NOTE TO USERS

This reproduction is the best copy available.

UMI[®]

A Load Balancing Algorithm Using Radio Signal Strength and Sensitivity for the Multipath Source Routing Protocol

By
Hadeel T. El-Kassabi

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfillment of
the requirements for the degree of
Master of Computer Science

Ottawa-Carleton Institute for Computer Science
School of Computer Science
Carleton University
Ottawa, Ontario

November 21, 2003

© Copyright
2003, Hadeel T. El-Kassabi

National Library
of Canada

Bibliothèque nationale
du Canada

Acquisitions and
Bibliographic Services

Acquisitons et
services bibliographiques

395 Wellington Street
Ottawa ON K1A 0N4
Canada

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

ISBN: 0-612-89835-0

Our file Notre référence

ISBN: 0-612-89835-0

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this dissertation.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de ce manuscrit.

While these forms may be included in the document page count, their removal does not represent any loss of content from the dissertation.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

Abstract

On-demand reactive routing protocols, such as DSR, are commonly used for MANETs. Multipath Source Routing (MSR) provides DSR with multiple paths to a single destination which offers more network resource utilization. Applying a weighted load balancing algorithm in MSR improves the performance of routing in MANET by reducing the end-to-end delay since it depends on the round trip time (RTT) delay as a factor to distribute traffic on available routes. However, this solution does not account for signal strength which is a major cause of packet errors in wireless networks. This thesis proposes a new route weighing mechanism that makes the MSR more adaptive to network errors resulting from loss of signal strength. The new strategy is to collect network information regarding the signal strength and the nodes' sensitivity to form a parameter called Received Power and Sensitivity (RPS), and use it as a weighing parameter with two flavors. The first evaluates the minimum RPS value among the legs between hops in each route. The second collects RPS cumulatively from all the legs in each route. Simulation results show that the new mechanism reduces the packet drop rate, and enhances the throughput with minimal additional overhead.

Acknowledgements

I first of all would like to thank Allah, the Creator, for giving me the ability to complete this thesis, and for guiding me while doing it.

I am forever indebted to my parents and my husband Waleed Khalil for their understanding, endless patience and encouragement when it was most required.

I would like to gratefully acknowledge the enthusiastic supervision of Prof. Kranakis and Prof. Barbeau during this work. I thank both of them for the technical discussions and inspiration, which was always there when I needed it.

Finally, I would like to express my gratitude to the members of my SmallSat group for their help and support. I would like also to thank all my friends especially Maged Elaasar, Dr. Walid Ibrahim and Hoda Amer for the enlightening technical discussions. This work would not have been possible without their support and encouragement.

Table of Contents

Abstract	iii
Acknowledgements	iv
List of Tables	viii
List of Figures	ix
List of Acronyms	xi
1 Introduction	1
1.1 Examples of MANET	2
1.2 Types of Protocol Algorithms for MANET	3
1.3 Problem Definition	5
1.4 Thesis Contribution	6
1.5 Thesis Outline	8
2 Background and Related Work	9
2.1 Mobile Ad Hoc Networks (MANET)	10
2.1.1 Radio System	11
2.2 Routing	13
2.3 Routing Issues in Some Mobile Networks Types	14
2.3.1 Virtual Topology Pattern	15
2.3.2 Virtual Node Concept	15
2.3.3 Virtual Connection-Oriented Approach	16
2.3.4 Datagram Based Routing Algorithms	17
2.3.5 Self-Organizing Protocol for Small Ad Hoc Networks	18
2.3.6 Problems	19
2.4 IP Routing and Mobility for MANET	21
2.5 Routing within MANET	22
2.5.1 Routing Protocol Characteristics and Metrics	24

2.5.2	Types of MANET Routing Protocol Algorithms	25
2.5.3	Proactive Routing Protocols	27
2.5.3.1	Destination-Sequenced Distance-Vector (DSDV) . . .	27
2.5.3.2	Wireless Routing Protocol (WRP)	28
2.5.3.3	Problems of Proactive algorithms	28
2.5.4	Reactive Routing Protocols	29
2.5.4.1	Temporally Ordered Routing Algorithm (TORA) . .	29
2.5.4.2	Associativity-Based Routing (ABR)	31
2.5.4.3	Signal Stability Routing (SSR)	33
2.5.4.4	Location Aided Routing (LAR)	34
2.5.4.5	Dynamic Source Routing (DSR)	36
2.5.4.6	Ad Hoc On-Demand Distance Vector Routing (AODV) .	38
2.5.5	Hybrid Routing Protocols	40
2.5.5.1	Zone Routing Protocol (ZRP)	40
2.5.6	General Comment on All Algorithms	41
2.6	Detailed Description of DSR Algorithm and Packet Formats	42
2.6.1	Route Construction	42
2.6.2	Route Maintenance	44
2.6.3	Packet Description	44
2.7	IPv4 Header	45
2.8	Strict Source Routing Header	46
2.8.1	DSR Header	47
2.8.1.1	Route Request Option	48
2.8.1.2	Route Reply Option	49
2.8.1.3	Route Error Option	50
2.8.2	Optimizations for DSR	52
2.8.2.1	Routing Cache Utilization	52
2.8.2.2	Piggyback the Data on Route Request Packets . . .	53
2.8.2.3	Update Routing Cache	54
2.8.2.4	Handling Errors	55
2.8.2.5	Multipath Support	55
2.9	Adaptive MSR Using RTT	57
2.10	Summary	60
3	Enhanced MSR	63
3.1	DSR and MSR in PIX	64
3.1.1	Data Structures	65
3.1.1.1	Route Cache	65
3.1.1.2	Self-Initiated Route Request Table	66
3.1.2	DSR Protocol Class	67

3.1.3	Algorithms and Data Flow Diagrams	68
3.1.3.1	Route Construction	68
3.1.3.2	Route Maintenance	76
3.1.3.3	Data Forwarding	78
3.2	Adaptive MSR Using RTT (WLB-RTT)	80
3.3	WLB-RTTJ	81
3.4	Adaptive MSR Using Minimum RPS (WLB-RPSmin)	84
3.4.1	Route Construction	86
3.4.2	Route Maintenance	89
3.4.3	Data Forwarding	89
3.5	Adaptive MSR Using Cumulative RPS (WLB-RPScum)	90
4	Simulation and Results Analysis	93
4.1	Simulation Environment	93
4.2	Simulation Configuration	96
4.2.1	Mobility Model	98
4.2.2	Simulation Parameters	98
4.2.3	Performance Metrics	100
4.3	Test Cases	101
4.3.1	The First Experiment	101
4.3.2	The Second Experiment	101
4.3.3	The Third Experiment	102
4.3.4	The Fourth Experiment	102
4.4	Results Analysis	102
4.4.1	Experiment 1 Expanding Network	105
4.4.1.1	Total Delay and Delay per Packet	105
4.4.1.2	Total Number of Packets Dropped and Delivery Ratio	107
4.4.1.3	Throughput	108
4.4.2	Experiment 2	109
4.4.2.1	Total Delay and Delay per Packet	112
4.4.2.2	Throughput	113
4.4.3	Experiment 3	114
4.4.4	Experiment 4	118
4.5	Summary	120
5	Conclusion and Future Work	121
	Bibliography	126
A	Measurement Data	132

List of Tables

A.1	Experiment 1 Results	133
A.2	Experiment 2 Results	134
A.3	Experiment 3 Results	135
A.4	Experiment 4 Results	136

List of Figures

2.1	IPv4 Packet Header Format	45
2.2	Strict Source Routing Header Format	46
2.3	Dynamic Source Routing Header Format	46
2.4	Route Request Option Format	47
2.5	Route Reply Option Format	49
2.6	Error Option Format	50
2.7	Error Option Type-Specific Information Format	52
3.1	PIX Packet Format	65
3.2	Route Cache	67
3.3	Generate a Route Request	69
3.4	Handle Incoming Route Request	71
3.5	Network Configuration Example	73
3.6	Route Reply Packet Format	73
3.7	Handle Incoming Route Reply	75
3.8	Handle Incoming Route Error	77
3.9	Handle Incoming Data Packet	79
3.10	New Route Reply Packet Format	83
3.11	Beacon Packet Format	84
3.12	Reply Packet Format with the RPSmin extension	87
3.13	Request Packet Format with the RPSmin extension	87
3.14	Network Configuration Example	90
4.1	A simulated network	97
4.2	Experiment 1: Total Delay	104
4.3	Experiment 1: Delay per Packet	104
4.4	Experiment 1: Total Packets Dropped	106
4.5	Experiment 1: Delivery Ratio	106
4.6	Experiment 1: Throughput	108
4.7	Experiment 2: Total Delay	110
4.8	Experiment 2: Total number of Packets Dropped	110

4.9	Experiment 2: Delivery Ratio	111
4.10	Experiment 2: Delay per Packet	111
4.11	Experiment 2: Throughput	113
4.12	Experiment 3: Total Delay	115
4.13	Experiment 3: Total number of Packets Dropped	115
4.14	Experiment 3: Delay per Packet	116
4.15	Experiment 3: Throughput	116
4.16	Experiment 3: Delivery Ratio	117
4.17	Experiment 4: Throughput	118
4.18	Experiment 4: Total number of Packets Dropped	119

List of Acronyms

ABR	Associativity-Based Routing
AODV	Ad Hoc On-Demand Distance Vector
APP	Application Protocol
ATM	Asynchronous Transfer Mode
BER	Bit Error Rate
CBR	Constant Bit Rate
dB	decibels
DRP	Dynamic Routing Protocol
DSDV	Destination-Sequenced Distance-Vector
DSR	Dynamic Source Routing
EIRP	Effective Isotropic Radiated Power
ESA	European Space Agency
FTP	File Transport Protocol
GEO	Geostationary Earth Orbiting Satellite
GP	Generative Programming
GPS	Global Positioning System
ICMP	Internet Control Message Protocol
IETF	Internet Engineering Task Force
IGMP	Internet Group Management Protocol
IMEP	Internet MANET Encapsulation Protocol
IP	Internet Protocol
ISL	Inter-Satellite Links
LAN	Local Area Network
LAR	Location Aided Routing
LEO	Low Earth Orbiting Satellite

MAC	Medium Access Control
MANET	Mobile Ad hoc Network
MRL	Message Retransmission List
MSR	Multipath Source Routing
NIST	National Institute of Standards and Technology
PIX	Protocol Implementation framework for Linux
POSIX	Portable Operating System Interface for X-based Systems
QLP	Queue Link Protocol
QoS	Quality of Service
RF	Radio-Frequency
RFC	Request For Comment
RPS	Received Power and Sensitivity
RREP	Route Reply
RREQ	Route Request
RTT	Round Trip Time
SRP	Static Routing Protocol
SSR	Signal Stability Routing
SST	signal stability table
SYN	Synchronization Flag
TCP	Transmission Control Protocol
TORA	Temporally Ordered Routing Algorithm
UDP	User Datagram Protocol
WLB	Weighted Load Balancing
WRP	Wireless Routing Protocol
ZRP	Zone Routing Protocol

Chapter 1

Introduction

Wireless networks have attained significant popularity over the last few decades. This is because they are used to support mobility in communication. Mobile nodes, such as *Low-Earth-Orbit (LEO)* satellites, were introduced to support more global coverage and less latency of data transmission, since they are launched in lower altitude orbits than *Geosationary (GEO)* satellites. The challenge is to get a mobile network up and running and to develop a suitable and efficient routing algorithm. This routing algorithm should adapt to the mobile nature of the network.

There are two types of wireless networks. *Infrastructured networks* include gateways that are fixed and connected with wires and base stations that work as bridges. The mobile nodes connect to the closest base station. When the mobile node gets out of the base station range, and enters the range of a new base station, a “hand-off” occurs. Another type of wireless networks is called an *Infrastructureless* mobile network or, alternatively, an *Ad Hoc Network*. Routers in ad hoc networks are mobile.

All mobile nodes can perform routing functionality and they can discover routes and maintain valid routes to other nodes in the network. In addition, mobile nodes can connect to the network dynamically [RT99].

This research focuses on routing protocol issues in Mobile Ad Hoc Networks. Developing a routing protocol for such mobile ad hoc networks is a challenging process that should take into consideration network behavior, resource limitations and other unique characteristics of such networks.

1.1 Examples of MANET

There are many mobile ad hoc network (MANET) network types. Satellite data networks are a typical example of MANET. Currently, there exist different satellite configurations such as distributed spacecraft constellations which use crosslink communications to share information. An example of such a network is Iridium. Another type of satellite configuration is formation flying missions which require crosslink communications to maintain position, such as Techsat 21 and MAXIM [Old03]. “Satellite networks have enabled a whole new class of applications by connecting telecommunication networks, empowering cellular networks, transmitting radio and video signals and collecting intelligence information” [Ela03]. The LEO satellites are less expensive, encouraging their use for internet connectivity, especially in areas which cannot afford the costs of terrestrial infrastructure setup. LEO satellites also exhibit shorter

transfer delay times and lower path loss [SSC96]. LEO satellites use orbits with lower altitudes than *Geosationary (GEO)* satellites. This feature allows a wide range of geographical coverage and provides higher capacity because of the ability to reuse communication frequencies, which are limited in earth space [WCA⁺01]. In addition, LEO-based satellite networks attract research consideration because of their ability to transfer data with lower delays and better bit-error rates of performance than do GEO satellites. This is why it is considered suitable for Internet and multimedia services [GPAA99]. It is considered as a “special type of mobile network” [HK00]. However, LEO satellites have never been used in practice, but have been proposed for the first time in by Binder et al. as a cost effective and robust means of global voice and data communications [BHGV87].

1.2 Types of Protocol Algorithms for MANET

There are some existing metrics for evaluating MANET routing protocols. Corson and Macker have defined several metrics to evaluate a routing protocol. The following characteristics or features reflect some of these metrics: distributed, loop-free, reactive or proactive operations; security; enablement of processor sleep operation to preserve processing power and save energy; support for all types of links, bi-directional and unidirectional; and end-to-end data throughput. Some other metrics include measuring efficiency, overhead, and transfer rate [CM99]. These metrics will be discussed in more detail in the second chapter of this thesis.

There are three different approaches used in developing routing algorithms in MANET: proactive, reactive and hybrid. A proactive algorithm is an adaptive system of routing. Routes are updated periodically by exchanging control packets between nodes. Therefore, route information for all nodes is updated constantly in the routing tables of each node. Whenever a request for a route is issued, the table will instantly provide the route. The draw back of this algorithm is that it wastes considerable bandwidth for the large number of control packets exchanged that may not be needed. The other type is the reactive protocol algorithm. Initially, no routes are constructed except when a route to a destination is requested, in which case an attempt to find a route is performed by flooding the control packets. This is called on-demand routing. This is opposite to proactive protocols, and hence bandwidth is only used to send required information. When the route is found, the whole bandwidth is used transmitting data packets. However, a large portion of the bandwidth is used in the initial flooding in order to search for the route; as a result, sent data packets may experience greater delays than under the proactive algorithm. Some attempts have been made to combine the advantages of both proactive and reactive protocol algorithms in a protocol category called hybrid protocol algorithms. In this protocol, a mixture of proactive and reactive schemes, or a portion of either one, is used.

Proactive protocols are also called a table-driven ad hoc routing approach. It has

many disadvantages when used in MANETs. It does allow for the connectionless approach of forwarding packets, but it does not take into consideration the need for the route being constructed. However, it has the advantage of having all existing routes in the table, which lowers delays for datagram transmission. It also results in significant overhead because of periodic updates of routing table information and the maintenance of table consistency. In addition, it wastes a huge amount of traffic and consumes a lot of power. This is a serious limitation, because bandwidth and battery power are limited resources in mobile computers and satellites [RT99]. Due to the reasons listed above, reactive or on-demand routing algorithms are better suited to mobile ad hoc networks.

1.3 Problem Definition

One of the commonly used reactive routing algorithms is Dynamic Source Routing, since it has low routing overhead [ZZS⁺02]. However, basic single path Dynamic Source Routing (DSR) does not efficiently utilize the network resources nor cope with link breakage and congestion. Wang et al. recommend an enhancement over DSR using a Multi-path Dynamic Source Routing algorithm which uses a multiple paths extension of the DSR. Wang et al. also propose traffic load balanced distribution over different paths to overcome congestion resulting from using single paths, and to increase network resource utilization. The traffic distribution is based on a heuristic equation which depends on the weight of each route, evaluated according to its RTT,

round trip time, delay [WSD⁺01].

On the other hand, some applications give more importance to stable connections to achieve a required level of QoS rather than fast connections. Stable links are important to reduce packet loss and avoid route maintenance overhead which is costly in mobile networks since it usually results in broadcast of control packets through the network [GWFM02]. The signal strength of the links in a route determines the stability of that route [AASS00]. The variation pattern of the received signal strength suggest a movement pattern of the mobile nodes and hence gives the chance to estimate the probability of connection loss [GWFM02]. The existing MSR protocol does not evaluate the route weight according to its signal strength, but only according to its round trip time delay. This way will allow transferring data through weak routes and allow for higher probability of packet loss.

1.4 Thesis Contribution

This thesis proposes a new weighing mechanism that makes the MSR protocol more adaptive to network errors resulting from the loss of signal strength. The new strategy is based on collecting network information regarding the received signal strength and node sensitivity, compiling this information to form a parameter called Received Power and Sensitivity (RPS), and using this value as a weighing parameter for each route (instead of round trip time delay parameter). There are two flavors of this algorithm. The first flavor evaluates the route weight using the minimum RPS value

among the legs between hops in each route. The second flavor collects RPS information cumulatively among all the legs in each route.

The simulation used in this thesis proves that the new weighing mechanism for MSR results in better throughput and reduces packet drop rates. The different experiments carried out in this thesis study the behavior of the basic DSR algorithm, MSR with RTT as the weighing parameter and MSR using RPS as a weighing parameter, with each of its two flavors.

Each experiment represents a different ad hoc network behavior. The first uses an expanding network, where all nodes move away from each other to represent an extreme network condition. The second experiment represents a network with a combination of expanding and converging nodes. The third experiment introduces random packet dropping. Finally, the fourth experiment compares the behavior of each algorithm under different packet dropping rates.

The performance metrics used to evaluate each algorithm under study are delivery ratio, throughput, end-to-end delay, delay per packet and total number of packets dropped.

