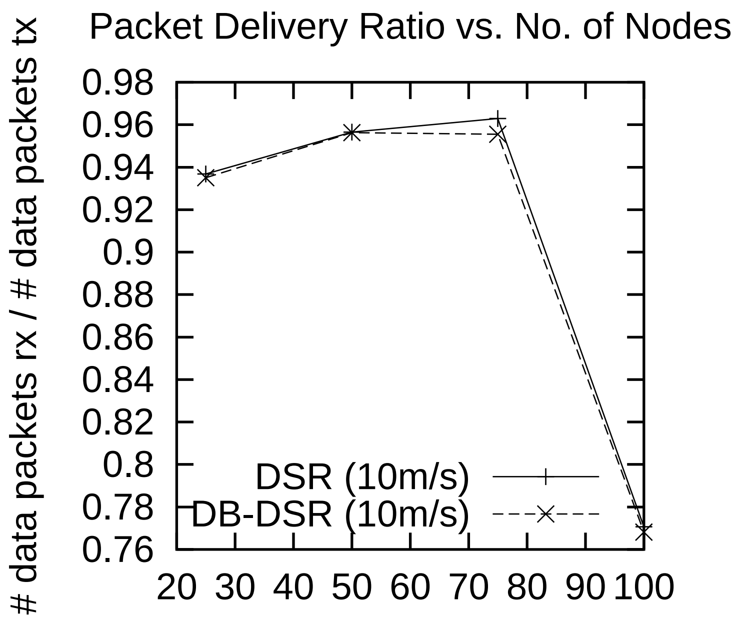

Figure 5.4: No. of Nodes (10m/sec)

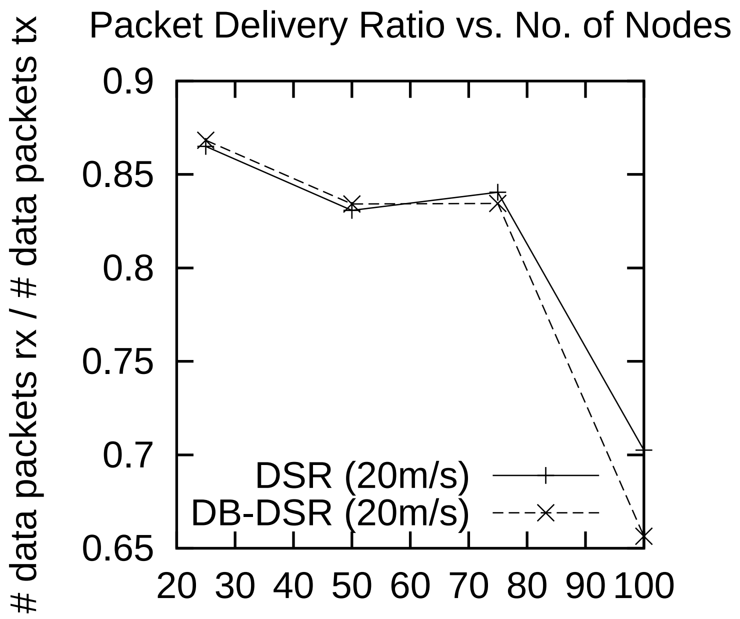

Figure 5.5: No. of Nodes (20m/sec)

