

**DESIGN, IMPLEMENTATION, AND SIMULATION OF WIRELESS
MESH NETWORKS ROUTING PROTOCOLS WITH LINK QUALITY
AND INTERFERENCE AWARENESS.**

MICHEL MBOUGNI

21248435

**SUBMITTED IN FULFILLMENT OF THE REQUIREMENTS FOR THE
DEGREE OF MASTER OF SCIENCE IN COMPUTER SCIENCE**

**DEPARTMENT OF COMPUTER SCIENCE
SCHOOL OF MATHEMATICAL AND PHYSICAL SCIENCE
FACULTY OF AGRICULTURE, SCIENCE AND TECHNOLOGY
NORTH WEST UNIVERSITY-MAFIKENG CAMPUS**

SUPERVISOR: PROF O.O EKABUA

NOVEMBER 2012

LIBRARY
MAFIKENG CAMPUS
CALL NO.:
2021 -02- 1 0
ACC.NO.:

Declaration

I declare that the research thesis on the **Design, Implementation and Simulation of Wireless Mesh Networks Routing Protocols with Link Quality and Interference Awareness** is my work and has never been presented for the award of any Degree at any University. All information used has been duly acknowledged both in text and references.

Signature:.....

Date:.....

Michel Mbougni

Approval

Signature:.....

Date:.....

Supervisor: **Prof O.O Ekabua**

Department of Computer Science

Faculty of Agriculture Science and Technology

North West University-Mafikeng Campus

Dedication

I dedicate this project to my late father, Mr Michel Mbougni, to the late Mr Teutu Lazare and to my mother, Mrs Micheline Tchongang Mbougni. They helped me to overcome the challenges of life.

Acknowledgements

I would like to thank the Almighty for his guidance and help during difficult times.

I am particularly grateful to my supervisor, Prof O. O. Ekabua, for his assistance and his comments. His positive inputs helped me to complete this research.

My special thanks go to family members, Christelle, Olivier, Anita and Eric for their moral, spiritual and financial support throughout my studies. My deepest gratitude goes to my beloved mother, Mrs Micheline Tchongang Mbougne for everything she has done for me; she has always pushed me to do better.

I express a lot of gratitude to the former Director of the school of Mathematical and Physical Sciences and now the acting Dean of the Faculty of Agriculture, Science and Technology, Prof Eno Ebenso for always encouraging me, even during difficult times.

I would like to thank Mr Boyce Sigweni for helping me to understand networks modelling concepts and for assisting me on how to use OPNET 14.0. Thanks to Mr Francis Lugayizi and Mr Mmoloki Mangwala for their technical advices. To my colleagues, Nosipho, Thuso, Nnena, Ifeoma and Peter, thank you for your constructive discussions; they helped in making this research better.

I would also like to thank Dr Ateba Collins, Mrs Ateba Tshepiso, Mr Ngomba Mbua, Mr Hamuli Erick, Mrs Hamuli Ntombikhayise Precious, the Noubissie's family and the Khuzwo's family for their moral support throughout my stay in South Africa.

I would like to thank my friends, Mr Bechuke Andre, Mr Ateba Benedict, Mr Nkamta Paul, Ms Eposi Mbua, Ms Christelle Meniago, Mrs Mah Sah Gisele, Dr Nyirenda Matthew, Dr Noutchuomie Clovis and Mrs Christelle Noutchuomie for always lifting me up when I needed it the most.

Special thanks goes to the TELKOM Center of Excellence for making the resources needed to complete this research available.