The simulation is implemented in PIX (Protocol Implementation Framework for

Linux) [Bar02, BB02, Zha02], developed in the School of Computer Science at Carleton University. The IP layer is the component that contains the algorithm implementation in PIX. PIX provides the implementation with a preestablished object-oriented design pattern, facility of configuration and deployment and access to lightweight libraries [Ela03].

There are also two more Protocol layers used for the PIX-based simulation framework. One is Application layer, which is a data streaming protocol. This layer is responsible for generating data traffic patterns used in the experiments. The other layer is called QLP, which is a link protocol based on POSIX message queues to simulate virtual networks on a single physical machine [Ela03, ELBK03].

1.5 Thesis Outline

This thesis is organized into four additional chapters. Chapter 2 reviews the literature of the existing routing protocols for MANET and some proposed enhancements for better performance. Chapter 3 explains the design of the new MSR weighing mechanisms in detail. Chapter 4 describes the simulation framework, experiments and test cases, and simulation results and analysis. Chapter 5 concludes the thesis and gives a summary, suggesting ideas for future work.

Chapter 2

Background and Related Work

Traditional routing protocols are not well suited to ad hoc networks since they are not designed to adapt to the continuous network topology changes resulting from the random movement of nodes. Mobile nodes, such as LEO satellites, impose more restrictions on routing protocols due to their limitations in bandwidth, memory and processing power. Adaptive on-demand routing protocols are commonly used in ad hoc networks since they are light weighted and adapt to network topology changes.

This chapter defines the characteristics of mobile ad hoc networks. The chapter also concentrates and examines routing issues in this type of a network in general. Some ideas regarding IP routing protocol are stated. In addition, different routing algorithms are evaluated to choose one suitable to be applied in our mobile network. Dynamic Source Routing (DSR) is found to be an efficient routing protocol suitable for mobile ad hoc networks. Moreover, the chapter reveals some details concerning DSR algorithm, design, and optimizations suggested for DSR, which will increase

its efficiency and suitability to mobile and wireless networks. This chapter will also describe in detail traffic load distribution among multiple paths in combination with Multiple Source Routing (MSR).

2.1 Mobile Ad Hoc Networks (MANET)

MANET refers to a collection of mobile nodes connected together through a wireless communication media. Nodes can be hosts or routers with wireless communication devices moving freely in a random manner. The nodes can exist in moving airplanes, ships, or satellites [CM99]. These nodes construct a dynamic network where there is no centralized control and no fixed structure. Moving nodes function as routers so they construct and maintain routes and forward packets destined for other nodes. Each node is considered a hop in the route path, which consists of many hops. This network can work as standalone network or it may have interfaces to a fixed network [CM99]. Home agents and foreign agents do not exist in an ad hoc network [Joh94]. There are many qualifications that distinguish the MANET. One of these is the concept of *dynamic topologies*, in which the network topology changes randomly and quickly in response to the random movement of nodes. It also consists of two types of links, bi-directional and unidirectional. Another characteristic is the constrained bandwidth availability. This is because the links are wireless; such links are known to have less capacity than hardwired links. Other facts that interact with the limited bandwidth are noise, interference and multiple accesses. This characteristic make

congestion occur in all links. Moreover, energy is limited in most of the nodes in MANET because most mobile nodes are battery operated. Thus, good operation design should conserve energy. In addition, security is not easy to achieve. This is because wireless links are more vulnerable to security problems than hardwired links. Some security techniques exist and are applied in wireless networks to resolve that issue [CM99]. In other words, the key characteristics of this network are mobility, and wireless communication [NNB00], while the network is constrained by the scarcity of bandwidth and energy, and the difficulty in achieving security. An example of MANET is the LEO-based satellite network. Small Satellites have lower costs and less data transmission delay than larger satellites. These features have attracted the attention of the world to their promising applications. One of these applications is communication, such as connecting to the Internet while travelling in a car or airplane. Other applications are scientific research such as checking ozone or radiation levels in space, or detecting earth disasters such as volcanoes or earth quakes [Hak01].

In order to develop an efficient routing protocol for MANET, it is important to study the physical characteristics of mobile nodes such as the wireless specifications. In the next section, some of radio information is discussed.

2.1.1 Radio System

A typical radio system consists of a transmitter, antenna, and transmission medium. Information is transmitted to the transmitter through an antenna which converts

the RF signal into an electromagnetic wave. The electromagnetic wave propagates through free space, which is the transmission medium in this case. The receiver antenna captures the electromagnetic wave and converts it back to an RF signal. Following are some definitions needed later in this research. Attenuation is the RF signal fading at the attenuator. Attenuation is expressed in dB decibels, and is calculated by

$$PdB = 10 \times \log \frac{P_{out}}{P_{in}}$$

where P_{in} is the incident power level at the attenuator input and P_{out} is the output power level at the attenuator output. Another definition is the Path Loss, which measures the reduction of the power of an RF signal due to propagation through space. It is also measured in dB. In fact, Path loss is affected by the distance between the transmitting and receiving antennas, the clearance of line of site between the transmitting and receiving antennas and the antenna height. Some antenna characteristics are defined as follows. An Isotropic antenna is an ideal antenna where the radiation intensity in all directions is equal; it is the 0 db gain reference. The Gain is “defined as the ratio of the radiation intensity in a given direction to the radiation intensity that would be obtained if the power accepted by the antenna was radiated equally in all directions (isotropically).” Receiver Sensitivity is one of the main system characteristics; it is defined as the lowest RF signal power that can be received at the input of a receiver antenna. EIRP (Effective Isotropic Radiated Power) is the

antenna transmitted power which is the net power of transmission, taking the cable loss out and adding the antenna gain in [Inc].

2.2 Routing

A network consists of a number of nodes communicating with each other. Each node has a network-wide unique address. Not all nodes are connected directly to each other. Nodes which are directly connected, can simply forward packets to each other. The need for “Routing” arises when a packet needs to be delivered from one node to a different node which it is not directly connected to. A node should find a best route path to the destination. A node can have many different paths to the destination; a routing algorithm is needed to define the rules to select which route to follow. Decision criteria of routing algorithms may be the least number of hops, least delay time, minimum CPU processing at nodes along the path, or least cost, which may be evaluated in many ways. Performance evaluation of routing algorithms is based on their behavior towards network conditions, congestion or failures, and their fast adaptation to network changes. There are two main techniques used in routing, one is “deterministic” and the other is “non-deterministic.” Deterministic techniques use routing tables to forward packets, whereas non-deterministic techniques do not use previous knowledge of network topology. Three techniques are used in non-deterministic routing: flooding, random routing, and hot potato. In flooding techniques, each node generates many copies of the same packet and sends them to all neighboring

nodes except for the sender of the packet. The problem with this technique is the huge amount of packets filling up the network. This situation may be controlled by forcing the packet to travel through a maximum number of hops. In addition, the destination should handle the reception of multiple copies of the same packet. On the other hand, flooding guarantees the quickest delivery of packets, since packets are sent through all possible combinations of routes. Random routing is another possible non-deterministic method. The selection of the next hop is based on the specific probability and weight given by the node according to different criteria. In hot potato techniques, the next hop is selected based on the least delay transmission time. The deterministic way of routing depends on computing routes for all nodes in the network according to the network topology and saving them in tables so that when packets arrive, routing tables are used to determine the next hop. These tables may also contain details about routes and their cost, transmission delay or number of hops [Bur99].

2.3 Routing Issues in Some Mobile Networks Types

LEO-based satellite network is one type of mobile networks. Satellites in this type of network move continuously with high speeds. This limits the time of satellite visibility to ground terminal to a few minutes. This continuous change in position leads to change in network topology and thus, routing becomes an important issue to discuss. Routing protocols should adapt to frequent changes in network topology and

work with almost the same efficiency as routing protocols used in static networks.

The following are some protocols used for such mobile network:

2.3.1 Virtual Topology Pattern

Wood mentioned two possible approaches in his paper “IP Routing Issues in Satellite Constellation Networks” [WCA⁺01]. The first one is *Path maintenance via virtual topology pattern*. The key to this algorithm is that it depends on the periodic form of network topology that the satellites exhibit. During a time interval between two time instances, the network is modelled as a constant graph where a routing algorithm can be used to construct routes to all nodes. This interval is chosen to be very small so that the relationships between Inter-Satellite Links (ISLs) are approximately constant. This will allow the defining of constant weight for the ISLs and the choosing of the ISL with the optimum weight or cost to construct the route. Any weight function can be chosen, such as distance between satellites, or the duration of time the satellites are active. The constructed routes in this interval are calculated in a ground terminal and uploaded to the satellites periodically [WCA⁺01].

2.3.2 Virtual Node Concept

Another algorithm explained by Wood is the *Virtual Node Concept* [RT99]. The virtual node is a fixed node relative to the surface of the earth. It contains information about the terrestrial network and ways to communicate with the nodes of that

network. In other words, this node holds the information about the network state, such as the routing tables and the channel allocation at a certain time instance. The satellite “embodies” this virtual node. As the satellite moves, a handover occurs and the information is transferred to the other satellite [RT99]. Henderson points out some problems in adopting the virtual node concept. Some of these problems appear when doing one-to-one mapping between ground terminals at the location served by the satellite and the satellite itself. The distribution of users is not the same all over the earth’s surface. Therefore, satellites serving a certain area will become congested while others are minimally loaded. Another problem occurs when a satellite is positioned in the same line of sight as the sun. This situation prevents communication and is called “sun outage.” Hence, the area served by this satellite will be deprived of communication service. Moreover, some satellites may be able to serve other locations better than the location they are assigned to, if they are located in the edge of the geographical map [Hen99].

2.3.3 Virtual Connection-Oriented Approach

Another algorithm used in LEO-based satellite networks is based on the *virtual connection-oriented approach*. It is introduced by Gounder, Prakash, and Abu-Amara [GPAA99]. The algorithm modifies the concept of ATM switching to reduce memory consumption. The approach eliminates the need to keep routing tables on-board satellites all the time. The network is viewed as a series of snapshots; each snapshot

represents a structured or fixed network. Snapshots can be determined ahead of time before it is formed, and routing tables can also be constructed in a ground terminal. This technique requires the uploading of some snapshot routing tables to the satellite. The chosen snapshots should resemble the network in the time interval between the sending time until the time estimated to reach the closest ground station where a different set of snapshots are loaded. This makes the satellite a “stored-program computer” and the routing tables are comparable to “program-instructions.” The routing algorithm adopted is similar to shortest path routing [GPAA99].

2.3.4 Datagram Based Routing Algorithms

Existing studies view satellite networks as a *semi-regular mesh topology* where satellites form a regular topology repeated periodically. Therefore, many of the routing algorithms developed tend to simplify routing algorithms by using a geographical-based technique. This strategy divides the earth space into cells served by a satellite. The logical position of the satellite is determined by the earth surface it serves. The route is calculated based on the minimum hop count to reach the destination. This technique is faced by many problems related to robustness. This idea is considered one type of virtual node concept. Another variation on this idea with less constraints is adopted by Ekici, Akyildiz and Bender [Hen99]. This algorithm assumes a certain network layout where there are a certain number of orbits distributed evenly around the earth with fixed distances between one orbit and another. The orbits cross each

other at the north and south poles. Each satellite has a longitude and latitude to identify its geographical location. Also, there is the logical location of satellites with respect to the earth surface, which is determined by the plane number and satellite number. The routing algorithm views the logical location as a hop. Using this method eliminates the need to consider the movement of the satellite. The routing algorithm works for each satellite by using decision maps created using the above assumptions and derived theorems. This algorithm chooses the propagation delay time to weigh the path between source and destination. The chosen path is the one with the minimum propagation delay time. There are two ways to decide the next hop: *Direction Estimation* and *Direction Enhancement*. Direction estimation considers all ISLs of the same length and calculates the minimum hop count path. Packets are routed in the horizontal direction to the plane which the destination satellite is in, then in the vertical direction to reach the destination satellite. However, this approach requires some enhancements because ISLs are usually of different lengths. The enhancements developed combine both the minimum hop metrics and the decision map, which can be stored on-board a satellite [EAB00].

2.3.5 Self-Organizing Protocol for Small Ad Hoc Networks

Another protocol developed for an ad hoc network is proposed by Huang et al. [HKR02]. It is a self-organizing protocol for small ad hoc networks with relatively large links, of tens of kilometers, and mobility, which is about tens of kilometers per

hour, called Layernet. This protocol is based on initial forming of a sparse tree topology using asynchronous communication as a first phase. The next phase is to have synchronous scheduling and establish more links so that the network is no longer a tree. This protocol is suitable for a network that has an unknown number of nodes, unknown node locations, and no global clock [HKR02].

2.3.6 Problems

Some problems facing the geographical-based routing algorithm are pointed out by Henderson and Katz [HK00]. One of these problems is routing in polar regions, which face some breaks in the topology and will cause packets to reach dead-ends and eventually packet drops. Another problem occurs in counter-rotating planes, because links are distorted around these planes and the ISLs there are difficult to synchronize. Some suggestions are proposed by Henderson and Katz to solve these problems. One of these suggestions is to use a geographical-based approach to reach the farthest node in the radius of the source satellite, and then to use the shortest path to reach the final destination. Due to the fact that each satellite holds only sub-graph information, each satellite may hold inconsistent information and may generate loops in the path constructed. Loop-free paths can be achieved by saving the path history and detecting the repetition of a node in the path. This idea reduces the complexity of processing to localize the information to each node. Another suggestion

is to increase the routing radius range of satellites to reduce the possibility of breakdown. This will also allow all nodes located in the polar regions to have routing information from all other nodes in this region. On the other hand, they failed to solve the problem of counter-rotating planes. None of the trials they implemented prevented packets from reaching dead-ends. In other words, to guarantee robustness using distributed geographical-based algorithms, complexity issues arise that cannot be handled with the limited processing power of small satellites [HK00]. In general, existing routing algorithms for LEO-based satellite systems are based on two basic techniques with different variations. Each of these techniques has its own problems. One technique is to pre-construct the routing tables in a ground station and upload these tables periodically or when the satellite moves from one region to a different one where the network topology is different, and the routing tables should therefore hold different information. One of the major problems using this approach is that there is a gradual transition between the old network topology and the new one in the new region that is difficult to be handled. This is because it will require more frequent processing in ground station and uploading to the satellite which is difficult considering the limited processing power and memory space on board of satellite. Also, covering all transitions is a challenging and complex process. Another technique is geographical based technique where each satellite is given a logical location with respect to its altitude and latitude on the surface of the earth. This technique

divides the earth surface into horizontal and vertical lines, and each crossing of these lines is a node, which is considered a satellite. This method avoids dealing with the satellite as a moving node; instead, it uses a mapping table of the logical locations of each satellite. This technique depends on each logical location being occupied by a satellite. In other words, this network should have enough satellites to cover the whole surface of the earth. If the number of satellites in the network is too small or they do not cover the whole earth surface, this approach will not work properly. Therefore, considering the above problems, we should consider applying the routing algorithms used for mobile ad hoc networks for satellite nodes in mobile networks. These algorithms assume that all nodes are moving randomly and that the network topology is changing all of the time. These algorithms also take into consideration the limited processing power and memory size of the mobile nodes, so they tend to be less complex than other algorithms. In the next section, some MANET routing algorithms will be investigated.

2.4 IP Routing and Mobility for MANET

The last decade has seen substantial progress in wireless communication and computer technology. As a result, mobile computers are expected to spread among global users interested in accessing Internet applications. This means that the use of Internet Protocol (IP) should be considered to support mobile communication with Internet networks. Routing for mobile hosts within the Internet environment is called *mobile*

IP, in which a mobile host can connect to the Internet without needing to have a fixed address; instead, some modifications need to be done to IP in address and network management. Routing algorithms in MANET allow mobile hosts and routers to communicate together through wireless links forming “network routing infrastructure in an ad hoc fashion” [CM99]. Using IP-layer routing and improving it to fit the mobile wireless networking will provide connectivity to the Internet, thereby achieving a key purpose of the Internet to connect different types of networks and different infrastructures all over the world. As stated by Corson and Macker, the advantage of using IP-level routing for MANET is that it allows consistent network levels over networks having different types of physical layer media [CM99].

2.5 Routing within MANET

Conventional algorithms are of two types, distance vector and link state. Routing tables in distance vector algorithms contain information on distances between the router itself and all destination nodes. This information is updated periodically to remain valid. This update is done by broadcasting messages to all neighbor nodes and waiting for their response with the updated distance values. The periodical update allows for the detection of down links quickly using the correct paths. However, it requires substantial CPU time and bandwidth to accommodate the broadcasting of update information. On the other hand, link state routing is based on keeping the whole network topology information in the routing tables of each router. The

information holds the link cost throughout the network. This information is updated periodically between all routers. Routers compute the shortest path to the destination. Packets are forwarded to the next hop according to the selected path computed. This algorithm updates the paths quickly when network changes occur. However, it requires significant CPU processing to calculate all the shortest paths and needs more bandwidth to broadcast update information [Joh94]. Conventional routing protocol algorithms have many problems when applied to mobile ad hoc networks. One of these problems is that not all wireless links are bi-directional; some are unidirectional. The conventional algorithms always assume links to be bi-directional and do not account for unidirectional links. Another point is that conventional algorithms will result in redundant routing path due to the fact that each node is a router, and many nodes can be in the range of the source. This may lead to large routing table sizes and large update packets being broadcast throughout the network. In hardwired networks, there is a small number of routers connecting two networks. That is why this problem does not appear in hard-wired networks. In addition, periodical updates for routing information wastes network bandwidth and nodes' battery power. These resources are limited in ad hoc networks. Moreover, conventional routing protocols do not account for the speed of network topology changes in ad hoc networks. This is true because changes in conventional networks are not substantial. Some links may go down, or the cost of links may change due to congestion, but these networks do

not experience as significant a change as the dramatic changes in network topology that occur in ad hoc networks. Solutions to this problem may require more frequent updates to be broadcast between the routers to converge quickly to the changes in network topology. However, this solution requires more bandwidth, which again is a limited resource in wireless networks. Johnson suggests a new approach to develop a routing algorithm that fits with the mobility nature of ad hoc networks [Joh94]. He suggests dividing the routing problem into two main parts, route discovery and routing maintenance. Routes are initially constructed using the route discovery technique. As long as network conditions are not changed, the route remains in use. When network conditions change and routes become invalid, the route maintenance technique is used to provide this information to nodes/routers so that a new route can be discovered [Joh94]. Besides this new approach for developing routing algorithms for MANET, there are some metrics that should be considered when designing the algorithm. Details for these metrics are discussed in the following section.