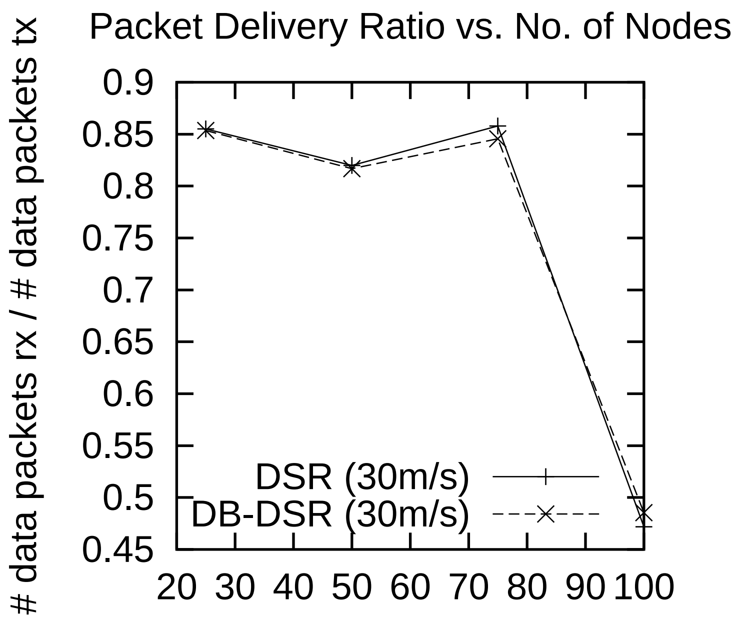

Figure 5.6: No. of Nodes (30m/sec)

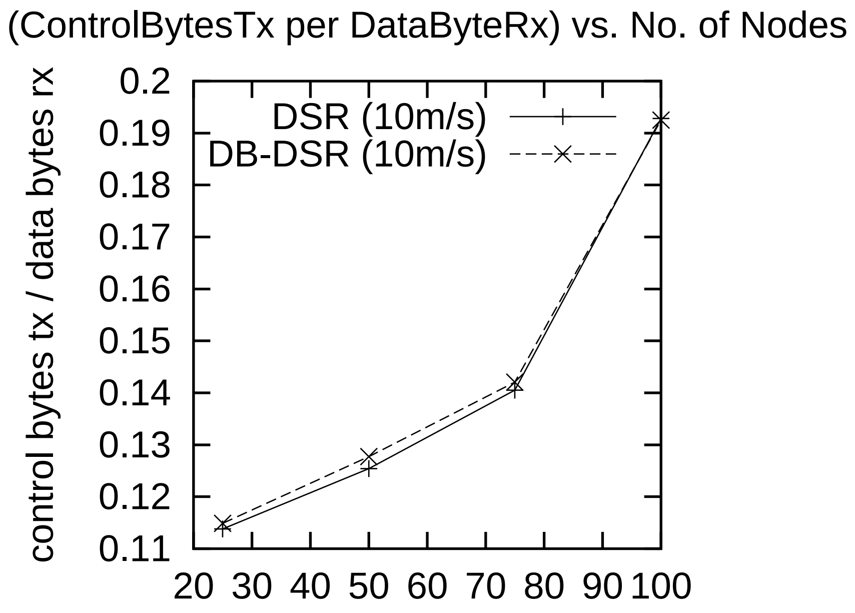

Figure 5.7: No. of Nodes (10m/sec)