Abstract

Routing metrics are vital when determining the performance of Wireless Mesh Networks (WMNs). During the implementation of routing protocols, routing metrics are assigned to different paths and the calculation of the best paths is a function for predicting the best routing path. This study proposes enhancements to the Multi-Hop Effective Bandwidth (MHEB) routing metric commonly used in WMNs. The proposed routing metrics are called improved Multi-Hop Effective Bandwidth (*i*MHEB) and extended improved Multi-Hop Effective Bandwidth (*Xi*MHEB). The *i*MHEB routing takes into consideration the wireless link asymmetry to improve the MHEB routing metric. The *Xi*MHEB, routing metric takes into account the average time spent for single data transmission to enhance the *i*MHEB routing metric. Our proposed metrics, *i*MHEB and *Xi*MHEB are respectively implemented into Ad hoc On Demand Distance Vector (AODV) and Dynamic Source Routing (DSR) routing protocols resulting into novel routing protocols. The simulation results based on OPNET modeler 14 show that the novel routing protocols perform better than traditional routing protocols in terms of throughput and delay. The novel routing protocols simultaneously achieve link quality and interference awareness.

Table of Contents

Declaration ii

Dedication..... iii

Acknowledgements iv

Abstract v

Table of Contents..... vi

Table of Figures ix

List of Tables..... xi

List of Acronyms and Abbreviations..... xii

CHAPTER 1..... 1

INTRODUCTION 1

1.1 Introduction and Background 1

1.2 Problem Statement..... 4

1.3 Research Rationale 5

1.4 Research Questions.....5

1.5 Research Goal..... 6

1.6 Research Objectives 6

1.7 Research Methodology 6

1.8 Research Contribution 7

1.9 Research Limitation..... 7

1.10 Included Publications 7

1.11 Chapter Summary 7

CHAPTER 2..... 9

LITERATURE SURVEY 9

2.1 Chapter Overview 9

2.2 Routing Metrics Overview..... 9

2.2.1 Routing Metrics Requirements.....9

2.2.2 Routing Metrics for Wireless Mesh Networks.....12

2.2.2.1 Expected Transmission Count (ETX). 12

Table of Contents

Declaration	ii
Dedication.....	iii
Acknowledgements	iv
Abstract	v
Table of Contents.....	vi
Table of Figures	ix
List of Tables	xi
List of Acronyms and Abbreviations	xii
CHAPTER 1.....	1
INTRODUCTION	1
1.1 Introduction and Background	1
1.2 Problem Statement.....	4
1.3 Research Rationale	5
1.4 Research Questions.....	5
1.5 Research Goal.....	6
1.6 Research Objectives	6
1.7 Research Methodology	6
1.8 Research Contribution	7
1.9 Research Limitation.....	7
1.10 Included Publications	7
1.11 Chapter Summary	7
CHAPTER 2.....	9
LITERATURE SURVEY	9
2.1 Chapter Overview	9
2.2 Routing Metrics Overview.....	9
2.2.1 Routing Metrics Requirements.....	9
2.2.2 Routing Metrics for Wireless Mesh Networks.....	12
2.2.2.1 Expected Transmission Count (ETX).	12

2.2.2.2	Expected Transmission Time (ETT)	13
2.2.2.3	Weighted Cumulative Expected Transmission Time (WCETT).....	14
2.2.2.4	Expected Link Performance	15
2.2.2.5	Multi-Hop Effective Bandwidth (MHEB)	16
2.3	Routing Protocols	17
2.3.1	Reactive Routing Protocols.....	17
2.3.1.1	Ad hoc On Distance Vector (AODV).....	18
2.3.1.2	Dynamic Source Routing (DSR)	23
2.3.1.3	Temporally-Ordered Routing Algorithm (TORA).....	25
2.3.2	Proactive routing protocols.....	29
2.3.2.1	Optimized Link State Routing (OLSR).....	29
2.4	Chapter Summary	35
CHAPTER 3.....	36	
ROUTING METRICS DESIGN, SIMULATION AND EXPERIMENTAL SETUP		
.....		36
3.1	Chapter Overview	36
3.2	Routing Metrics Design	36
3.2.1	Improved Multi-Hop Effective Bandwidth.....	36
3.2.2	Extended improved Multi-Hop Effective Bandwidth.....	37
3.3	Routing Metrics Implementation	38
3.3.1	Improved Multi Hop Effective Bandwidth Implementation.....	39
3.3.2	Extended improved Multi Hop Effective Bandwidth Implementation.....	40
3.4	Simulation Software Tool.....	42
3.5	Simulation Setup.....	43
3.5.1	Performance Metrics.....	43
3.5.2	Random Topology.....	45
3.5.3	Grid Topology.....	47
3.6	Chapter Summary	49
CHAPTER 4.....	50	
RESULTS AND DISCUSSION.....		50
4.1	Chapter Overview	50
4.2	Simulation Results for <i>i</i> MHEB.....	50