2.5.1 Routing Protocol Characteristics and Metrics

Corson and Macker identified some metrics to help in evaluating the routing protocol performance and appropriateness [CM99]. The stated metrics are considered general and can be applied to any routing protocol. The metrics are divided into two categories, qualitative and quantitative. Some of the qualitative metrics are: *loop-freedom*, which is a feature that prevents packets from going into loops between

nodes wasting network resources; *security*, which is another property that is important to prevent the problem of IP header changes and the redirection of packets; and *sleep period operation*, which allows the routing protocol to adapt to mobile nodes that have limited processing power. These mobile nodes stop sending and receiving packets for a random period of time to save energy. The routing protocol will need to communicate with the link layer in order to achieve this property. There are also *quantitative* measures for routing protocol performance. One of these metrics is the *End-to-End Data Throughput and Delay*. This metric measures the overall delay time for the packets to be transferred from source to destination. Another metric is the *Route Acquisition Time*, which is the time required for new route to be constructed. There are other *quantitative* metrics used in MANET, such as *Percentage Out-of-Order Delivery*, which is important for layers above IP like TCP, and *Efficiency*, which measures the internal performance of the routing protocol [CM99].

2.5.2 Types of MANET Routing Protocol Algorithms

Routing protocol algorithms in MANET are divided into three categories according to the approach followed. The first type is proactive protocols, the second is reactive and the third is hybrid. The main characteristic of proactive algorithms is that all routes to all destinations are constructed and ready whether they are needed or not. Usually, changes in the network topology are reflected in the routing tables using periodical update messages exchanged between mobile nodes. This ensures that routing

tables are consistent and carry valid routing information. When a route to a destination is needed, the routing table directly provides route. However, the problem of this algorithm is the waste of bandwidth due to the periodical update messages between all nodes. Not all the update information needs to be exchanged and not all the routes need to be constructed because they may never be used. The solution to this problem is proposed by the reactive routing protocols. An example of a proactive routing protocol algorithm is Destination-Sequenced Distance-Vector (DSDV) and Wireless Routing Protocol (WRP). The reactive protocol algorithm is a very efficient technique used for routing in MANET. The idea behind this technique is that routes are constructed on-demand. In other words, when a route is needed, control packets are flooded all over the network to find the route to the destination. This is the difference between the reactive and the proactive protocols. This technique saves a lot of bandwidth because only needed routes are constructed. After finding the route, only data packets are sent through the network, fully utilizing the bandwidth. The drawback of this algorithm is that data packet may experience some delay in transmission because of the initial route construction. In addition, a lot of bandwidth is wasted in flooding the network trying to find the route to the destination. There are many algorithms, such as Temporally Ordered Routing Algorithm (TORA), Associativity-Based Routing (ABR), Signal Stability Routing (SSR), Location Aided Routing (LAR), Dynamic Source Routing (DSR) and Ad Hoc On-Demand Distance Vector Routing (AODV),

which adopt a reactive algorithm approach [BMJ⁺98, RT99, BFH99]. Hybrid protocol algorithms are a combination of reactive and proactive algorithms. The idea is to add the benefits of reactive and proactive protocol algorithms together. In some ways, combined protocol algorithms experience some complexity in implementation. Zone Routing Protocol (ZRP) is one example of a hybrid protocol algorithm. The following is an explanation of some examples of different routing protocol algorithms used in MANET.

2.5.3 Proactive Routing Protocols

2.5.3.1 Destination-Sequenced Distance-Vector (DSDV)

DSDV is a proactive table driven routing algorithm. This algorithm is derived from the Bellman-Ford routing technique with modifications to generate loop-free routing paths. Routing tables are created in each mobile node. They contain paths to all possible destinations in the network, with the number of hops in each path. Each entry in the table is associated with a sequence number assigned by the destination. The algorithm uses the value of the number to determine if the path is valid, newly created, or old. Tables are updated periodically to assure the consistency of all tables among all nodes. Updates can either be transmitting all the table information, or only the changed information in that table. New route broadcasts contain the destination address along with the number of hops to reach the destination. Also, there is a sequence number of destinations and an additional sequence number associated with

the broadcast. The route with the most recent sequence number is always chosen to forward the packet. In case two routes have the same sequence number, the route with the smaller metric, for example fewer hops, is chosen to shorten the path [RT99].

2.5.3.2 Wireless Routing Protocol (WRP)

WRP is a proactive algorithm. The main key is that each node holds information about all nodes in the network. There are four tables to be stored in each node. Distance, Routing, Link-cost, and Message Retransmission List (MRL) tables. The MRL table entry contains the sequence number of the update message, and the vector contains a flag for the acknowledgement required from each neighboring node, as well as a list of updates sent in the update message. The update messages are used to inform neighboring nodes about network changes. Nodes send updates when receiving and processing an update or when detecting a change in a link status in the network. "Hello" messages are needed so that other nodes know it still exists; otherwise, a node is considered failed if it does not send any message within a specific period of time. New nodes also send hello messages so that neighboring nodes can add it to their routing tables [RT99]. This algorithm accomplishes the loop freedom metric and also converges quickly to network changes.

2.5.3.3 Problems of Proactive algorithms

Proactive algorithms require large overhead resulting from periodical updates. Considerable portions of the update message are used for updating routes which are not

needed and may not be used. Though this feature allows nodes to provide route information right on the spot whenever it is needed, it wastes limited bandwidth available for wireless connections [RT99]. Moreover, when the mobility rate increases, proactive algorithms will not perform well. Simulations performed by Broch et al. proved that DSDV does not perform well in delivering packets with higher mobility rates [BMJ⁺98]. These limitations discouraged the use of proactive protocol algorithms in MANET and opened the door for more investigation and enhancement of reactive protocol algorithms.

2.5.4 Reactive Routing Protocols

2.5.4.1 Temporally Ordered Routing Algorithm (TORA)

TORA is an on-demand reactive routing protocol algorithm. It is characterized by many advantages. One of these advantages is that it provides multiple routes for each source-destination pair. In addition, it is able to construct new routes quickly. Moreover, it does not waste much bandwidth and minimizes overhead, since changes of information in the network are propagated locally among neighboring nodes. The drawback is that constructed routes are not the shortest. Longer paths are preferred to save the effort of constructing new routes. TORA follows a water flow model in which links are represented by tubes, packets by water and nodes by junctions between tubes. Each node has a height measure according to its relationship with its destination nodes. If the link between two nodes is congested, the height value

is increased to indicate that packets (water) cannot go up to the next node. When a destination is requested, a QUERY packet is broadcast. The packet is forwarded from one node to another until the destination is reached or a node which has the rest of the route path to the destination is reached. This node, which receives the query packet, broadcasts an UPDATE message, including its height with respect to the destination. Each node receiving the update message changes its height to a greater value than the height of the sender node. This way creates a directed path between the source, which originally generated the QUERY message, and the node which originated the UPDATE message. If the node discovers a link failure, it sets its height to a maximum value among its neighbors and then sends an UPDATE message. If no neighbors have a valid height value to the destination, then the node discovers a new route. In addition, a CLEAR message is used when a network partition occurs and a route is no longer valid [BMJ⁺98]. TORA is suitable for large networks, but it has some drawbacks. One is the requirement of time synchronization between all nodes using GPS, which is an external hardware. In addition, there is a possibility of oscillation occurring when many nodes simultaneously detect network partitions and link failures and try to erase routes [RT99]. In a simulation conducted by Broch et al., TORA greater overhead than other algorithms. The packet overhead in TORA is caused by the IMEP layer, which requires a periodical exchange of hello messages for neighbor discovery. In addition, the control packets needed for route discovery and

maintenance contribute to the overhead. Another observation related to TORA is that the packet delivery ratio drops as the number of nodes in the network increases [BMJ⁺98].

2.5.4.2 Associativity-Based Routing (ABR)

This algorithm employs a different technique from other algorithms used in MANET. The main idea is to use a new routing metric based on the “degree of association stability”; the higher the degree, the higher the weight of the link. Each node has to send a *beacon* periodically to prove its existence. The receiving node will increase the associativity counters associated with the source node. Higher numbers indicate greater node stability. This number is reset when the node moves out of range. This algorithm is divided into three phases: Route Discovery, Route Reconstruction, and Route Deletion. Route Discovery is achieved by broadcasting a route request query searching for a path to the destination and waiting for a reply. Receiving intermediate nodes append their addresses, their associativity ticks with their neighbors and QoS information to the query packet. The next node in the path removes the associativity ticks of the previous node neighbors and keeps the associativity ticks concerning itself and the previous nodes so that the associativity ticks related to only the nodes along the path are saved. The chosen path is the one with the highest associativity tick value or, if more than one path has the same value, the one with the least number of hops is chosen. Then, a reply packet is sent along the path. As the reply packet travels, the

route is activated for each node along the path. Other routes are deactivated so that duplicate packets arriving to the destination are avoided. The Route Reconstruction phase is initiated if the source moves or the destination moves out of the path. Source movement causes a broadcast query and wait-reply process to be invoked. It also creates a message to erase the route to the following nodes along the path. On the other hand, if the destination moves, the node located immediately before it along the path erases its path and issues a local request to the destination with a timeout period. If a reply is received, the partial route is constructed from this intermediate node to the destination. Otherwise, if it times out, the request is propagated to the previous node along the path, which does the same job. Route Deletion broadcasts a message when a route is no longer used. The delete message is propagated to all nodes along the path. This message is broadcast to all nodes because the source node may not be aware of whether route reconstruction has occurred [RT99]. ABR requires beacon exchanges between nodes to ensure the validity of routes in the routing table. In addition, ABR uses a routing table, neighbor table, and seen table. These tables may have very large sizes in the worst-case scenario. As a result of using beacons, these tables will also be filled with information even if no traffic is exchanged throughout the network [Lee00]. This may cause overhead and more CPU consumption.

2.5.4.3 Signal Stability Routing (SSR)

SSR is an on-demand protocol which uses the strength of the signal between nodes to choose the route path. The stronger the signal, the more it indicates that the node's location is stable. This algorithm can be divided into two protocols, Static and Dynamic routing protocols. The Dynamic Routing Protocol (DRP) controls two tables, the signal stability table (SST) and routing table. The SST holds information on the signal strength of neighboring nodes. This information is updated by sending periodical beacons from the neighboring nodes. The strength of the signal of the received packets is recorded as either weak or strong. DRP updates the information in all tables and then passes the packet to the Static Routing Protocol (SRP). The SRP either passes the packet to the upper layers of the protocol stack or forwards it to another node. If no route to the destination is found in the table, a search request is broadcast. The packets are only propagated if they are received over a strong signal channel. The destination sends back the first arriving route request packet using the DRP, which assigns a route reply message to be sent over reversed path to the source. DRP allows nodes along the path to update their routing tables. This assumes that the chosen path is the shortest or the least congested and the fastest since it depends on the first packet that reaches the destination. It is also based on strong signal links. If the source node does not receive a route reply message within a timeout period, it changes a field in the header PREF to indicate that weak signals are accepted because

they may be the only available links. If a node along the path detects a link failure, it sends an error message to the source containing the failed channel information. Upon receiving the error message, the source initiates a new route request message to find a new path. Moreover, the source sends a path error message to intermediate nodes along the invalid path to notify them concerning the broken link information [RT99]. One of the limitations of SSR is that the discovered routes are not the optimum routes or the shortest ones. The algorithm prefers more stable paths even if they are longer to save the bandwidth required for constructing new routes. Another problem is that intermediate nodes are not allowed to send route reply messages to the source if they have the rest of the route to the destination in their routing table. This causes more delays in the route discovery process. In addition, link failures force the source to discover a new route; partial route discovery is not performed [RT99].

2.5.4.4 Location Aided Routing (LAR)

LAR is another on-demand routing protocol algorithm. The key idea behind LAR is that it uses the location information to construct the route path. There are two parameters used by the source node in this algorithm, the expected zone and the request zone. If the source has no information about the destination, then the expected zone is considered to be the whole network. The route discovery request is flooded through the expected zone, which is in this case the whole network. On the other hand, the source may have information about the destination using the GPS.

This information includes the destination's location at a certain time instance, and its speed. The source then calculates the expected location of the destination at the current time. This expected location is called the expected zone. This method reduces the size of the expected zone and hence reduces the overhead of flooding the packet throughout the whole network. The request zone is determined by the source node. This parameter is used by the intermediate nodes receiving the route discovery packet. An intermediate node checks if it is in the request zone, and if so, forwards the packet. Otherwise, the packet is discarded. The larger the request zone, the more likely it is that the route will be found, but greater overhead is required. The request zone includes the expected zone and may include other zones too. In this way, no periodic updates are needed, and routing tables do not include information about all nodes in the network. Routing tables contain location and speed information about nodes which were previously needed as destinations. If a destination is requested, the source looks for the destination in the routing table. If it is found, then it calculates the request zone and sends the route request packet through the request zone. If the destination entry is not found in the routing table, a route request message is flooded through the entire network. This algorithm does not need periodic update information to maintain the routing table. In addition, entries in the table are small and only contain the speed and location of each node. Hence, it does not use large CPU processing because each node along the path needs to do simple calculations

to determine where it should forward the packet. The LAR algorithm is scalable because it does not require major increases in the number of packets flooded when the number of nodes in the network increases [BFH99].

2.5.4.5 Dynamic Source Routing (DSR)

DSR is another reactive protocol algorithm proposed by Johnson and Maltz [BMJ⁺98]. DSR assumes that each client acts as a router. It also assumes that communication between two nodes is not necessarily bi-directional. It uses source routing instead of hop-by-hop routing. Packets hold the routing information, which means that intermediate nodes do not need to keep up-to-date information about the network [BMJ⁺98].

Functionality: The DSR algorithm consists of two phases, Route Discovery and Route Maintenance. Route discovery is the process of finding a route to a destination from a source upon demand. The source will flood the network with a Route Request packet and it then waits for the Route Reply packet from either the destination or any intermediate node that has the rest of the route information to the destination. Route maintenance is needed when a node detects a broken link or an unusable route due to changes in the network topology. This node will notify the source node with this change by sending a Route Error packet. The sender can use the other routes stored in its cache for the desired destination [BMJ⁺98].

Memory: Each host must maintain in the cache the whole path to the destination it wants to communicate with. In worst-case scenarios, if the node needs to

communicate with all nodes in the network, the routing table size will be very large. In addition, the information will be replicated across many nodes carrying the same information [BFH99]. On the other hand, in best-case scenarios when there is no traffic exchanged in the network, the routing cache size would be zero [Lee00].

CPU: DSR requires large processing power because each node in the network will perform processing to the packet to forward it. In addition, optimizations used to enhance the performance of the DSR algorithm regarding the quick response towards changes in network topology require large processing power [BFH99].

Bandwidth Utilization: DSR uses an optimized form of flooding in the route discovery process in which a node only broadcasts to neighbors, and if it receives no response, it will broadcast to the entire network, limited to 10 hops. In addition, DSR use no hello message, so overall, with respect to other algorithms, there is relatively little bandwidth overhead used [BFH99]. In other words, if no data packets need to be sent, nothing is transmitted and bandwidth is saved. This feature is good for accommodating the sleep periods of the mobile nodes.

Scalability: This protocol is not very scalable because each host must keep information on the full path to all nodes that it is communicating with. The amount of information grows linearly with the network growth. This state will grow in order of n to the power 2, where n is the number of hosts in the network. However, in most of the conventional networks, minimal changes are needed to update the tables when

the network grows [BFH99].

Path Discovery: Cost Packet overhead is 2.6 times the optimum when the network is most unstable [BFH99]. DSR is suitable for networks where the speed at which the mobile nodes move is not very high with respect to packet transmission latency. DSR is not scalable for large networks, so the network diameter should be small. It does not perform regular broadcasts of routing information, so it saves bandwidth and power consumption. In addition, DSR is capable of establishing multiple routes for the same source destination pair in case of a broken link, and this makes route recovery faster. Moreover, route optimization can be achieved in DSR [RT99].

2.5.4.6 Ad Hoc On-Demand Distance Vector Routing (AODV)

AODV can be considered a combination of both DSR and DSDV. The on-demand functionality for route discovery and route maintenance is the same as in DSR. In addition, it adds the functionality of hop-by-hop routing, sequence numbers, and periodic beacons from DSDV [BMJ⁺98]. It assumes that communication between nodes is bi-directional. In addition, it establishes loop-free routes quickly. Therefore, it performs route replies with less delay [Lao99].

The idea of route sequence is used in AODV in order not to have stale routes. These sequence numbers are considered as *virtual time*. The high numbers occur on time later than the lower sequence numbers. The sequence numbers correspond to destination nodes. These numbers are increased by time to represent network

topology changes [BFH99].

Functionality: When a source requests a destination, a route request (RREQ) packet is broadcast to neighbor nodes to search for a route. Each node adds its ID to the reverse path before forwarding to its neighbors. When the RREQ reaches the destination node or a node that has the route to the destination, it generates a RREP message and sends it to the source using the path that was constructed in the RREQ [Lao99]. Route maintenance is done by a periodic hello message with a default rate of one per second. The link is considered down if no hello message is received from the neighbor in 3 periods (3 seconds) [BMJ⁺98]. Detection of link breaks is done quickly because the timeouts used in AODV are very small. Link error information propagates to all active nodes (which used the routes recently) which is better than DSR, where all nodes receive the updated information even if the route has not been used for a long time.

Memory: Entries of the routing table have a fixed number of elements (next-hop, timer, active). This means that the largest the table size can get is $O(n)$. However, the actual size of the table is unlikely to reach the maximum size because it is an on-demand algorithm [BFH99].

CPU: No significant additional CPU consumption is used by AODV. The overhead is only related to route discovery [BFH99].

Bandwidth Utilization: Total utilization of the bandwidth is not more than 90%,

so the overhead is only 10% [BFH99].

Scalability: AODV requires less memory and less CPU processing power per node. Thus, it is considered scalable [BFH99].

Path Discovery Cost: The number of packets used for path discovery is n to the power 2 in case every node is a neighbor. The total time to discover the path is $O(n)$ [BFH99].

AODV is distinctive in that it supports multicasting. AODV uses a lower data packet size than DSR because in DSR the packet must carry the full route path, not just the destination address. Furthermore, the size of packet replies in AODV is less than that of DSR, because it only carries the destination address and a sequence number. In addition, the table size of AODV is less than that of DSR because it does not need to carry the addresses of the all nodes along each path [RT99].