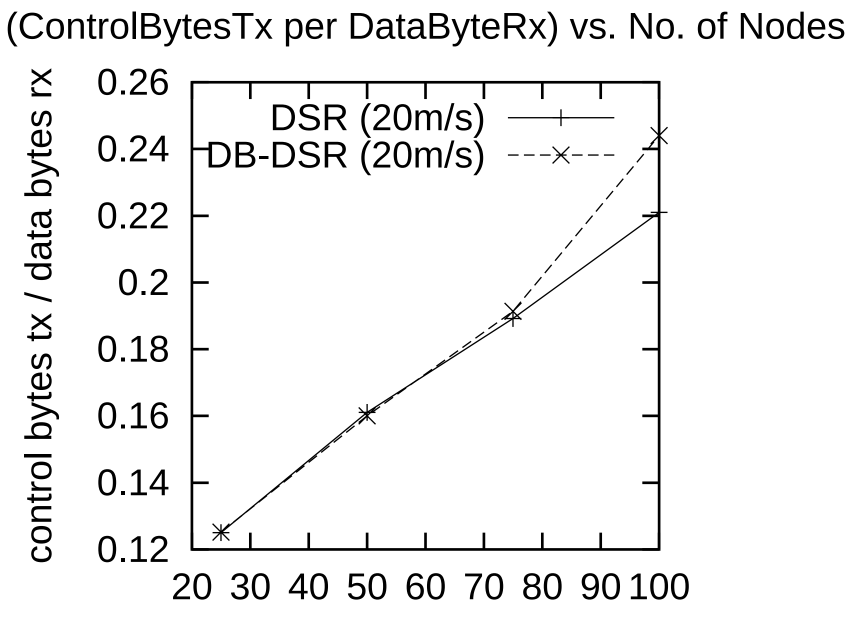

Figure 5.8: No. of Nodes (20m/sec)

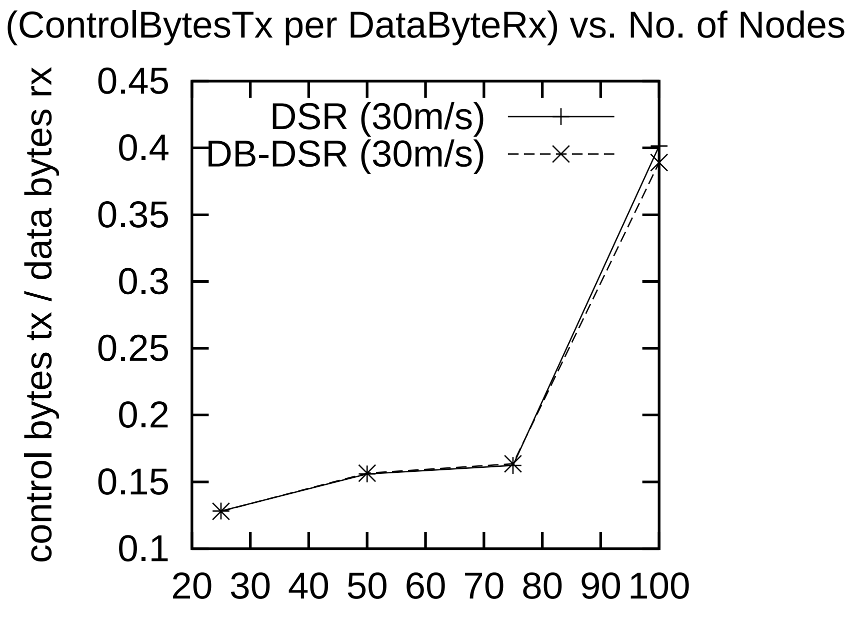

Figure 5.9: No. of Nodes (30m/sec)

as the routing header of each data packet. Only data packets that have actually been received by the destination (not lost packets) are measured here.

Figures 5.7, 5.8 and 5.9 show the results. Again the two protocols seem to perform very similarly, with DB-DSR performing slightly worse in the 100 nodes case moving at 20 m/sec (figure 5.8). As expected from the previous experiments, scalability is poor as node density increases. It is interesting to note that the overhead caused by control messages increases by a small fraction up until a density of 75 nodes. After that when the population increases to 100 nodes, efficiency deteriorates rapidly coming as close as to 2.5 data bytes received for every control byte transmitted. This is an especially important scalability issue that has been identified here regarding DSR, which shows that in densely populated, low-bandwidth MANETs a huge amount of bandwidth will be consumed from control messages trying to give nodes an up to date view of the network.

5.4 Summary

This chapter has presented the results extracted from *ns* for evaluating Direction Based DSR against the standard DSR implementation. Performance testing was conducted measuring the longevity of routes, routing protocol efficiency and channel efficiency. These values have been extracted after tests ran for 900 seconds of simulated time with varying the number of nodes and speed they moved. A difference of 2 means test was performed on a fraction of these results to discover any statistical significance among them and found them not to be statistical significant. This means that more sample data is needed since the differences shown might be produced by factors not accounted for.