4.2.1 Random Topology.....	50
4.2.1.1 Throughput Comparisons Under Light and Heavy HTTP loads.....	50
4.2.1.2 Delay Comparisons Under Light and Heavy HTTP Loads.....	52
4.2.1.3 Routing Overhead Comparisons Under Light and Heavy HTTP Loads	54
4.2.2 Grid Topology.....	55
4.2.2.1 Throughput Comparisons Under Light and Heavy HTTP Loads.....	55
4.2.2.2 Delay Comparisons Under Light and Heavy HTTP Loads.....	57
4.2.2.3 Routing Overhead Comparisons Under Light and Heavy HTTP Loads	58
4.3 Chapter Summary	60
CHAPTER 5.....	61
SUMMARY, CONCLUSIONS AND FUTURE WORK	61
5.1 Summary	61
5.2 Concluding Remarks.....	63
5.3 Future Work.....	63
References.....	65

Table of Figures

Figure 1.1: Clients WMN Architecture [3].....	2
Figure 1.2: Infrastructure WMN Architecture [3].....	2
Figure 1.3: Hybrid WMNs architecture [3].....	3
Figure 2.1: Inter-flow interference between two nodes.....	11
Figure 2.2: Inter-flow interference between two nodes.....	11
Figure 2.3: RREQ message format [26].....	19
Figure 2.4: RREP message format [26].....	20
Figure 2.5: RRER message format [26].....	20
Figure 2.6: RREQ message broadcasted by the source node.....	22
Figure 2.7: RREP message unicasted to the next hop towards the source node.....	22
Figure 2.8: DSR header format [29].....	23
Figure 2.9: Query (QRY) Packet Format [31].....	26
Figure 2.10: UDP packet format [31].....	26
Figure 2.11: CLR packet format [31].....	27
Figure 2.12: OPT packet [31].....	27
Figure 2.13: OLSR packet format [33].....	30
Figure 2.14: Normal broadcast (left) and optimized broadcast using MPRs (right)...	32
Figure 2.15: TC message format [33].....	33
Figure 3.1: Flow chart of the AODV- <i>i</i> MHEB.....	39
Figure 3.2: Flowchart of the DSR- <i>Xi</i> MHEB.....	41
Figure 3.3: OPNET workflow.....	43
Figure 3.4: Random topology network model.....	45
Figure 3.5: Simulation node model.....	46
Figure 3.6: Process model of <i>i</i> MHEB in random topology.....	47

Figure 3.7: Grid Topology network model.....	48
Figure 3.8: Process model for the grid topology.....	49
Figure 4.1: Throughput under light HTTP traffic.....	51
Figure 4.2: Throughput under heavy HTTP traffic.....	52
Figure 4.3: Delay under light HTTP traffic.....	53
Figure 4.4: Delay under heavy HTTP traffic.....	53
Figure 4.5: Routing overhead under light load HTTP traffic.....	54
Figure 4.6: Routing overhead under heavy load HTTP traffic.....	55
Figure 4.7: Throughput under light http load.....	56
Figure 4.8: Throughput under heavy HTTP load.....	56
Figure 4.9: Delay under light load.....	57
Figure 4.10: Delay under heavy load.....	58
Figure 4.11: Routing Overhead under low load.....	59
Figure 4.12: Routing Overhead under heavy load.....	59

List of Tables

Table 2.1: summary and comparison of the routing protocols studied earlier [34]....	34
Table 3.1: Routing metrics used for performance evaluation.....	44