2.5.5 Hybrid Routing Protocols

2.5.5.1 Zone Routing Protocol (ZRP)

ZRP is a hybrid protocol combining proactive and reactive techniques. Each node has a zone with a predefined radius around it. If the destination node is located within this zone, routing will use the DSDV protocol, which is a proactive algorithm. Otherwise, if the destination node is located outside the predefined zone, DSR is used for routing packets, which is a reactive scheme. Within the zone, if the link state changes, a notice is sent to all nodes within that zone, limited by number of hops. This saves

some overhead because the update information is sent to a limited zone only. If the destination is outside the zone, a route request is broadcast to all nodes located on the zone border. If one of the nodes on the border finds the destination in its radius or zone, it issues a route reply. Otherwise, it forwards the packets in the same way. For data forwarding, the node checks the routing table carrying information on its zone to forward the packet to the next node until it reaches the destination node or zone border. This is because the path recorded in the source routing only includes the border zone nodes and not all nodes along the path. Link deletion is broadcast within the zone, but if changes occur outside the zone, the same method followed in DSR is executed by notifying the source node about the link failure [WLTS99].

2.5.6 General Comment on All Algorithms

From the above descriptions of routing protocol algorithms used in MANET, it can be said that each algorithm has its own advantages over other algorithms, along with some disadvantages. There is no clear judgement of whether an algorithm is good or bad. However, the decision on which algorithm to apply depends on the network conditions and parameters. Network size, mobility levels, and link stability are major factors in deciding on a certain algorithm. The used experimental network in this research is not large in size and it has high levels of mobility because almost all the nodes are mobile. The algorithm needed for this type of network must be relatively simple, since the processing power of the CPU is very limited. It should

also minimize memory usage. Reactive routing protocol algorithms are suitable for this network because they satisfy all these conditions. In addition, algorithms using external hardware devices must be eliminated, so TORA and LAR will not be chosen. Simulation is done by Das et al. of performance comparisons between different routing protocol algorithms [DCY00]. In this simulation, DSR, AODV and other protocols were evaluated. The results of this simulation proved that AODV performs lower to some extent than DSR in the packet delivery ratio measured by the number of data packets delivered with respect to the number of data packets sent. This is due to the route expiration feature of AODV, which forces route discovery more frequently than in DSR, causing packets coming during route discovery to drop. Investigating the *routing load*, which is the number of bytes of routing packets per byte of data packets, shows that DSR and AODV achieve better performance with respect to other algorithms [DCYS98]. DSR surpasses the AODV algorithm in this criterion, especially with lower numbers of connections [DCY00]. DSR is preferable in our mobile network, provided that some optimization techniques are applied to guarantee better efficiency.

2.6 Detailed Description of DSR Algorithm and Packet Formats

2.6.1 Route Construction

When a source node wants to send a data packet to a destination, it looks in its route cache for an existing route to that destination. If no route is found, a route

discovery procedure is initiated. Route discovery algorithms start by constructing a *Route Request* packet which is broadcast throughout the network until it reaches the destination. The Route Request origin node is called the initiator and the destination node is called the target. Both the initiator and the target addresses, with a unique number given by the initiator, make up the packet identifier. The route request packet also holds the route list, which records every intermediate node address that receives the route request along the path to the target. When an intermediate node receives the route request packet, it first compares the target to its local address. If it is not the target, the node will drop the packet if it has previously received this packet, having the same initiator, target and identification, or has found its own address in the recorded route list. Then, the node appends its local address to the route list in the route request packet and broadcasts it with the same identification value to all neighboring nodes. Otherwise, if the node is the target, it will construct a route reply packet to the initiator filling the route list with the reversed list of recorded hops received in the route request. The route reply packet will follow the path in the route list. The assumption here is that the wireless links are bidirectional [JMH03]. In a single path DSR, the target only sends the one reply packet to the first route request received and drops the rest of the packets.

In Multiple Source Routing, the target is allowed to send reply packets to all the route requests which were received from the same initiator node [WSD⁺01].

2.6.2 Route Maintenance

In order to detect link errors, every node along the path will check if the forwarded packet reaches the next hop in the source route. One way this can be done is using the automatic link layer acknowledgement frame defined in IEEE 802.11 at MAC protocol. Such a mechanism can be built-in. Another way is called *passive acknowledgement*, achieved by a node overhearing the next hop node forwarding the packet to its neighbor. DSR also allows for an explicit software acknowledgement mechanism. This way, an explicit packet is sent back to the sender node to acknowledge packet reception between two hops [JMH03].

In case no acknowledgement is received, the link to the next hop is assumed to be broken, the route cache is updated by removing this link, and an error packet is sent back to the originator of the packet and each node previously sent this packet. When the initiator node receives a route error, it also removes this link from its route cache and sends the rest of the packets through a different route if there is more than one route existing in that table. Otherwise, it will generate a new route request to construct a new route to the desired destination.

2.6.3 Packet Description

DSR uses a special header, which contains the necessary information. This header is contained in an IP packet. The DSR header is divided into two main portions. The first is a fixed-sized 4-octets part which includes information about the next layer's

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1		
Version				IHL				Type of Service								Total Length																	
Identification												Flags				Fragment Offset																	
Time To Live										Protocol								Header Checksum															
Source IP Address																																	
Destination IP Address																																	
Options																								Padding									

Figure 2.1: IPv4 Packet Header Format

header and the length of the whole DSR header. The second portion is a variable-sized DSR options section. It can contain zero or more DSR options. The DSR header is located in an IP packet just after the IP header and before any following layer header such as the transport layer, TCP or UDP. A field in the IP header called the protocol field is used to specify that the DSR header follows the IP header. The DSR header also uses a field called next header to specify the type of protocol layer header following the DSR header, such as the transport layer header. The total length of the DSR header including the DSR options is a multiple of 4-octets. This condition is needed for alignment of the following headers in the IP packet.

2.7 IPv4 Header

The following Figure 2.1 is an IPv4 packet format. The Options field contains the DSR header, followed by one or more DSR options. It can also contain the Strict Source and Record Route header [Pos81].

The Source Address field is of size 32-bit. It contains the address of the originator

Figure 2.2: Strict Source Routing Header Format

Figure 2.3: Dynamic Source Routing Header Format

of the packet. The Destination Address is of size 32-bit. It contains the address of the intended recipient of the packet.

2.8 Strict Source Routing Header

The strict source and record route option in Figure 2.2 is one way of IP packet data transmission. It contains information on the whole route from the source to the destination so that the nodes can forward the packet hop-by-hop until it reaches the destination. The Type field is the option type, which in this case has the value of 137. The Length is the length of the whole option, including the type and length octets. The Pointer holds the offset of the next node address in the source list to be processed [Pos81].

Figure 2.4: Route Request Option Format

2.8.1 DSR Header

DSR header is an IPv4 extension header, hop-by-hop option. The fixed portion is the first three elements as in Figure 2.3. The Next Header field defines the type of header that will follow the DSR header. This field uses the same values as the IPv4 and Ipv6 Protocol field [RP94, HD98]. The Reserved field is sent as 0, and is ignored on reception. The value in the Payload Length field represents the total length of the options portion of the DSR header. The 4-octet fixed portion is not included. The Options field is a Variable-length field. The length of the Options field is specified by the Payload Length field in this DSR header. It contains a sequence of DSR options, and may contain one or more option type. The following section describes in detail each of the DSR options that can be included in any DSR header.

2.8.1.1 Route Request Option

For the route request option, there are certain fields required from the IP header. One of these fields is the Source Address. This field holds the address of the node originating the packet. Intermediate nodes will not change this field during route request processing and forwarding. Another field is the Destination Address. In the route request option, the route request is broadcast, so this field should contain the following address 255.255.255.255. The intermediate nodes should not change this field. In addition, the Hop Limit field is needed from the IP header. The Hop Limit field can take values from 1 to 255. It is used to limit the broadcast range and hence the number of nodes in the constructed route. This packet will be dropped if the source route length exceeds the number of this field. Figure 2.4 shows the Route request option format. The following are details about each of the option fields.

Option Type: Value of 2 to represent the Route request option. **Opt Data Len:** This field is of size 8 bits and is used to hold the value representing the size of the option in octets. The length value does not include the option type or the option data length fields. **Identification:** A unique sequence number identifies the request packets. The source node issuing the request packet gives a unique number to every request packet it generates. This number may be used by intermediate nodes to prevent the re-broadcasting of route request packets. **Target Address:** This field contains the address of the destination node that the route request packet is trying to discover the

Figure 2.5: Route Reply Option Format

path to. It is the end target of the Route Request. Address[1..n]: This field contains a list of the addresses of the nodes constructing the path to the destination. The address value of the Address[1] field is the first hop in the path receiving the route request packet directly from the originator of the packet. Each intermediate node adds its address to the list and increases the data length field value by 4 octets.

2.8.1.2 Route Reply Option

This option needs some fields from the IP header. The Source Address field holds the address of the node originating the packet. Intermediate nodes will not change this field during route request processing and forwarding. Also, the Destination Address is used from the IP header. In this option, this field contains the address of the source node of the route, which is the node originating the route request. Figure 2.5 shows the Reply Option format. Option Type: Holds the value of 3 to indicate a route

Figure 2.6: Error Option Format

reply packet. Opt Data Len: This field is of size 8 bits and is used to hold the value representing the size of the option in octets. The length value does not include the option type or the option data length fields. Identification: A unique sequence number relating the route reply packet to the request packet that caused the reply packet to be generated. This value is obtained from that request packet. Address [1..n]: This field is the list of the nodes constructing the source route for the route reply packet. The origin of the route is the destination node specified in the destination address field of the IP header of the route reply packet. The route indicates a sequence of hops, originating at the source node specified in the Destination Address field of the IP header of the reply packet. The ending node Address[n]: is the destination node.

2.8.1.3 Route Error Option

From Figure 2.6 route error option fields are: Option Type: This field has a value of 4 to represent the Route Error option. Opt Data Len: This value has a size of

8-bit unsigned integer. It represents the length of the option, in octets. The Option Type and Opt Data Len fields are not included in the length calculation. In the DSR internet draft, this field takes the value of 10 plus the size of any Type-Specific Information present in the Route Error. Additional extensions to the Route Error option format may also be included after the Type-Specific Information portion of the Route Error option specified above. The Opt Data Len field will show if such extensions exist. So, if the Opt Data Len is greater than the size required for the fixed portion of the Route Error and the Type-Specific Information, the remaining octets are interpreted as extensions.

Error Type: This field holds the type of error encountered. The current implementation type value is 1, which means the node is unreachable. More values of the Error Type field are reserved for future use.

Error Source Address: This field holds the address of the node initially generating the Route Error. For example, it could be the node that failed to forward a packet because of a link failure.

Error Destination Address: This field holds the address of the final target node that needs to get the route error packet. For example, this field will hold the address of the node that originated the packet that failed to be forwarded.

Type-Specific Information: Information related to the Error Type of this Route Error message. The Type-Specific Information field is defined only for Route Error messages of type `NODE_UNREACHABLE`. In this case, the Type-Specific Information field is defined in Figure 2.7.

Figure 2.7: Error Option Type-Specific Information Format

Unreachable Node Address: This field holds the address of the unreached node. In other words, it is the next hop neighbor that this node tried to transmit to and failed. A Route Error option is allowed to exist more than once in one DSR header.

2.8.2 Optimizations for DSR

Many optimization techniques are suggested to be applied to DSR to enhance its functionality and efficiency. Johnson and Maltz suggest four ways to enhance DSR. They suggest more utilization techniques for the route cache, piggybacking the data on route request packets, updating the routing cache with shorter paths found, and improved error-handling [JM96]. Nasipuri and Das propose another extension for DSR by adding a multipath support [ND99].

2.8.2.1 Routing Cache Utilization

When an intermediate node receives a route reply packet and attempts to forward it to another node, it can check the packet's contents and learn new routing information, which it can add to its cache. In addition, enabling the promiscuous mode also allows the node to add new routes to its cache when overhearing new information from packets transferred between neighboring nodes. An intermediate node can also read

the received route request packet to check if it has the path to its target destination in its cache. If the route is found, it appends it to the route in the route request packet and generates a route reply to the source node instead of rebroadcasting it to all nodes. However, a problem may occur when two or more nodes generate a route reply packet to the same source at the same time because of a collision. The solution to this problem is to allow for a random time delay for each node before sending the route reply. In addition, route loops may be generated when an intermediate node returns route replies from its cache information. This problem is solved when the node checks if its address is included in the path of the route request packet. If its address is found, the packet is dropped. To reduce overhead and the wasting the bandwidth, a node can limit the route request packet to be propagated through a maximum number of hops. The maximum number of hops is set to 1 to send the first route request. If a response is not returned within a time out period, the route request is sent again with a maximum number of hops set to a predefined number such as 10 for example [JM96].

2.8.2.2 Piggyback the Data on Route Request Packets

To reduce the time delay associated with data transmission during the route discovery process, one idea is to piggyback the data on the route request packet. Usually, if the destination does not exist in the cache, a route request packet is broadcast and the data packet is dropped, depending on the upper layers to retransmit the data

again. Instead of dropping the data packet, it is better to piggyback it on the route request packet. However, the data should be small in size to avoid large size control packet broadcasting. So, route reply packets can be piggybacked on route request packets destined back to the source. Another example of small data packets is the SYN packets used for TCP connection. The problem that may face this suggestion is that route replies may be generated from intermediate nodes and the data may not reach the destination. In other words, if the route request is received by a host which has the rest of the route in its cache, a route reply is sent to the source. In case data is piggybacked, data will not be delivered to the destination. The solution to this problem depends on the intermediate node generating the route reply. This node should forward the data to its destination using the path found in its routing cache, setting the source address to the original source address. Then, it should generate the route reply packet to the source [JM96].

2.8.2.3 Update Routing Cache

Enabling promiscuous mode allows nodes not only to add new routes, but also to detect shorter paths than the already existing paths in its routing caches. In addition, a node detecting shorter path may send a route reply packet to the originator of the route request informing it with the shorter path detected [JM96].

2.8.2.4 Handling Errors

A problem usually occurs because of network partitioning; nodes keep sending route requests to a destination that is no longer reachable, leading to link congestion and wasted overhead of route request packets that will not be useful. One solution to this problem is to allow for a time period between sending route requests for the same destination. If more packets are received by the node and headed for the same destination, these packets may be buffered until a route reply is received. Another enhancement in handling errors is to allow promiscuous mode to overhear route error packets exchanged between neighboring nodes and update the routing cache according to the error information received. Because of the existence of unidirectional links in wireless networks, some of the nodes may not be informed concerning link errors while traversing error packets to the source node. This problem may be solved by allowing the source node, upon receiving the error packet, to broadcast it so that nodes along the way are notified of the link error [JM96].

2.8.2.5 Multipath Support

Multipath extension is proposed to DSR. The idea is that the source node receives many replies from the destination node with different routes to the destination. This suggestion is intended to reduce overhead resulting from packet discovery broadcasts throughout the network. In the multipath extension, the destination node selects a set of route discovery packets coming from the same source to generate route reply

packets returning back to the originator. A *primary* source route is chosen to be the first route request that reaches the destination. The destination node then saves the primary route so that it compares it to other source routes received in the route request packets. Route reply packets are generated for the source routes that are disjoint from each other, so that failure in one route does not affect the other routes. This selection criterion limits the number of reply packets sent back to the source and eliminates the possibility of reply flooding. On the other side, the source adds all routes from received reply packets in its route cache. The source uses only the primary route for forwarding data packets until it fails. Then, the shortest path is chosen among the other routes existing in the routing cache. When all routes fail, a new route discovery packet is broadcast to all nodes in the network [ND99]. The above approach may have a problem in losing data during link failure and the propagation of error packets to the source. Nasipuri and Das suggest another modification, which allows intermediate nodes to save multiple routes to the destination in their caches [ND99]. In this case, link failures will cause an intermediate node to change the data route from the failed route to the alternate route until the route error packet reaches the source and the source switches to the alternate path from its cache. This modification can be achieved by allowing the destination node to send replies to all intermediate nodes along the path to the source. Each intermediate node receives a

reply packet from the Destination, and saves the route in its cache [ND99]. A numerical result of comparing the two modifications to DSR shows that the frequency of route discovery in single path routing is higher than the frequency in multipath routing. In addition, the second modification has a lower frequency than the frequency of the first modification. Measuring the expected time between route discovery packet generation shows that multipath support has superior performance compared with single path support. Simulation results also prove that the multipath protocol was able to decrease the number of path request packets by one-third. Also, the enhancement reduces the number of data packets lost. This is because the alternate paths already exist and the data packets that would have been lost during link failure and discovery of a new route are retained. In this situation, there is no buffering of data packets during the route discovery process. More end-to-end transmission delays are expected because in multipath protocol, the alternate paths are usually longer than the primary path. Moreover, simulations show that alternate paths are used in 30% to 40% of cases for forwarding data packets. While the primary path is usually 4 hops, the alternate path lengths are generally longer [NCD01].

2.9 Adaptive MSR Using RTT

Bandwidth limitations in mobile ad hoc networks is one of the greatest challenges facing routing protocols. One of the characteristics of MANET is the continuous change in the network topology. Conventional routing protocols are designed to

accommodate neither changes in the network topology nor the limit bandwidth. As we explained previously, DSR is one of the most popular routing algorithms used in MANET. However, DSR was originally designed to provide a single route for each destination. In case of a route error, another Route Request broadcast message is needed to establish a new route for a certain destination. Other studies on DSR suggest that the source keeps multiple routes for one destination [NCD01]. It is mentioned in one of the previous sections that simulation experiments proved that about 30% to 40% of data forwarding use the alternate routes due to route errors [NCD01]. Wang et al. propose an adaptive MSR algorithm which distributes the data packets among the multiple paths saved in the source route cache to the destination. The data distribution is based on a weight which is given to each route, based on the Round Trip Time (RTT) delay measured for that route. This has the advantage of efficient bandwidth use and thus enhances performance. In addition, it provides the applications with flexibility to choose the route to use for sending data packets to the destination according to the quality of service requirements. On the other hand, MSR requires more memory space for larger route cache sizes, and more processing complexity for the maintenance and route error handling for the additional routes per destination. However, because MSR is based on DSR, the route cache size has already reduced cache sizes thanks to the on-demand characteristic and the fact that the node will not have global network information. The additional complexity is

insignificant because in source routing, the processing is specific only to the source node, not to intermediate nodes. Intermediate nodes require no intelligence, but only forward data packets to the next hop in the source route [WSD⁺01].