Results have shown that both DSR and DB-DSR perform on about the same levels, though surprisingly DB-DSR performs worse in some marginal cases. DSR as a protocol faces scalability problems in dense networks, with rapid deterioration of throughput as route discovery broadcasts flood the network creating packet collisions and contention. The density threshold for protocol efficiency seems to be around the 75 nodes roaming in an area of $1500m \times 300m$.

Node velocity affected network performance by creating a rapidly changing network topology. Although the set of speeds chosen might have been high (30 m/sec) it shows that DSR has problems adapting to fast changing topologies. Still the result indicate that network density has a far worse effect in performance than high node mobility.

Chapter 6

Conclusion

This thesis has described the analysis, design and implementation of a new routing algorithm for mobile ad hoc networks that is based on the similarity of direction of the participating nodes. The algorithm was implemented in the *ns* network simulator as an extension to the Dynamic Source Routing protocol. A comparative performance evaluation was also conducted, between the standard Dynamic Source Routing and the modified version described here.

An introduction to mobile ad hoc networks is presented in chapter 1 which leads into design considerations for routing algorithms in MANETs as described in chapter 2. Most of the ideas behind the subject of this thesis are discussed and analysed in chapter 3, which introduces the concept of Direction Based Routing.

6.1 Objectives Fulfilled

In the introduction to this thesis, section 1.3, mentions four objectives. These aims scope and explain the project. However, they are too abstract as they give no indication on the rationale behind this idea and the expected areas of improvement. Section 3.1.1 mentions longevity of routes and reduced packet overhead as the objectives for the evaluation of the Direction Based Routing algorithm.

All of the aims were fulfilled having a simulated implementation and a performance evaluation that is mentioned in chapter 5. There were 60 scenarios that were simulated for each one of the two routing protocol. The scenarios explored routing behaviour by changing the density of the nodes and the speed by which nodes move in an area. The random waypoint model was used as a mobility scenario, while nodes established connections and sent data using the Constant Bit Rate (CBR) protocol.

The results of the simulation were used as input to three performance metrics, namely, route longevity, routing efficiency and channel efficiency. The extracted values are shown and explained in section 5.3. In short, the two protocols are found to perform very similarly in all of the cases. More significant was the result about the scalability of DSR in dense networks. DSR and of course Direction Based Routing which is based on DSR, perform well up until a density of 75 nodes in an area of $1500m \times 300m$. When node density rises above the 75 nodes threshold, there is a severe degradation of performance effecting routing and channel efficiency leading to poor network throughput.

6.2 Future Work

This section describes future work that can be done to provide more performance results and enhance the algorithm devised in this thesis. Work can be directed to the following areas,

- Test and evaluate the algorithm with different mobility models than the random waypoint. Ideally, real life movement patterns should be used so that the directional aspect of the algorithm could be utilised to the maximum extend.
- More sample data is needed for a better evaluation. Tests for each case should probably be run between 20 and 30 times.
- When node density increases, the simulated area should increase correspondingly giving a constant density value. This will exclude the performance de-

terioration in DSR due to population density when measuring other protocol characteristics.

- A new cache eviction scheme should be realised for DSR. Entries that contain paths with higher directional metrics should be evicted as late as possible. Right now the eviction scheme works in a FIFO order.

- Route longevity should be measured per path to destination (rather than per route, which can encapsulate many paths) so that the affect of different values of the directional metric can be measured against the lifetime of a path.

Bibliography

[BCSW99] S. Basagni, I. Chlamtac, V. Syrotiuk, and B. Woodward. A Distance Routing Effect Algorithm for Mobility (DREAM), 1999.

[BG87] Dimitri P. Bertsekas and Robert Gallager. *Data Networks*. Prentice Hall, 1987.

[BMJ$^+$98] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In *Mobile Computing and Networking*, pages 85–97, 1998.