Considering that MSR allows a source node to have multiple routes to a single destination, Wang et al. propose the idea of distributing the data packets among the available routes. The distribution occurs in a load-balanced manner using a heuristic function based on analytical analysis of the ad hoc network model. The load distribution of data allows more network resource utilization, reduces congestion levels, and reduces end-to-end delivery time. The only additional processing is done at the source node to choose a route for each data packet or determine the number of consecutive data packets to be sent on each route. Intermediate nodes only forward the packets to the next hop along the source route and no additional processing is required for the data load balancing. In Wang et al.'s research, they distribute load among disjoint routes. They only accept disjoint routes to achieve route independence [WSD⁺01]. A weight value is assigned to each stored route. This value corresponds to the number of consecutive packets to be sent on that route. Selecting the weight heuristic function is very challenging due to the continuous changes in the network topology. However, in their research they assume that the bandwidth delay product is constant in ad hoc networks as compared to the variation of bandwidth delay in WAN and MAN and the heterogeneity of the nodes in these types of networks. Round-trip

delay is inversely proportional to the bandwidth. So the data packet distribution among the available routes should also be proportional to the RTT delay of each route. Below is the weighing function:

$$Weight_i = \min(\frac{Delay_{max}}{Delay_i}, U) \times R$$

$Delay_{max}$ is the maximum RTT delay among all routes to one destination, $Delay_i$ is the RTT delay of a route, U is a constant to give a ceiling to the maximum number of packets sent consecutively on that route, and R is a factor to determine the switching rate between routes. In their research they chose $R = 3$ for a queue size of 50 which reduces the out-of-order delivery in TCP.

In addition, a probing mechanism is used to enhance the maintenance algorithm of DSR. A constant packet is used to update the route cache with the latest round trip delay and the validity of detecting link breakage of the paths [WSD⁺01].

2.10 Summary

Mobile ad hoc networks are a collection of mobile nodes that communicate with each other through wireless communication. Routing in this network is a challenging problem because of the limited resources the mobile nodes have and continuous changes in the network topology. Since the Internet is the widest used protocol in the world, the Internet Protocol (IP) should be considered to support mobile communication

with the Internet networks. Supporting mobility networking within the Internet Protocol is called mobile IP. Using IP routing allows for interconnection to the Internet, supports straightforward multicasting, and provides quality of service for real time applications. Routing in mobile network such as LEO satellite networks offers many alternative approaches. One is to define snapshots at different time instances. Each snapshot represents the network with a different topology structure. Routing tables are pre-calculated in a ground station and uploaded to the satellites when needed. Another approach uses external resource hardware like GPS to determine the location of other satellites and calculate the route based on the collected information. A different approach depends on the geographical distribution of satellites over the earth surface. These approaches are difficult to implement and require large processing power. Other routing algorithms used for MANET may be more suitable for our mobile network. There are many routing protocol algorithms developed for MANET. They are divided into three main categories: Proactive, Reactive and Hybrid. There is greater interest in using reactive algorithms, because they save bandwidth and the processing power of satellites. Some of the main reactive algorithms are TORA, AODV, LAR, ABR, SSR, and DSR. The characteristics of DSR are suitable to be applied in our satellite network. Some modifications and optimizations may be done for DSR to enhance its functionality and performance. Adaptive Multipath Source Routing MSR uses Dynamic source routing with multiple path extensions and distributes

data traffic among available paths in a weighted load balanced manner, depending on the round trip time delay for each route. The next chapter will elaborate on optimizations and modifications that we choose to apply to MSR. In addition, it will show how these changes will affect the functionality and performance of DSR to fit our mobile network.

Chapter 3

Enhanced MSR

Coming up with an efficient routing protocol algorithm for MANET is a very tricky process. Zhang et al.'s paper "Load Balancing of Multipath Source Routing in Ad Hoc Networks" introduces an efficient routing protocol algorithm called MSR, based on DSR with a multi-route extension. The algorithm performs traffic load balancing among available routes for one destination. The data distribution is based on a heuristic weighing mechanism which depends on RTT delay for each route [ZZS⁺02]. The MSR algorithm, as mentioned before, depends on RTT as a factor to distribute data packets among multiple routes, which provides good results for end-to-end delay and gives some indication as to the capacity level of the route. If the RTT delay is shorter, then the buffering delay in the nodes along the route will also be shorter, thus indicating that the nodes have more available capacity. This research suggests another way to evaluate the weight for each route, which is based on wireless specifications such as the Received Power and Sensitivity (RPS) of each route. The new

algorithm reduces the packet drop rate since the higher weight is given to the most stable route in the ad hoc environment. This research also investigated the possibility of using joint routes instead of using only disjoint routes to distribute the traffic. Accepting disjoint routes increase the possibility of having more routes to distribute the traffic on and might increase the network resource utilization. However, the results of using joint paths did not meet the expected performance levels and did not give good results with respect to the increase of processing overhead required for implementing this algorithm. For this reason, only disjoint paths are accepted in the RPS algorithms. This chapter provides a detailed design of the MSR algorithm and the different weighing algorithms RTT and RPS, in addition to the data structures used and the message sequence charts. The MSR algorithm is implemented in a PIX framework [Zha02].

3.1 DSR and MSR in PIX

In this research, basic DSR and MSR algorithms are implemented as a sub-layer of the IP layer in the PIX framework. The main reason for deploying IP and DSR protocols in new PIX-based communication protocol stack is the requirement to launch an experimental mobile wireless network.

In our experiment, the IP protocol layer is deployed on top of a physical link layer protocol such as the Wireless LAN protocol. The IP layer also provides routing

Physical/Wireless LAN Header	IP Header (DSR Extension)	TCP/UDP Header	Application Header	Data
---------------------------------	------------------------------	-------------------	-----------------------	------

Figure 3.1: PIX Packet Format

services to upper layers like TCP, which in turn provides a reliable byte streaming service to application level protocols such as the File Transfer Protocol (FTP) and UDP Constant Bit Rate (CBR). Figure 3.1 shows the PIX packet format.

3.1.1 Data Structures

Only two of the DSR data structures are needed for the MSR implemented in this research: the route cache and the self initiated route requests table. In standard DSR, there are many other data structures, which are only implemented in this research for the sake of simulation and comparison between the MSR routing protocol and standard protocols such as the route reply table and the recently seen route requests. Only the tables which are used in the simulation experiments are explained. For more details, please refer to [JMH03]. In the next section, each data structure, the purpose of using it, its contents, and its type will be described.

3.1.1.1 Route Cache

The route cache is owned by the IP Protocol Layer and accessed by the DSR Protocol Class. It is the routing table for the IP protocol layer, and contains routes used to

forward data packets. Each time a packet is required to be sent to a certain destination, the route cache is checked to get the source route for that destination. The container type is a map from the PIX framework template library, which contains a number of data entries. Each entry contains a unique Key which is the destination IP Address. The Data section of the entry contains the number of available paths to that destination and the source routes for each path. In addition, the path structure contains not only the source route, but also some information to be used to evaluate the weight for each path to determine the pattern of packet distribution among available routes (ie. the number of packets to be sent consecutively on each route). Figure 3.2 shows the route cache structure. More details about data distribution and the fields used in this algorithm will follow in the next section.

3.1.1.2 Self-Initiated Route Request Table

This table contains a list of all sent requests initiated by the local node. The purpose of having this table is to eliminate the repetition of route requests as data packets targeting the same destination address arrive in the IP session layer. It is also needed to re-send route request packets in case no route reply is received within a timeout period associated with the table. The container type is a map from the PIX framework template library, which contains a list of structures called Self Initiated Route Request Map. The key of the Map is the unique destination IP address, and is therefore unique. The data field contains the number of retransmission times to date. The number of

Route Cache					
Destination	numberOfRts	pathList (Length, Weight, more info, source route)			

Path List					
len	weight	bottle_neck_node	maxload	minRFS	route (Addr List)
					...
...	...	...	...	...	...

Figure 3.2: Route Cache

retransmissions is updated each time a timeout is fired before receiving a route reply. The information stored in the table allows for control of the number of route requests issued to one destination.

3.1.2 DSR Protocol Class

This class is responsible for handling the routing algorithm and processing all routing control packets. It is implemented with a PIX framework flavor. The IP Session triggers a route request broadcast message in DSR Protocol Class to establish a new route to a needed destination. Route Error functionality is initiated by the lower layer protocol, the Wireless Lan in our case, to request a route error packet to be sent if it detects a link failure to the next hop. When the IP Protocol layer receives any of the

DSR control packets, such as Route request, route reply or route error packets, a DSR Protocol class method is triggered to parse it into each of the received packets and handle it according to DSR algorithm requirements. More details on the handling of DSR Option packets will be presented in the following sections.

3.1.3 Algorithms and Data Flow Diagrams

3.1.3.1 Route Construction

We implemented an MSR algorithm which uses the DSR route construction and maintenance algorithms. The route construction algorithm is triggered as follows. A client session will call a method to allow for the pushing of data messages down an IP connection. The opened IP session checks the route cache owned by the IP Protocol for an existing route to the required destination. In case no route to that destination is found in the route cache, the data packet is buffered and the DSR algorithm is triggered to construct a route request packet to be broadcast. The DSR Class requires some parameters, such as operation code which should be a route request, and the IP address of the required destination to construct a route request broadcast message. Upon receiving the route request message, the DSR class checks in the self-initiated requests table to see if there is a pending request by another session to the same destination to save unnecessary route request broadcasts. If there no route request packet has already been issued, the destination address is inserted into the self-initiated requests table, and the the route request is generated and sent over the

Wireless LAN session. Figure 3.3 shows the route request generation procedure. The route request packet format conforms to the standard DSR packet format described in the previous chapter.

Figure 3.3: Generate a Route Request

In the PIX framework, an intermediate node will receive a route request packet

first in the physical layer session. This session will create a PIX Message structure and will call a method on the upper layer protocol, in our case the IP Protocol, to handle the received packet. The IP protocol will examine the received message and will determine if the packet is a DSR control message. If so, it will call a method in the DSR Protocol Class to handle the packet. When DSR receives the request message, it checks if its own local IP address matches any of the IP addresses in the source route to detect loops. If so it drops the packet. Then, it will check if the target address matches its own local address. If so, it will start the procedure of constructing a route reply heading back to the originator of the route request message. Figure 3.4 illustrates the route request handling procedure.

Algorithm 3.1.1: INCOMING PACKET(*pkt*)

```

if receivedPacketOptType = routeRequest
    then {
        if myAddr in sourceRoute
            then discardPkt
        if target is myAddr
            then { constructRouteReplyPacket
                  sendRouteReplyBackToSource
            }
        else if No.Ofaddresses < hopLimit
            then { addMyAddrToRteReqPacket
                  broadcastRteRequestPacket
            }
        else discardPacket
    }

```

The node first reverses the source route list, then updates the source and destination addresses with its own IP address and the initiator IP addresses respectively. If the node is not the target of the route request, it will add its own IP at the end of

Figure 3.4: Handle Incoming Route Request

the source route list, update the packet length, then broadcast the message by sending it to the lower layer opened session. Intermediate nodes in our implementation do not check if the route request was seen before, as in standard DSR. In standard DSR, intermediate nodes drop a route request if a similar route request has passed by this node previously, to reduce the traffic resulting from broadcasting requests. However, in our implementation we have allowed intermediate nodes to forward all packet requests sent from the same initiator heading to the same target. This is done to serve the purpose of providing multiple routes at the initiator. Actually, this method will allow the source to receive all possible routes to the target. Although this will increase the network load during route construction, the additional number of routes collected are needed for data traffic distribution among multiple paths. This in turn might reduce the possible additional broadcasts of route requests in case of future route errors. For example, consider the following situation where we have a network in Figure 3.5. There are two disjoint routes between node S and node D . The two routes are (S, N_1, N_2, D) and (S, N_3, N_4, D) . When S broadcasts a route request message, N_1 receives it, processes it, and then broadcasts it. If N_3 in this case receives the route request coming from N_1 before receiving the request from S due to distance differences or some propagation delay factors, for example, N_3 will process the request coming from N_1 and forward it to N_4 . Then, it will drop the second request coming from S because it has seen it before. At the end, S will get

Figure 3.5: Network Configuration Example

two route replies for two joint routes (S, N_1, N_2, D) and (S, N_1, N_3, N_4, D) . In the basic algorithm, joint routes are not accepted, so only one route will be kept. This is the reason we allowed a node to keep an already seen route request packet, so the source node receives all possible routes to a desired destination.

Figure 3.6: Route Reply Packet Format

The route reply packet is constructed at the target node. This experiment makes use of the reserved field, which is ignored at reception in the standard DSR, to hold

the number of hops left in that source route to reach the destination (initiator of the Route Request), so we can calculate the next hop easier and more quickly. This field is called `segLeft`. This idea is borrowed from NIST DSR Model [oSN02]. Figure 3.6 shows the used route reply packet format. The target node fills this field with the number of nodes in the source route. The reply packet then follows the path that the request packet passed through. Each intermediate node along the path receives the route reply at the link layer. The link layer session creates a message thread and sends it over to the upper layer protocol, in this case, IP Protocol. The IP Protocol examines the packet and determines whether it is a DSR control packet; then, it calls a DSR Protocol class method to process the reply packet. In the DSR Protocol level, the `segLeft` field value is checked; if it is greater than 1, then the next hop address is determined and `segLeft` is decremented. If any of the data forwarding optimizations are enabled, then proper information is updated in the route reply packet. This will be explained in more detail in the next section. When the route reply reaches its destination, and the `segLeft` value is 1, the node saves the route in its route cache. The DSR Protocol triggers the IP session to send the queued data, which is waiting for a route to be constructed to that destination address. The node will also delete the destination entry from the self initiated requests table, since it received the reply to that request. Figure 3.7 illustrates the handling process of an incoming route reply message.

Figure 3.7: Handle Incoming Route Reply

Algorithm 3.1.2: INCOMING PACKET(*pkt*)

```

if receivedPacketOptType = routeReply
  then {
    if target is myAddr
      then { addRouteToRouteCache
            TriggerIPToSendWaitingDataStoredInSendBuffer
          }
      comment: assume bi-directional links
    else { getNextHop
             $segLeft \leftarrow segLeft - 1$ 
            forwardPacketToNextHop
          }
  }

```

3.1.3.2 Route Maintenance

This experiment follows the implementation of the standard DSR maintenance algorithm. In PIX framework, it is assumed that built-in link layer acknowledgement mechanism exists and there is no need to use the explicit acknowledgement option in DSR. In addition, this research suggests the use of beacon packets that can be sent periodically over the existing routes to update the route cache with route information. This is the idea that is borrowed from Wang et al.'s research [WSD⁺01]. The collected route information is used for data forwarding. The format of beacon packets and their content will be explained in a following section.

A route error packet is sent to the initiator of the packet, which is the origin of the source route, when a link failure is detected. The route error packet has the same format as the standard DSR route error option. When a node receives a route error at the DSR Protocol layer, it will check whether it is the target of the packet itself, and will remove all the routes containing the erroneous node from its route cache.

Figure 3.8: Handle Incoming Route Error

If no more routes remain in the route cache to that destination, then a new route request is broadcast to establish new routes to the required destination. The new route request is issued, assuming that a route to the destination is still required to deliver the rest of the data packets. This is represented in Figure 3.8.

Algorithm 3.1.3: INCOMING PACKET(*pkt*)

```

if receivedPacketOptType = routeError
    then {
        if target is myAddr
            then {
                removeRteFromCache
                if alternateRteInCache
                    then useNewRte
                    else generateNewRteReq
            }
        else {
            comment: deal with packet as route reply packet
            getNextHop
            segLeft  $\leftarrow$  segLeft - 1
            forwardPacketToNextHop
        }
    }

```

3.1.3.3 Data Forwarding

For data forwarding, this experiment chooses the Strict Source Routing algorithm. It goes along with the DSR algorithm, since the DSR provides source routes at the source nodes. The SSR algorithm also requires minimal processing at intermediate nodes, saving battery power and memory usage, which are limited resources in the wireless nodes. The data forwarding procedure is illustrated in Figure 3.9.

Figure 3.9: Handle Incoming Data Packet

Algorithm 3.1.4: INCOMING PACKET(*pkt*)

```

if receivedPacketOptType = Datapacket
  then {
    if target is myAddr
      then sendPktToUpperLayerForHandling
    else {
      getNextHop
      forwardPacketToNextHop
    }
  }

```

In the PIX IP Protocol layer, the data packets are distributed among the multiple routes available in the route cache for one destination. The number of packets sent over one of the routes is proportional to the weight given to that route. There are two different ways implemented for evaluating the routes weights. The first algorithm is borrowed from Wang et al's research paper [WSD⁺01]. The following section contains a detailed description of the route weighing algorithm and motivation.

3.2 Adaptive MSR Using RTT (WLB-RTT)

In this algorithm we use Wang et al's idea of distributing data proportional to the Round Trip Time delay for each route. We follow the same weighing heuristic function they use in their research [WSD⁺01]. In our implementation, the route cache is a MAP structure designed within the PIX framework. The PIX framework will be explained in more detail in the simulation environment section in the next chapter. The Map structure is a light weighted searchable hash table. Each entry in the route cache has a unique key which is the destination IP address. The entry also contains the number

of routes available to that destination and a list of routes. Each route in this list contains the source route, RTT delay, weight and other information necessary for the route weight evaluation used for data forwarding. Figure 3.2 shows the route cache structure. In the previous section shows the way the route cache is designed. In this algorithm, probing packets are sent periodically over the available routes to update the route cache with the round trip delay time. It is also useful to detect route errors and link breakage early [WSD⁺01].

3.3 WLB-RTTJ

In Wang et al.'s research, they distribute the load among disjoint routes. They only accept disjoint routes to achieve route independence [WSD⁺01]. However, we introduce a new approach, in which we accept joint routes. Accepting joint routes gives an opportunity to increase the number of routes to one destination. On the other hand, if we are to distribute the load among the joint routes, we might overload the joining node. This problem leads to the idea of collecting information about the bottleneck node in each route. We reject the routes if their joining node is the bottleneck of one or both of the routes. The information on each route bottleneck is saved in the route cache along with the source route. Please refer to Figure 3.2 in previous section.

The bottleneck is chosen by measuring the service time that the route reply packet

experiences in each intermediate node along the source route and determining the maximum delay. The node which has the longest service time, taking into consideration the buffering delay time, stamps its source route index in the packet along with the time value that the packet spent. Since the route reply packet size is constant along the route, greater service time means less available capacity at the node. From [ZZS⁺02]:

$$AverageServiceTime = \frac{AverageLengthofthePacket}{CapacityoftheNode}$$

When the target node receives the route request packet, it constructs a route reply to send it back to the initiator. We allocate two additional bytes in the route reply packet. One byte is to hold the index of the intermediate node, having the minimum capacity or the largest service time, in the source list. The other byte is used to hold the longest delay value stamp. Here is how the algorithm works; each node has to determine the service time each packet takes from the time it is received till the time is ready to be forwarded. When a node finishes processing the route reply packets and is ready to forward it, it checks the stamped delay time in the packet. The stamped value represents the maximum delay time the packet has experienced along the path so far. If the stamped delay time is greater than the time the packet experienced in this node, then the node forwards the packet normally. Otherwise, the node replaces the delay stamp with the current delay value, updates the packet with its index in the source list, and then forwards the packet to the next hop in the path. The following

Figure 3.10: New Route Reply Packet Format

Figure 3.10 shows the tail of the Route reply packet which contains the bottleneck information.

Probing packets called *Beacons* are sent periodically over the available routes to refresh the route cache with updated information. RTT delay is recalculated upon the reception of each beacon. The beacons are also used to collect certain network status information, such as the bottleneck node in the route and its delay. We choose to use the *Beacon* packet as a DSR option packet type. We use an option type that is different from all the defined option types in the DSR internet draft document [JMH03]. The node receiving this packet is able to drop the packets if it does not handle this option. This is achieved by choosing the most significant 3 bits to be 0's. Setting the most significant bit to 0 indicates that the node receiving this packet should not send a route error if it cannot handle the option. Having zeros in the two following bits in the Option Type value tells the node to ignore that option. The

affected significantly by each node's signal strength and the distances between nodes. These facts lead to the idea of distributing the data packets among routes which are relatively stable and have higher signal strength than others. This allows to reduce the probability of packet drops and eventually enhance network performance. It is reasonable to assume that the stronger the signal of the route, the more stable the route is. The signal strength is the key concept in this research. Hence, it is important to understand the factors that affect the signal strength. According to Breezcom, signal strength is defined in terms of the following parameters [Inc]:

$$Si = P_{out} - Ct + Gt - Pl + Gr - Cr$$

Where P_{out} is the output power of transmitted in dBm, Ct is the transmitter cable attenuation in dB, Gt is the transmitting antenna gain in dBi, Gr is the receiving antenna gain in dBi, Pl is the path loss in dB, Cr is the receiver cable attenuation is dB, Si is the received power level at receiver input in dBm, and Ps is the receiver sensitivity in dBm.

However, the signal strength is not the only factor that affects the wireless environment. Receiver sensitivity is another element that may contribute to the performance of the wireless network. For example, all of the above factors may result in a strong signal, yet receiver sensitivity may require an even stronger signal, leading to packet drop or signal loss. The sensitivity of a node is defined as the minimum RF signal power level required at the input of a receiver for certain performance. We need to

combine both the received signal strength and the receiver sensitivity in one equation to get one value which represents a relative measure to the link budget with respect to strength of the signal. The value suggested is called Received Power and Sensitivity (RPS).

$$RPS = ReceivedPower - ReceiverSensitivity$$

This leads to a new algorithm named Weighted Load Balancing using RPS (WLB-RPS). It uses a new route weighing mechanism that makes the MSR protocol more adaptive to network errors resulting from loss of signal strength. This algorithm accepts only disjoint routes for data distribution. RPS parameter is inversely proportional to Bit Error Rate (BER). The higher the RPS value the lower the probability of packet drops due to bit error occurrence. If the RPS value is less than 0, the packets might still be transmitted but with poor quality. In this thesis we only accept routes with positive RPS values to increase the quality of delivery. The received power and receiver sensitivity are two parameters that can be obtained through system calls in Linux environment.

3.4.1 Route Construction

As discussed before, route construction is the same as in DSR, but the difference will be during the route request message propagation. In this algorithm, when the target node receives the route request, it constructs a route reply to send it back to the initiator. One byte is used to hold the RPS value. The target node inserts an invalid

Figure 3.12: Reply Packet Format with the RPSmin extension

Figure 3.13: Request Packet Format with the RPSmin extension

value (eg. a very high number) in this field. Each intermediate node in the source route then calculates its own RPS value as described above and compares it to the stamped RPS value. The minimum RPS value is needed to be kept along the path so it can be used as the weight of the route. If the stamped value is smaller than the local RPS value, then the node forwards the packet normally. Otherwise, the node replaces the RPS stamp with its local RPS value, and then forwards the packet to the next hop in the path. At the end of the route, when the reply packet reaches the initiator node, the packet will contain the minimum RPS value, which will be saved in the route cache to be used as a parameter in evaluating the route's weight. Figure 3.12 and 3.13 show the tail of the route reply packet, which contains the RPS information and the route request packet respectively.

Algorithm 3.4.1: WLB-RPSMIN INCOMING PACKET(*pkt*)

```

if receivedPacketOptType = routeRequest
    if myAddr in sourceRoute
        then discardPkt
    if target is myAddr
        then { constructRouteReplyPacket
              sendRouteReplyBackToSource
            }
        else if No.Ofaddresses < hopLimit
            then { addMyAddrToRteReqPacket
                  evaluateLocalRPS
                  if pkt.RPS > localRPS
                      then pkt.RPS ← localRPS
                      broadcastRteRequestPacket
                  else discardPacket
            }

```

3.4.2 Route Maintenance

Beacon packets will be sent periodically over each one of the available routes to refresh the RPS value for each route. This will help to keep the route cache updated with recent information about the strength of the route. The beacon packet format is shown in the previous section in Figure 3.11.

3.4.3 Data Forwarding

Like in WLB-RTT algorithm, we use a heuristic function to calculate the number of packets to be send consequently over a certain route. We also choose a threshold value, below which we do not use the route in packet forwarding unless there are no alternative routes available.

$$Weight_i = \min(\frac{RPS_i}{RPS_{min}}, U) \times R$$

For a route i , $Weight_i$ is weight given to route i . The RPS_{min} is the minimum RPS value among all routes to one destination, RPS_i is the RPS value of a route i , U is a constant to give a ceiling to the maximum number of packets sent consecutively on that route, R is a factor to determine the switching rate between routes. In our implementation we keep the values for R and U constants the same for all algorithms to achieve more accurate and fair results.

Figure 3.14: Network Configuration Example

3.5 Adaptive MSR Using Cumulative RPS (WLB-RPScum)

A closer study of WLB-RPSmin algorithm shows a limitation in some cases. Taking the minimum RPS link value in the route does not necessarily reduce the number of packet drops. The packet drop rate is inversely proportional to the RPS value, as mentioned earlier. The example in Figure 3.14 represents a small network configuration where two disjoint routes connect a source and a destination. The WLB-RPSmin algorithm will choose route (S,N1,D) having minimum RPS is 3 over route (S,N2,D) which has minimum RPS value of 2. The probability of packet drop in the leg (S,N2) is inversely proportional to $RPS = 8$. The probability of packet drop in the leg

(N2,D) is inversely proportional to $RPS = 2$. So the probability of packet drop in route (S,N2,D) is inversely proportional to 16.

$$Prob.of Packetdrop(S, N_1, D) \propto \frac{1}{RPS_1 \times RPS_2} = \frac{1}{3 \times 4} = 0.083$$

$$Prob.of Packetdrop(S, N_2, D) \propto \frac{1}{RPS_1 \times RPS_2} = \frac{1}{2 \times 8} = 0.0625$$

From this analysis, the algorithm should have preferred route (S,N2,D) over (S,N1,D) since the probability of packet drop is lower. A suggested solution to overcome this limitation is to develop a different model to evaluate route weights according to cumulative RPS value collection. The route request packet in the enhanced algorithm could collect the product of the RPS values of all the links in the route's path, rather than the minimum RPS value among all links. However, the product of the RPS values is a huge number that will not fit into one byte in the route request packet. Using a logarithmic value would reduce the value size.

$$\log(RPS_1 \times RPS_2) = \log(RPS_1) + \log(RPS_2)$$

So, the source will receive the summation of the logarithmic value of the RPS of all legs in the route, or cumulative RPS_{sum}. The larger RPS_{sum} is, the higher the RPS product will be and the lower the probability of packet drop in that route. The source node will then weigh the route according to the RPS_{sum} value. In the previous example, the modified algorithm will result in:

$$RPS_{cum}(S, N_1, D) = \log(4) + \log(3) = 0.602 + 0.477 = 1.079$$

$$RPS_{cum}(S, N_2, D) = \log(8) + \log(2) = 0.903 + 0.301 = 1.204$$

The selected route will be (S,N2,D,) since it will have a lower probability of packet drop than the other route. The following pseudocode explains how the route request packet is handled.

Algorithm 3.5.1: WLB-RPSCUM INCOMING PACKET(*pkt*)

```

if receivedPacketOptType = routeRequest
    then {
        if myAddr in sourceRoute
            then discardPkt
        if target is myAddr
            then { constructRouteReplyPacket
                  sendRouteReplyBackToSource
            }
        else if No.Ofaddresses < hopLimit
            then { addMyAddrToRteReqPacket
                  evaluateLocalRPS
                  pkt.RPS ← pkt.RPS + log(localRPS)
                  broadcastRteRequestPacket
            }
        else discardPacket
    }

```

Chapter 4

Simulation and Results Analysis

As explained in previous chapters, the WLB algorithm can use a modified route weighing scheme to improve the performance of applications. The modified weighing mechanism introduced in this research utilizes collected wireless configuration parameter information to anticipate the packet drop rate and hence reduce it. Simulation is used to evaluate the modified algorithm under different network mobility configurations, and then compare it with the original WLB algorithm and the basic DSR algorithm. The following outlines further details concerning the simulation environment configuration, performance metrics, test cases used to qualify the modified algorithm and the results analysis.

4.1 Simulation Environment

In this research, basic DSR and MSR algorithms are implemented as a sub-layer of the IP layer in the PIX framework. Carleton University has developed, using C++, software that is used to provide an experimental environment for protocol

implementation in mobile and wireless communications. PIX is the name given to this software, which like x-kernel provides a thread-per-message model and also provides several supporting libraries, including map library, message library, and event library. What distinguishes PIX is its use for Generative Programming (GP) and its improved configurability [Zha02].

Choosing between ready-made simulators and building our own simulator was an early decision made in this research. Going with an already existing simulator would have meant that less implementation effort would need to be expended, and that a comparative set of results would have been possible. On the other hand, building a new dedicated simulator would have allowed the running of the simulation in user mode outside the Kernel, which in turn would facilitate the debugging and logging process. Furthermore, this latter option could be tested using original code instead of having to re-implement the code in a simulator's pseudo-language, allowing for more integral simulation. This research was done based on a dedicated simulator for the afore-mentioned advantages; the extra implementation effort will be justified by further study by the research community.

The PIX stack in this simulation consists of an application layer, IP layer and link layer. The application layer is a file streaming protocol. The application protocol is composed of client and server sessions. Each client session is responsible for streaming a file stored in local disk space to the network through the lower layers. This session

is also responsible for dividing the file into message chunks. In this simulation, the message chunk size chosen was 900 bytes, so that the packet size did not exceed the allowed IP MTU size after adding the IP header to the data packet. The file size to be streamed was set at 1 Megabyte. On the other hand, the server session is responsible for storing the incoming stream of data into a file stored on the local disk.

The link Protocol is called QLP [Ela03, ELBK03]. It is developed to allow inter-process communication and the sending of data between simulated network nodes on a single machine. It uses POSIX message queues. Each session is assigned a message queue identifier and has a map of configured neighboring nodes message queue identifiers. This setup facilitates configuring the different network topologies used in this simulation. The QLP layer, however, does not guarantee packet delivery and does not inform the upper layers concerning packet send errors.

IP Protocol layer contains the implementation of the routing algorithm DSR used for route construction and maintenance. Data forwarding uses the WLB algorithm being tested. For the purpose of simulation, additional algorithms are implemented to simulate packet loss, and congestion is represented by delay. Packet loss is simulated in two different situations. The packet is dropped by the receiving node if its configured sensitivity is greater than the senders configured transmission power. In order to simulate the wireless environment and the received power, the packet is stamped with the senders transmission power and the receiver node checks the stamped value

and compares it with its own sensitivity to determine if the packet can be forwarded or should be dropped.

4.2 Simulation Configuration

The simulation configures a network topology consisting of mobile nodes. Among the 50 nodes, a connection is opened between a source and destination. The configuration used in this simulation is shown in Figure 4.1. A source node and a terminal node have a communication stack which is composed of an Application protocol layer, an IP routing protocol layer, and a Link protocol layer, respectively, from top to bottom. Router nodes which exist between the source and destination have the same communication stack, except that they do not contain an application layer, since this layer is not used in the router nodes. Each network node is run as a separate multithreaded heavy weighted process. All node processes run on the same physical machine.

The source node is configured to establish a connection session to the destination node and streams a data file, which is configured to be 1 Megabyte in size, from physical storage disk. The flow of data is continuous with no delay. In this simulation, 20 connection sessions are opened between the source and destination. Each data packet is of size 900 bytes, to have the maximum IP packet size after the insertion of the IP header.

The destination node at the end of file transfer expects a token packet as a marker

Figure 4.1: A simulated network

of the end of file transfer. This special packet is never dropped, and contains the start of transfer time stamp and the file size originally sent by the source node. This packet is needed to help the destination calculate the total time duration for the file transfer and the number of packets received and dropped per connection.

In this simulation, each run established 20 concurrent source-destination connections. The connections are established one by one, each one second apart. The reason for having multiple concurrent connections that are established consecutively is to observe the behavior of the algorithms under accumulated traffic. The traffic is generated by the adding connection establishments during time each run. In this simulation, the connection between each pair of nodes is considered to be a bi-directional communication channel. Hence, the physical link layer is configured to have bi-directional communication links.

4.2.1 Mobility Model

This simulation model is a simplified derivative of the Random Walk mobility model, which is considered one of the widely used synthetic models [CBD02, BCSW98]. This simulation model reflects mobility to a certain extent. It is an abstract model where certain details are ignored.

This research use RPS parameter to determine the distribution pattern of the data traffic. Since the path loss, and hence the RPS value, is calculated based on the absolute distance between two neighboring nodes, the simulation does not consider the exact direction of a node's movement. The direction of a receiver node is either towards its neighboring sender node or away from it. We only simulate a random increase or decrease of the absolute distance between each two neighboring nodes. The distance between two neighboring nodes is calculated in terms of the elapsed time assuming constant speed. There are no pause times between changes in a node's movement direction.

4.2.2 Simulation Parameters

Each node in the simulated network is assigned 3 values representing the delay mean value, sensitivity and transmission power. These values are needed for the testing of each data forwarding algorithm and the comparison of their performances. Each of the above parameters is assigned randomly to each node using an exponential distribution. The values range between 0 and 255. There is also a fourth parameter

called path loss used to simulate nodes movement. The first parameter is the delay mean value t_i , which is a constant value assigned to the node at initialization. As each packet passes by a node, it is delayed for a random amount of time, exponentially distributed with mean t_i . This way of modelling is used to introduce background traffic and to compensate for not establishing multiple source destination connections. In this simulation we assume that each node receives data from random number of nodes and is affected by background traffic which randomly affects the queuing delay of that node. The second parameter represents the wireless sensitivity of the node. It configures the minimum received power that the node can accept for a valid data packet. The packet should be received at a power level higher than the node's sensitivity in order to avoid being dropped. This parameter is also a constant value given to the node at initialization. The third parameter is the transmission power of the node. This value is randomly generated and assigned to each node at initialization time. Since we do not have a real wireless network, the received power level cannot be determined, except using simulation. In this simulation, each node sending a packet will stamp its own transmission power onto that packet. The stamped transmission power value helps the receiving node to compare it to its own sensitivity and determine whether the packet should be dropped or accepted. To simulate node movement, another parameter is used, called the path loss. The path loss is a value that increases or decreases over time, depending on the movement of the node, either away from the

neighboring node or closer to it. The path loss parameter is directly dependent on the distance between two nodes and is evaluated by multiplying the speed of the node (10 m/s) with the elapsed time. If the two neighboring nodes are moving away from each other, the path loss is decremented from the difference between the stamped transmission power and the node's sensitivity to determine whether the packet will be dropped or accepted and added if the nodes are moving towards each other. To determine whether the path loss is incrementing or decrementing over time, a random boolean parameter is assigned to each link connecting two nodes.

The above configuration helps to simulate a mobile wireless network, to enable the study of the differing behavior of the single path basic DSR algorithm, MSR with WLB-RTT, MSR with WLB-RTT accepting joint paths, and MSR with WLB-RPS.

4.2.3 Performance Metrics

In this simulation we use some performance metrics suggested by Wang et al. in [WSD⁺01].

Packet Delivery Ratio Number of packets sent over number of packets received.

Throughput Number of packets received over total transfer delay time.

End-to-End delay Total duration time for file transfer.

Delay per packet Total file transfer time divided by the number of received packet.

Number of packets dropped Total number of packets dropped.

4.3 Test Cases

Using an expanding network, and a network with elements of both expansion and convergence, we introduce additional random packet dropping to address other wireless environment conditions. Mobile network nodes are moving, hence a network is expanding when nodes are moving away from each other; alternatively, the network may converge or exhibit some combination of the two. The three experiments help to study the effects of differing network behavior on algorithm performance.

4.3.1 The First Experiment

In this experiment, it is assumed that the network is expanding and all the nodes in this network are moving away from each other. The path power loss increases by time between all nodes. This experiment does not correspond to a real situation. It is a validation step, since it represents the ideal case for the new WLB-RPS algorithm.

4.3.2 The Second Experiment

Since in reality, most of the mobile ad hoc networks have combination of expanding and converging nodes, this second experiment simulates such a network. An average of 50% of the nodes in this network move toward each other and the rest move away from each other. This setup allows for a variety of received signal power at each node, and is thereby closer to simulating a real mobile network.

4.3.3 The Third Experiment

This experiment uses the same network configuration as experiment 2, but introduces random packet loss in each node to encounter bit errors and other conditions that affect the wireless network behavior. The packet drop in each node occurs as an exponential distribution with mean equal to the sensitivity parameter assigned to the node at initialization.