[CM99] M. Corson and J. Macker. RFC 2501: Mobile Ad hoc NETworking (MANET): Routing Protocol Performance Issues and Evaluation Considerations, 1999.

[DPR00] Samir Ranjan Das, Charles E. Perkins, and Elizabeth E. Royer. Performance Comparison of Two On-demand Routing Protocols for Ad Hoc Networks. In *INFOCOM (1)*, pages 3–12, 2000.

[DTMB99] Diane Tang and Mary Baker. Analysis of a Metropolitan-Area Wireless Network. In *Mobile Computing and Networking*, pages 13–23, 1999.

[HJ00] Y. Hu and D. Johnson. Caching Strategies in On-Demand Routing Protocols for Wireless Ad Hoc Networks, 2000.

[HP97] Z. Haas and M. Pearlman. A New Routing Protocol for the Reconfigurable Wireless Networks, 1997.

[JM96] David B Johnson and David A Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[JMHJ00] D. Johnson, D. Maltz, Yih-Chun Hu, and J. Jetcheva. RFC DRAFT: The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks, 2000.

[KSU99] Evangelos Kranakis, Harvinder Singh, and Jorge Urrutia. Compass Routing on Geometric Networks. In *Proc. 11 th Canadian Conference on Computational Geometry*, pages 51–54, Vancouver, August 1999.

[LJD$^+$00] J. Li, J. Jannotti, D. De Couto, D. Karger, and R. Morris. A Scalable Location Service for Geographic Ad Hoc Routing, 2000.

[LS] X. Lin and I. Stojmenovic. GPS Based Distributed Routing Algorithms for Wireless Networks.

[Mal94] G. Malkin. RFC 1723: RIP version 2 — carrying additional information, November 1994. Status: DRAFT STANDARD.

[MBJ99] D. Maltz, J. Broch, and D. Johnson. Experiences Designing and Building a Multi-Hop Wireless Ad Hoc Network testbed, 1999.

[MF] S. McCanne and S. Floyd. ns—Network Simulator. `http://www-mash.cs.berkeley.edu/ns/`.

[Moy94] J. Moy. RFC 1583: OSPF version 2, March 1994. Status: DRAFT STANDARD.

[MP] The CMU Monarch Project. The CMU Monarch Project's Wireless and Mobility Extensions to *ns*. `ftp://ftp.monarch.cs.cmu.edu/pub/monarch/wireless-sim/ns-cmu.ps`.

[NI97] Julio C. Navas and Tomasz Imielinski. GeoCast - Geographic Addressing and Routing. In *Mobile Computing and Networking*, pages 66–76, 1997.

[PB94] Charles Perkins and Pravin Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications*, pages 234–244, 1994.

[PC98] V. Park and M. Corson. Temporally-Ordered Routing Algorithm, 1998.

[Per01] Charles E. Perkins. *Ad Hoc Networking*. Addison-Wessley, 2001.

[PMH] Miguel Sánchez Pietro, Pietro Manzoni, and Zygmunt J. Haas. Determination of Critical Transmission Range in Ad-Hoc Networks.

[PR99] C. Perkins and E. Royer. Ad-Hoc On-Demand Distance Vector, 1999.

[SLG00] W. Su, S. Lee, and M. Gerla. Mobility Prediction in Wireless Networks, 2000.

[SYNYCTYSCJPS99] Sze-Yao Ni, Yu-Chee Tseng, Yuh-Shyan Chen, and Jang-Ping Sheu. The Broadcast Storm Problem in a Mobile Ad Hoc Network. In *Mobile Computing and Networking*, pages 151–162, 1999.

[YBKV98] Young-Bae Ko and N. Vaidya. Location-Aided Routing (LAR) in Mobile Ad Hoc Networks. In *Proc. MOBICOM*, pages 66–75, 1998.

[YC] Wen-Hwa Liao Yu-Chee. GRID: A Fully Location-Aware Routing Protocol for Mobile Ad Hoc Networks.