4.3.4 The Fourth Experiment

This experiment studies the behavior of each algorithm under different packet drop rates. The intent of this experiment is to observe the adaptability of each algorithm to the increase and decrease of packet drop rates. It has the same network configuration as experiment 3, but uses different packet drop rates. In this test, drop rates vary from 10% to 90%. The drop rates vary by multiplying the random exponentially distributed drop rate with mean equal to the sensitivity of the node by a constant parameter α to increase or decrease the drop rate accordingly. The chosen α values are 100, 70, 50, 20, and 10. The higher the α value, the lower the percentage of packets dropped.

4.4 Results Analysis

In this section, the results of the experiments are documented and explained. Analytical comments are also applied to each result of the experiments. Throughout this

chapter, each of the tested algorithms is given a short name for easier tracking. The original algorithm, the standard DSR with single path support, is referred to as *Basic DSR*. The MSR weighted load balancing algorithm using RTT for route weight evaluation [WSD⁺01] is called *WLB-RTT*. The modified WLB algorithm accepting joint routes is referred to as *WLB-J*. In addition, the weight load balancing algorithm using received power and sensitivity parameters as a weight evaluation parameter is called *WLB-RPS*. There are two flavors of the last algorithm. The first flavor evaluates the route weight using the minimum RPS value among the links in each route. This flavor is referred to as *WLB-RPS_{min}*. The second flavor collects RPS information cumulatively among all the links in each route, and is referred to as *WLB-RPS_{cum}*. Each modified algorithm will be handled as a different algorithm to be evaluated and analyzed.

In each experiment, the total transfer time, total number of packets received and total number of packets dropped are tracked. The complete measurement data, along with their standard deviation and 95% confidence level, are reported in Appendix A. The collected data are used to evaluate each of the following performance metrics: packet delivery ratio, throughput, end-to-end delay, delay per packet, and total number of packets dropped. Each of the above performance metrics are evaluated for each algorithm tested.

Figure 4.2: Experiment 1: Total Delay

Figure 4.3: Experiment 1: Delay per Packet

4.4.1 Experiment 1 Expanding Network

In this experiment, each node is moving away from its neighboring nodes. Hence, the received power for each packet decreases over time due to the continuous increase of path loss power between two communicating nodes. This means that the packet drop rate increases over time. This experiment allows a comparison of the behavior of each algorithm under extreme expanding network conditions. Choosing RPS values to evaluate the route weight helps in choosing the routes which have the best signal strength, and hence the longest life-time. In other words, choosing the routes with the best RPS values enhances performance by reducing the probability of packet loss, since the packet errors are inversely proportional to the RPS value. However, using RTT as a weighing mechanism will not help detect this network behavior and derive an expectation of route life-time, exposing the algorithm to the possibility of an increased packet loss rate and more network overhead due to the exchange of route error packets and the need to evaluate new route construction to complete the packet delivery.

4.4.1.1 Total Delay and Delay per Packet

The graph in Figure 4.2 shows the total time required to transfer a 1 Mbyte file from a source to a destination using the different algorithms under study. Each plotted point represents the total transfer time spent by one established connection between one source and one destination. The figure shows that the total time per connection

Figure 4.4: Experiment 1: Total Packets Dropped

Figure 4.5: Experiment 1: Delivery Ratio

increases when an additional connection is established. This is because the network becomes more congested with each additional connection, and hence each subsequent connection requires more time.

The total time is also proportional to the number of packets received. The more packets reach the destination, the more time is spent during total packet transfer. Since WLB-RPScum is capable of delivering more packets because it has less packet drop rate, taking into consideration that the same number of packets is originated from the source through all runs. As shown in Figure 4.4, the total duration is longer than that of the rest of the algorithms. WLB-RPSmin has the second longest duration, since it delivers fewer packets than WLB-RPScum, but more than WLB-RTT, WLB-J, and Basic DSR. Basic DSR and WLB-J have similar results, since using joint routes for packet delivery may congest some of the joining nodes and hence affect the performance of all of the joint routes through this node. In Figure 4.3, the delay per packet is represented and shows that WLB-RPScum presents the smallest delay per packet.

4.4.1.2 Total Number of Packets Dropped and Delivery Ratio

Figure 4.4 shows the relationship between connection numbers and the number of dropped packets using all the tested algorithms.

WLB-RPScum performs best, having the least number of packets dropped. Second is the WLB-RPSmin algorithm, since it also weighs the routes with respect to their

Figure 4.6: Experiment 1: Throughput

minimum RPS value. Both WLB-RTT and WLB-J, which weigh the routes according to their RTT, experience greater packet dropping rates. This happens in the condition where the fast routes chosen are with nodes having less RPS quality. Figure 4.5 shows the delivery ratio of each algorithm. Similar to the drop rate graph, the delivery ratio presents the same behavior.

4.4.1.3 Throughput

Figure 4.6 shows the throughput of each algorithm tested. The general behavior shows that the throughput decreases with higher connection numbers since the more connections are established the more congested the network becomes, thereby affecting the throughput of the additional connections. The maximum throughput is achieved

by WLB-RPS_{sum}, since this route weighing mechanism takes into consideration the RPS value of each leg in the route. This alternative gives more accurate information and exercises better judgment in choosing which route to deliver the packets through. Next is WLB-RPS_{min}, which has better throughput than both WLB-RTT and WLB_J. This behavior is also because the weighing mechanism uses the RPS value. Instead of using the full information about the whole route, just the value of the worst leg in each route is used for route weight evaluation. Figure 4.6 also shows that in this experiment WLB and the rest of the algorithms result in lower throughput since they have greater packet drop rates.

4.4.2 Experiment 2

In this experiment, the network is configured to have some expanding nodes and some converging nodes in a distribution of approximately 50% each. It represents a more realistic network, where nodes move freely in space. In the converging nodes, the path loss between these nodes is simulated to decrease over time and the expanding nodes act oppositely as in experiment 1. This simulation will give the WLB-RTT algorithm an opportunity for improved performance, in case the routes chosen for packet delivery are converging and might not drop with the number of packets as in experiment 1. Again, there is a 50% probability of this happening, since each route has a 50% probability of either converging or expanding.

Figure 4.7: Experiment 2: Total Delay

Figure 4.8: Experiment 2: Total number of Packets Dropped

Figure 4.9: Experiment 2: Delivery Ratio

Figure 4.10: Experiment 2: Delay per Packet

4.4.2.1 Total Delay and Delay per Packet

As explained earlier, the graph in Figure 4.7 shows the time duration required to transfer a 1 Mbyte file from a source to a destination using the different algorithms under study. Similar to experiment 1 results, the total duration to transfer the 1 Mbyte file increases with the number of concurrent connections.

In this experiment, the WLB-RPScum also requires more time to deliver all the packets. This is due to the fact that it delivered more packets than the rest of the algorithms under study. WLBJ in this experiment required more time to deliver the packets. In this case, this algorithm delivers more packets than in experiment 1. The algorithm in this experiment could have selected a converging path where the nodes in this path get closer to each other, thereby reducing the number of packet drops and the packet delivery ratio, as shown in Graphs 4.8 and 4.9.

Since WLB-RPSmin has the capability of detecting better RPS values, it selects better quality paths that have less number of packet drops. This algorithm in this experiment is third in terms of time required. The WLB-RTT and Basic DSR have the shortest delivery time since both have more packet drops.

In Figure 4.10, the delay per packet is represented and shows that WLB-RPScum has the shortest delay per packet. Then comes WLBJ and WLB-RPSmin, which have similar delay per packet values. Basic DSR has the longest delay per packet, since it has the greatest number of packet drops.

Figure 4.11: Experiment 2: Throughput

4.4.2.2 Throughput

The throughput of each algorithm in experiment 2 is shown in Figure 4.11. As is explained in the previous section as a general behavior in this experiment, the performance of WLB-RPScum is the best among the rest of algorithms, since this route weighing mechanism takes into consideration the RPS value of each leg in the route. This allows for more accurate information and better judgment in choosing which route to deliver the packets through. Next comes WLB-J, which has similar results to WLB-RPSmin. The routes selected for WLB-J might have superior RPS values, since the network topology in this experiment have mixed converging and expanding

nodes. WLB-RPSmin performs relatively well because the weighing mechanism uses RPS values. Instead of using the full information about the whole route, just the value of the worst leg in each route is used for route weight evaluation. Basic DSR has the least throughput in this experiment.

4.4.3 Experiment 3

Random packet drop is added to the same network configuration as the one used in Experiment 2. Introducing random packet dropping is more to simulate bit error occurrence, which may also be related to the sensitivity of nodes and other external factors such as weather conditions. The purpose of the experiment is to study the effect of additional packet drop rates on the behavior of each of algorithm. WLB-RPScum exhibits superior performance in this experiment, since it is more prone to choose routes with nodes having better sensitivity and transmission power, lowering the probability of packet drops.

For total delay of file transfer, WLB-RPSmin has the longest time. This might be due to its choosing routes with congested nodes, since this algorithm only uses the RPS parameter to evaluate the route weight and not the RTT value. Next in total delay comes the WLB-RPScum algorithm, which delivers more data packets than the rest of the algorithms. This is shown in Figure 4.12. WLB-J out-performed WLB-RTT in this experiment, having fewer packet drops, as shown in Figure 4.13. The routes chosen for data delivery in both algorithms are different and neither consider the RPS

Figure 4.12: Experiment 3: Total Delay

Figure 4.13: Experiment 3: Total number of Packets Dropped

Figure 4.14: Experiment 3: Delay per Packet

Figure 4.15: Experiment 3: Throughput

Figure 4.16: Experiment 3: Delivery Ratio

value. The difference in packet drop rates may be a result of the routes chosen by the WLB-J algorithm coincidentally having better RPS values than the routes chosen by WLB-RTT. The behavior of each algorithm is reflected consistently in all the performance metrics. It is found that WLB-RPScum is the best in throughput and delivery ratio measurements, as seen in Figures 4.15 and 4.16. Next comes the WLB-RPSmin, since both algorithms consider the received power and sensitivity parameter to weigh the routes selected. Both WLB-J and WLB-RTT exhibit similar results with some minor changes in performance, since they use RTT delay to evaluate the selected route weights.

Figure 4.17: Experiment 4: Throughput

4.4.4 Experiment 4

This experiment studies the behavior of the WLB-RPScum, WLB-RPSmin, and WLB-RTT algorithms under different packet dropping rates. In this test, dropping rates vary from 10% to 90%. Figure 4.17 shows that WLB-RPScum gives the best throughput values among the studied algorithms in all the dropping rates, yielding a 35.8% performance enhancement at a drop rate of 10%, to a 227.8% enhancement at a 90% drop rate. WLB-RPSmin also performs better than WLB-RTT in all drop rates. It is also noted that WLB-RTTs performance decreases by more than 50% when

increasing the drop rate from 10% to 90%. The drop rates are related to node sensitivity, and since WLB-RTT does not consider node sensitivity in its route evaluation mechanism, it suffers from greater packet drops. On the other hand, both WLB-RPS algorithms maintain their performance under different drop rate conditions, due to the fact that the sensitivity parameter significantly contributes to the route weight evaluation and selection. The total number of packets dropped is reflected in Figure 4.18.

Figure 4.18: Experiment 4: Total number of Packets Dropped

4.5 Summary

The four experiment results analyzed above show that WLB-RPScum yields better throughput and lower packet drop rates when compared with the other algorithms under the simulation configuration stated in section 4.2. WLB-RPSmin also performs well in throughput and packet drop rates, but not as well as the WLB-RPScum algorithm. This is because the WLB-RPScum algorithm collects relatively global network information regarding the received power and node sensitivity, instead of the partial information (minimum RPS leg value) that is collected by WLB-RPSmin. WLB-RTT generally has a shorter total delay time since it depends on the RTT parameter to evaluate the route weights. However, the WLB-RTT algorithm sometimes does not achieve the best total transfer delay time, since it might go under severe packet drop rates and packet recovery procedures.

Chapter 5

Conclusion and Future Work

The nature of mobile ad hoc networks makes it very challenging to come up with an efficient routing protocol. This is due to continuous changes in network topology and the limitation of resources at the node level and the link level, specifically the wireless nodes' processing power and memory size. DSR routing protocol is a commonly used routing protocol in MANET. However, some enhancements were added to the basic DSR to overcome some of its limitations. One of the interesting enhancements is that of using multipath source routing and load balancing traffic over the available routes [WSD⁺01]. However, MSR along with the load distribution algorithm do not give much attention to the wireless links specification and the signaling power, which affect the performance of the data transfer and application QoS. Stable connections is required by many applications that are highly affected by packet loss and route maintenance procedure overheads. Choosing the fastest routes for data distribution might mean using shorter or less congested routes, but not strong or long lasting

ones. So, a different route weighing mechanism is needed to overcome the problem resulting from using weak routes.

In this thesis, a new distribution algorithm that adapts to the signal strength power and the wireless specifications to the nodes in the network is presented. The strategy is to assign weights to each route according to the calculated value related to the received signal strength and to the sensitivity of the node. This calculated value is named Received Power and Sensitivity (RPS). The RPS value is calculated at each intermediate hop. Each node receiving a packet will evaluate the RPS value by decrementing the received signal power from the node's sensitivity value.

There are two flavors of this route weight calculation algorithm. The first is to use partial route information, by using the minimum RPS value among the links between hops as the route weight. The second flavor is to use the product of the RPS values of all links connecting the route. The latter algorithm gives more accuracy for weight evaluation, since the RPS value is inversely proportional to the probability of error rate.

The testing of this new algorithm is achieved by implementing the basic DSR protocol algorithm and the MSR extension. The routing protocol is implemented as part of the IP protocol layer of the PIX framework. PIX is an object oriented protocol stack that has the advantage of providing light weighted libraries, well-defined interfaces, and ease of configuration and use. The testing compares the performance

behavior of the basic DSR, the MSR with data distribution according to the RTT value of each route, and the two new algorithms, using MSR with data distribution according to RPS values, called WLB-RPScum and WLB-RPSmin.

The simulation environment is represented as a mobile network. Different network movement behavior is represented to evaluate the algorithms under different conditions. Each experiment executed simulates a network of 50 moving nodes. Each experiment has a different movement behavior that varies from expanding nodes to a combination of expanding and converging nodes. Random packet drop rates are also introduced to some of the experiments conducted. Two other protocol layers are implemented in the PIX framework to simulate mobile network. The first is the application layer, which is a data-streaming layer, and the second is a physical link layer, which uses POSIX message queues to allow inter-process communication on a single machine. The simulation establishes 20 concurrent connections to evaluate the algorithms under additional traffic. A data file is transferred from one source to one destination in each connection. In each experiment the total data transfer time, the number of received packets, and the number of dropped packets are collected for each connection.

The test results, according to the current simulation environment configuration described in Chapter 4, demonstrate that the WLB-RPScum algorithm outperforms the rest of the algorithms under consideration, resulting in reduced packet drop rates

and higher throughput values with a minimum of 35.8% over the WLB-RTT from Experiment 4 and a minimum of 9.45% over WLB-RPSmin from Experiment 4. WLB-RPSmin also offers good performance in throughput and packet drop rates, but still inferior to the WLB-RPScum algorithm. This is expected, since the WLB-RPScum algorithm collects relatively global network information regarding the received power and node sensitivity rather than the partial information (minimum RPS leg value) that is collected by the WLB-RPSmin, which might not give the best packet drop rate probability approximation. The original WLB-RTT algorithm generally reduces the total transfer time since it depends on the RTT parameter to evaluate the route weights. However, WLB-RTT algorithm sometimes does not achieve the best total transfer delay time, since it might experience severe packet drop rates and packet recovery procedures.

Our simulation environment have some limitations that could not be overcome due to time limitation. For future work, we suggest to add some enhancements to improve the simulation to get more accurate results. It would be good to introduce background traffic throughout the network and to have congestion modelling too.

There are many other points that could not be achieved within the time frame of this thesis, that are worth investigating in future work. One of these points is to evaluate the overall performance of multiple applications that exist above the IP Layer to select the load balancing scheme according to the applications' needs. For

example, applications which give more priority to reducing the time of delivery would choose the WLB-RTT algorithm, and applications which emphasize reducing the packet drop rate would choose the WLB-RPS algorithm (Example: CBR UDP).

For future work also we could consider using other factors than the received signal strength and sensitivity to distribute the traffic. One of these factors is the signal-to-noise ratio which affects the bit error rate and packet drops.

One of the important points that might be considered in the future is to apply an optimum data distribution algorithm using a combination of RTT and RPS values as parameters for the evaluation of each route. This idea might yield a better performance result than the results obtained using the heuristic scheme.

Another idea is to use the upper layers (above the IP layer) end-to-end acknowledgement mechanism to update each route's RTT and RPS values. This would reduce the number of control beacons in the probing mechanism to keep the route cache up to date with fresh RTT and RPS values.

Bibliography

- [AASS00] S. Agarwal, A. Ahuja, J. Pal Singh, and R. Shorey. Route-lifetime assessment based routing (RABR) protocol for mobile ad-hoc networks. In *ICC (3)*, pages 1697–1701, 2000.
- [Bar02] M. Barbeau. Protocol implementation framework for linux (PIX). Technical report, School of Computer Science, Carleton University, 2002.
- [BB02] M. Barbeau and F. Bordeleau. A protocol stack development tool using generative programming. In *Proceedings of Generative Programming and Component Engineering (GPCE)*, 2002. Lecture Notes in Computer Science 2487.
- [BCSW98] S. Basagni, I. Chlamtac, V.R. Syrotiuk, and B.A. Woodward. A distance routing effect algorithm for mobility (DREAM). In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM)*, pages 76–84, 1998.
- [BFH99] J. Bellardo, J. Fang, and G. Hamerly. Comparison of ad hoc network routing protocols. November 1999. http://www-cse.ucsd.edu/users/ghamerly/academic/papers/ad_hoc.pdf.
- [BHGV87] R. Binder, S. Huffman, I. Gurantz, and P. Vena. Crosslink architecture for a multiple satellite systems. volume 75, pages 75–82, January 1987.
- [BMJ⁺98] J. Broch, D. A. Maltz, D. B. Johnson, Y. C. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad-hoc network routing

- protocols. In *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pages 85–97, October 1998.
- [Bur99] P. Burden. Routing. Technical report, School of Computing and Information Technology, University of Wolverhampton, 1999. Lecture Notes of Communications Systems CP3340.
- [CBD02] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless Comm. and Mobile Computing (WCMC): Special Issue on Mobile Ad Hoc Networking: Research, Trends, and Applications*, 2(5):483–502, 2002.
- [CM99] S. Corson and J. Macker. Mobile Ad hoc Networking (MANET):routing protocol performance issues and evaluation considerations. RFC 2501, January 1999. <http://ftp.isi.edu/in-notes/rfc2501.txt>.
- [DCY00] S. R. Das, R. Castaeda, and J. Yan. Simulation based performance evaluation of routing protocols for mobile ad hoc networks. *Mobile Networks and Applications*, 5(3):179–189, September 2000.
- [DCYS98] S. Das, R. Castaneda, J. Yan, and R. Sengupta. Comparative performance evaluation of routing protocols for mobile, ad hoc networks. In *Proceedings of IEEE the Seventh International Conference on Computer Communications and Networks (IC3N '98)*, pages 153–161, October 1998.
- [EAB00] E. Ekici, I. Akyildiz, and M. Bender. Datagram routing algorithm for LEO satellite networks. In *Proceedings of INFOCOM 2000, Tel Aviv*, 2000. <http://citeseer.nj.nec.com/ekici00datagram.html>.
- [Ela03] M. E. Elaasar. XSTP: eXtended Satellite Transport Protocol. Master’s thesis, School of Computer Science, Carleton University, January 2003.

- [ELBK03] M. Elaasar, Z. Li, M. Barbeau, and E. Kranakis. The extended satellite transport protocol: Its design and evaluation. In *Proceedings of 17th Annual AIAA/USU Conference on Small Satellites, Logan, Utah*, 2003.
- [GPAA99] Vidyashankar V. Gounder, Ravi Prakash, and Hosame Abu-Amara. Routing in leo-based satellite networks. 1999. <http://citeseer.nj.nec.com/5006.html>.
- [GWFM02] M. Gerharz, C. Waal, M. Frank, and P. Martini. Link stability in mobile wireless ad hoc networks. In *Proceedings of the 27th Annual IEEE Conference on Local Computer Networks (LCN'02)*, pages 30–39, Tampa, FL, November 2002.
- [Hak01] M. Hakkak. Small satellite systems and applications: A review. In *IEEE ICT2001 Preliminary Program*, June 2001. http://www.ict2001.ro/program/conf_program.html.
- [HD98] R. Hinden and S. Deering. Internet Protocol, version 6 IPv6 specification. RFC 2460, December 1998.
- [Hen99] T. Henderson. *Networking over Next-Generation Satellite Systems*. PhD thesis, University of California, Berkeley, 1999. <http://citeseer.nj.nec.com/henderson99networking.html>.
- [HK00] T. R. Henderson and R. H. Katz. On distributed, geographic-based packet routing for LEO satellite networks. In *Proceedings of GLOBE-COM*, volume 2, pages 1119–1123, December 2000.
- [HKR02] H. Huang, A. Khendry, and T.G. Robertazzi. Layernet: a self-organizing protocol for small ad hoc networks. *IEEE Trans. on Aerospace and Electronic Syst.*, 38:378–387, April 2002.
- [Inc] Breezcom Inc. Radio signal propagation. http://didier.quartier-rural.org/implic/ran/sat_wifi/sigprop.pdf.

- [JM96] D. Johnson and D. A. Maltz. Dynamic note routing in ad hoc wireless networks. *Mobile Computing*, Kluwer Academic Publishers, 353:153–181, 1996.
- [JMH03] D. Johnson, D. A. Maltz, and Y. Hu. The dynamic note routing protocol for mobile ad hoc networks. Internet draft, draft-ietf-manet-dsr-09.txt, March 2003. Work in progress, <http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-09.txt>.
- [Joh94] D. Johnson. Routing in ad hoc networks of mobile hosts. In *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications*, December 1994. <http://citeseer.nj.nec.com/johnson94routing.html>.
- [Lan01] D. Lane. *HyperStat Online*. <http://davidmlane.com/hyperstat/index.html>, 2001.
- [Lao99] A. Laouiti. Routing protocols for MANET. October 1999. Presentation slides for INRIA, <http://www-r2.u-strasbg.fr/rge/rge14101999/manet/sld001.htm>.
- [Lee00] S. J. Lee. *Routing and Multicasting Strategies in Wireless Mobile Ad Hoc Networks*. PhD thesis, Computer Science Department, UCLA, 2000. <http://citeseer.nj.nec.com/lee00routing.html>.
- [NCD01] A. Nasipuri, R. Castaeda, and S. R. Das. Performance of multipath routing for on-demand protocols in mobile ad hoc networks. *ACM/Baltzer Mobile Networks and Applications (MONET) Journal*, 6:339–349, 2001.
- [ND99] A. Nasipuri and S. Das. On-demand multipath routing for mobile ad hoc networks. pages 64–70, October 1999.
- [NNB00] H. Labiod N. Nikaein and C. Bonnet. Ddr- distributed dynamic routing algorithm for mobile ad hoc networks. In *Proceedings of the MobiHOC*

2000 : *First Annual Workshop on Mobile Ad Hoc Networking and Computing, USA/Boston*, pages 19 – 27, 2000.

- [Old03] D. R. Oldham. Mobile ad hoc networks for space and surface systems. June 2003. Presentation slides for NASA Space Internet Workshop, http://scp.grc.nasa.gov/siw/presentations/Session_D/D_09_Oldham.pdf.
- [oSN02] National Institute of Standards and Technology (NIST). Nist dsr model appendix. 2002. http://w3.antd.nist.gov/wctg/prd_dsrfiles.html.
- [Pos81] J. Postel. IP: Internet Protocol. RFC 791, September 1981.
- [RP94] J. Reynolds and J. Postel. Assigned numbers. RFC 1700, October 1994.
- [RT99] E.M. Royer and C-K Toh. A review of current routing protocols for ad-hoc mobile wireless networks. *IEEE Personal Communications*, April 1999. <http://citeseer.nj.nec.com/royer99review.html>.
- [SSC96] W. Sun, M. Sweeting, and A. Curiel. LEO satellite constellation for regional communication. Technical report, Center for Satellite Engineering Research, University of Surrey, UK, 1996.
- [WCA⁺01] L. Wood, A. Clerget, I. Andrikopoulos, G. Pavlou, and W. Dabbous. IP routing issues in satellite constellation networks. *Refereed journal paper, Special issue on IP, International Journal of Satellite Communications (IJSC)*, John Wiley and Sons, 19(1):69–92, January/February 2001.
- [WLTS99] S.-L. Wu, T.-K. Lin, Y.-C. Tseng, and J.-P. Sheu. Route optimization on wireless mobile ad-hoc networks. In *The 5th Mobile Computing Workshop, Taiwan*, pages 143–150, 1999.
- [WSD⁺01] L. Wang, Y. Shu, M. Dong, L. Zhang, and O. W. Yang. Adaptive multipath note routing in wireless ad hoc networks. *IEEE International Conference on Communications*, June 2001.

- [Zha02] S. Zhang. Channel management, network representation and event handling of a protocol implementation framework for Linux using generative programming. Master's thesis, School of Computer Science, Carleton University, June 2002.

- [ZZS⁺02] L. Zhang, Z. Zhao, Y. Shu, L. Wang, and O. W.W. Yang. Load balancing of multipath note routing in ad hoc networks. In *Proceedings of IEEE International Conference on Communications (ICC 2002)*, April 2002.

Appendix A

Measurement Data

The following tables show the complete results of the simulation tests for the four different experiments conducted. For the total transfer delay, packet drop, throughput, delivery ratio and delay per packet measurements, the standard deviation and the 95% confidence interval are calculated. The standard deviation (SD) σ is calculated by the following formula ¹[Lan01]:

$$\sigma = \sqrt{\frac{\sum_{k=1}^n (X_k - \bar{X})^2}{n - 1}}$$

where n is the number of replications, $\bar{X}$ is the mean value and X_k is the k th value.

The confidence interval CI is calculated by the following formula ²[Lan01]:

$$CI = \bar{X} \pm Z \times \frac{\sigma}{\sqrt{n}}$$

where n is the number of replications, $\bar{X}$ is the mean value, σ is the standard deviation and Z is a statistical value corresponding to the desired confidence interval.

¹This formula calculates the unbiased value of the standard deviation.

²This formula is customized for a 95% confidence interval by setting Z to 1.96.

Exp 1			Packet Drop		Delivery Ratio		Throughput		Delay-per-Packet		Total Delay	
Basic DSR	Value		1015.932833		7.811497147		0.163002559		7.811497147		915.088463	
	STD		0.329408991		0.017166229		0.00152249		0.017166229		7.99934615	
	95% CI	Max	1016.017202		7.819020583		7.819020583		0.163670256		918.594462	
WLB-RTT	Value	Min	1015.789464		7.803973711		7.803973711		0.162334859		911.582463	
	STD		867.5		3.90864164		0.234109875		3.90864164		922.334598	
	95% CI	Max	15.26433752		0.174042422		0.017695494		0.174042422		8.00372644	
WLB-J	Value	Min	874.1898909		3.982919094		3.982919094		0.331865268		925.842386	
	STD		860.8101091		3.830364165		3.830364165		0.316354483		918.82681	
	95% CI	Max	761		1.000726275		1.270835108		1.000726275		916.185292	
WLB-RPScum	Value		33.8538401		3.23362E-05		0.011103295		3.23362E-05		7.8363719	
	STD		775.8392909		1.000740447		1.000740447		1.275701342		919.619734	
	95% CI	Max	746.1607091		1.000712103		1.000712103		1.265968874		912.750851	
WLB-RPScum	Value	Min	355.3330591		1.847106161		0.67673148		1.847106161		931.001775	
	STD		287.6174586		0.076926973		0.030574711		0.076926973		6.63206035	
	95% CI	Max	472.6215817		1.880820894		1.880820894		0.692131438		933.908404	
WLB-RPScum	Value	Min	238.0445365		1.813391429		1.813391429		0.665331522		928.035147	
	STD		747.0277778		1.00023564		1.234628459		1.00023564		951.212399	
	95% CI	Max	12.70639658		1.21996E-05		0.01091983		1.21996E-05		7.64280809	
WLB-RPScum	Value	Min	252.6318588		1.000240967		1.000240967		1.223019842		954.562047	
	STD		241.4238968		1.000230293		1.000230293		1.220237074		947.862751	
	95% CI	Max										

Table A.1: Experiment 1 Results

Exp 2						
Basic DSR	Value	Packet Drop		Delivery Ratio	Throughput	Delay-per-Packet
	STD	707.95		2.231750537	0.655589794	1.756582579
	95% CI	Max	Min	0.635972184	0.318393573	0.502966124
		716.356832		2.510477636	0.79513193	1.977118079
		899.543168		1.9530023438	0.516047658	1.536247079
WLB-RTT	Value	531.7456267		1.732060157	0.755122386	1.36787465
	STD	5.274932405		0.250687961	0.174516158	0.169541022
	95% CI	Max	Min	1.841926817	0.831651192	1.454932918
		534.0575077		1.622193497	0.678593381	1.280616384
		529.4338256		1.000337525	1.228436854	0.813759106
WLB-J	Value	353.8		2.015735-05	0.009863711	0.006422671
	STD	21.12269919		1.000346359	1.233261816	0.816574053
	95% CI	Max	Min	1.00032869	1.224615833	0.810944159
		363.0574311		1.000372119	30.63198042	0.30130344
		344.5425689		1.83427E-05	45.99022385	0.351825412
WLB-RPSmin	Value	390.0496111		1.000380158	50.78808472	0.459685419
	STD	19.21953127		1.00036406	10.41507612	0.142731462
	95% CI	Max	Min	1.000103995	1.1578724	0.863713993
		393.4729421		1.82593E-05	0.010069571	0.007367834
		381.6352801		1.0001119	1.162285584	0.865942989
WLB-RPScum	Value	108.9330556		1.000095895	1.153479216	0.860484797
	STD	19.14205569		1.000095895	1.153479216	0.860484797
	95% CI	Max	Min	1.000095895	1.153479216	0.860484797
		117.3274445		1.000095895	1.153479216	0.860484797
		100.5436668		1.000095895	1.153479216	0.860484797
						Total Delay
						916.519147
						5.26833144
						918.828095
						914.210198
						920.098521
						7.77327992
						923.905311
						916.691731
						947.7772
						7.46503483
						951.048896
						944.505504
						965.407574
						8.96806386
						969.338001
						961.477147
						1096.1941
						8.57498414
						1009.95276
						1002.43595

Table A.2: Experiment 2 Results

Exp 3						
Basic DSR	Value	Packet Drop	Delivery Ratio	Throughput	Delay-per-Packet	Total Delay
	STD	720 3225556	2 622271765	0 484060623	2 064088939	917 126682
	95% CI	14 86037334	0 039962611	0 015614227	0 067571492	3 35263672
	Min	726 8356205	2 661629706	0 49180386	2 093702451	918 598039
WLB-RTT	Value	713 8094906	2 582843625	0 478117386	2 034474426	915 659324
	STD	342 9333333	1 000327154	1 255533238	0 790207439	920 35685
	95% CI	5 225443824	4 93687E-06	0 011218324	0 006875838	8 00871577
	Min	345 2234851	1 000329339	1 270499883	0 793220896	923 866624
WLB-J	Value	340 6431816	1 000324968	1 260686589	0 76719396	916 846876
	STD	368 2609444	1 000370412	1 253340179	0 797924503	929 3044
	95% CI	9 438952716	9 00831E-06	0 010922103	0 006827489	7 94592168
	Min	392 3977479	1 00037436	1 258127001	0 800916782	932 788854
WLB-RPSmin	Value	384 12414	1 000366464	1 248553366	0 794932234	925 821946
	STD	120 2325568	1 000114676	1 206618221	0 82877791	965 48475
	95% CI	3 245991342	3 03633E-06	0 010319058	0 006962739	8 11125709
	Min	121 655174	1 000116033	1 211200747	0 831829465	969 039665
WLB-RPScum	Value	118 8093371	1 000113319	1 203155695	0 825726355	961 929835
	STD	11 04733333	1 000010526	1 223940697	0 817071989	951 9469
	95% CI	0 966048642	9 40389E-07	0 008756217	0 005725427	6 69880763
	Min	11 4794829	1 000010948	1 227779155	0 819581248	954 870112
WLB-RPScum	Value	10 61517837	1 000010124	1 220102239	0 81456269	949 023688
	STD					
	95% CI					
	Min					

Table A.3: Experiment 3 Results

Exp. #		Packet Drop Rate = 50%				Packet Drop Rate = 50%			
		Packet Drop	Total Time (sec)	Throughput (bytes/sec)	Packet Drop	Total Time (sec)	Throughput (bytes/sec)	Packet Drop	Total Time (sec)
WLB-RTT	Value	821.977222	9.030695	318.453424	545.132444	11.83383	471.114356	471.114356	
	STD	18.657253	7.7691187	19.4069806	64.4619907	53.4744387	44.8015275	44.8015275	
	95%								
	Max	830.154644	97.811345	326.338824	583.3641646	101.77441	490.749265	490.749265	
	Min	813.800401	186.23295	309.949132	508.001003	1154.80789	451.479097	451.479097	
WLB-RP Smin	Value	199.930287	945.407	917.953453	13.133111	948.167	969.656453	969.656453	
	STD	12.6348378	8.3365897	17.8375975	9.41841819	7.7210134	14.4875248	14.4875248	
	95%								
	Max	265.427003	950.92371	925.789874	157.757567	921.230823	987.625765	987.625765	
	Min	184.3042431	942.14323	910.137011	138.009205	944.763117	954.399205	954.399205	
WLB-RP Ssum	Value	75.935	6.030475	104.102658	53.30552558	934.263173	1059.663197	1059.663197	
	STD	1.98372079	7.9044129	10.8271031	1.44694476	8.97952482	7.70173024	7.70173024	
	95%								
	Max	79.7618227	942.39329	1048.74934	58.7332178	947.620856	1063.344101	1063.344101	
	Min	75.1061673	939.446301	1039.45692	46.7338383	940.749698	1056.569293	1056.569293	
Exp. 4									
WLB-RTT	Value	824.618444	236.5219	559.190636	476.819	1252.96445	537.524318	537.524318	
	STD	76.2191611	21.63507	28.70043827	12.4710188	64.5793827	11.3074521	11.3074521	
	95%								
	Max	476.3093984	1240.2014	581.484332	422.2641394	1257.12953	542.620242	542.620242	
	Min	303.2379804	1226.81246	541.1880256	411.3533208	1243.23637	537.5422211	537.5422211	
WLB-RP Smin	Value	142.8589528	938.8831	983.3871689	11.5371779	944.7862	1000.63370	1000.63370	
	STD	27.8544305	13.5339420	26.1422771	19.8976731	10.4173035	19.5444024	19.5444024	
	95%								
	Max	154.9798478	947.505323	984.1826577	116.3819871	952.345181	1089.18843	1089.18843	
	Min	130.7341633	939.730371	979.934556	106.8630584	943.214016	962.057059	962.057059	
WLB-RP Ssum	Value	13.70261827	546.21335	1089.432336	13.6	948.30325	1085.204375	1085.204375	
	STD	3.97619235	3.45101714	9.31223235	4.17592216	8.9448447	6.85233504	6.85233504	
	95%								
	Max	21.47676591	349.621672	1093.8573	15.4267193	949.829371	1096.004756	1096.004756	
	Min	17.9833742	347.520328	1085.14106	11.7712808	942.777929	1091.32518	1091.32518	
Exp. 4									
WLB-RTT	Value	343.971222	870.1515	507.477616					
	STD	4.7607108	8.90631021	8.15076965					
	95%								
	Max	347.653758	923.35917	508.82011					
	Min	341.4907109	916.64153	799.8111022					
WLB-RP Smin	Value	153.232858	265.48475	974.042314					
	STD	3.24391347	5.1115709	6.64417303					
	95%								
	Max	121.655174	959.039655	977.277605					
	Min	148.909271	951.92635	916.176943					
WLB-RP Ssum	Value	11.0473333	261.9469	1091.11754					
	STD	0.2804841	6.65409783	8.50655207					
	95%								
	Max	15.4744829	954.87012	1094.84845					
	Min	10.6151787	949.023668	1087.38866					

Table A.4: Experiment 4 Results