List of Acronyms and Abbreviations

ABIRF	Achievable Bandwidth under Intra-Flow Interference
ABITF	Achievable Bandwidth under Inter-Flow Interference
AODV	Ad-hoc On Demand Distance Vector
CLR	Clear Packet
DSN	Destination Sequence Number
DSR	Dynamic Source Routing
EPL	Expected Link Performance
ETT	Expected Transmission Time
ETX	Expected Transmission Count
FSM	Finite State Machine
/ABITF	Improved Achievable Bandwidth under Inter-Flow Interference
IEEE	Institute of Electrical and Electronical Engineers
IETF	Internet Engineering Task Force
/MHEB	Improved Multi-Hop Effective Bandwidth
IP	Internet Protocol
MANETs	Mobile Ad hoc Networks
MHEB	Multi-Hop Effective Bandwidth
MPR	Multi Point Relay
NS-2	Network Simulator version 2
OLSR	Optimized Link State Routing
OPNET	Optimized Network Engineering Tools
OPT	Optimization Packet

QRY	Query Packet
RERR	Route Error
RREP	Route Reply
RREQ	Route Request
TC	Topology Control
UDP	User Datagram Protocol
UPD	Update Packet
WCETT	Weighted Cumulative Expected Transmission Time
WMN	Wireless Mesh Network
XiABITF	Extended Improved Achievable Bandwidth under Inter-Flow Interference.
XiMHEB	Extended Improved Multi-Hop Effective Bandwidth.

CHAPTER 1

INTRODUCTION

1.1 Introduction and Background

Wireless mesh networks (WMNs) have become very popular in the world of wireless networks especially for their low cost and applicability [1]. This is due mainly to their autonomous capabilities (self-organizing, self-configuring and self-healing) [2, 3]. This attribute brings several advantages to WMNs such as low up-front cost, easy network maintenance, robustness, and reliable service coverage. Classical WMNs consist of mesh routers and mesh clients. Mesh routers generally have minimal mobility in a mesh network and form the backbone of WMNs. Mesh routers are usually equipped with a multiple radio interface to improve the capacity of the network. The mesh clients could be either stationary or mobile and can form self-organized ad-hoc networks which can access services by relaying requests to wireless backbone networks. They can be devices such as laptop, Personal Digital Assistants, IP (Internet Protocol) phone and etc. Based on the functionality of the nodes, there exists three types of WMN architecture [3]:

- (i) Client WMNs: In this type of architecture, mesh routers are not required since the mesh clients form the real network. Mesh clients have self-organizing, self-configuring capabilities as well as routing capabilities. Figure 1.1 shows a clients WMNs architecture;
- (ii) Infrastructure/Backbone WMNs: This architecture consists of interconnection mesh routers forming a backbone for clients connected to them. Mesh routers with gateway capabilities can then be connected to the internet. An Infrastructure/Backbone WMNs is shown in Figure 1.2; and
- (iii) Hybrid WMNs: This type of architecture combines both clients WMNs and Infrastructure/ Backbone WMNs. The mesh clients communicate with each

other and can also access the backbone directly via the routers. Figure 1.3 shows an example of Hybrid WMNs.

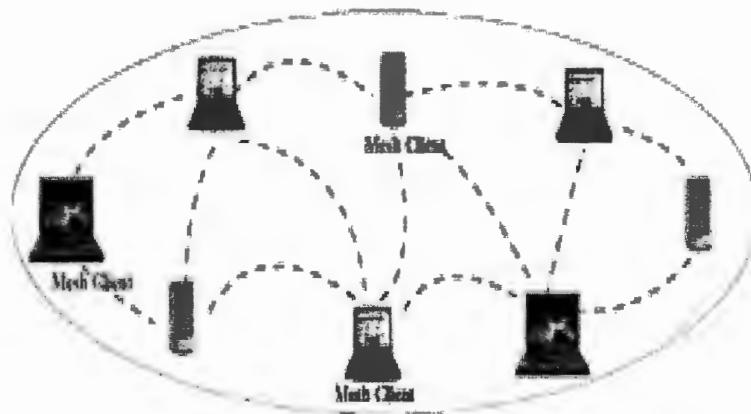


Figure 1.1: Clients WMN Architecture [3]

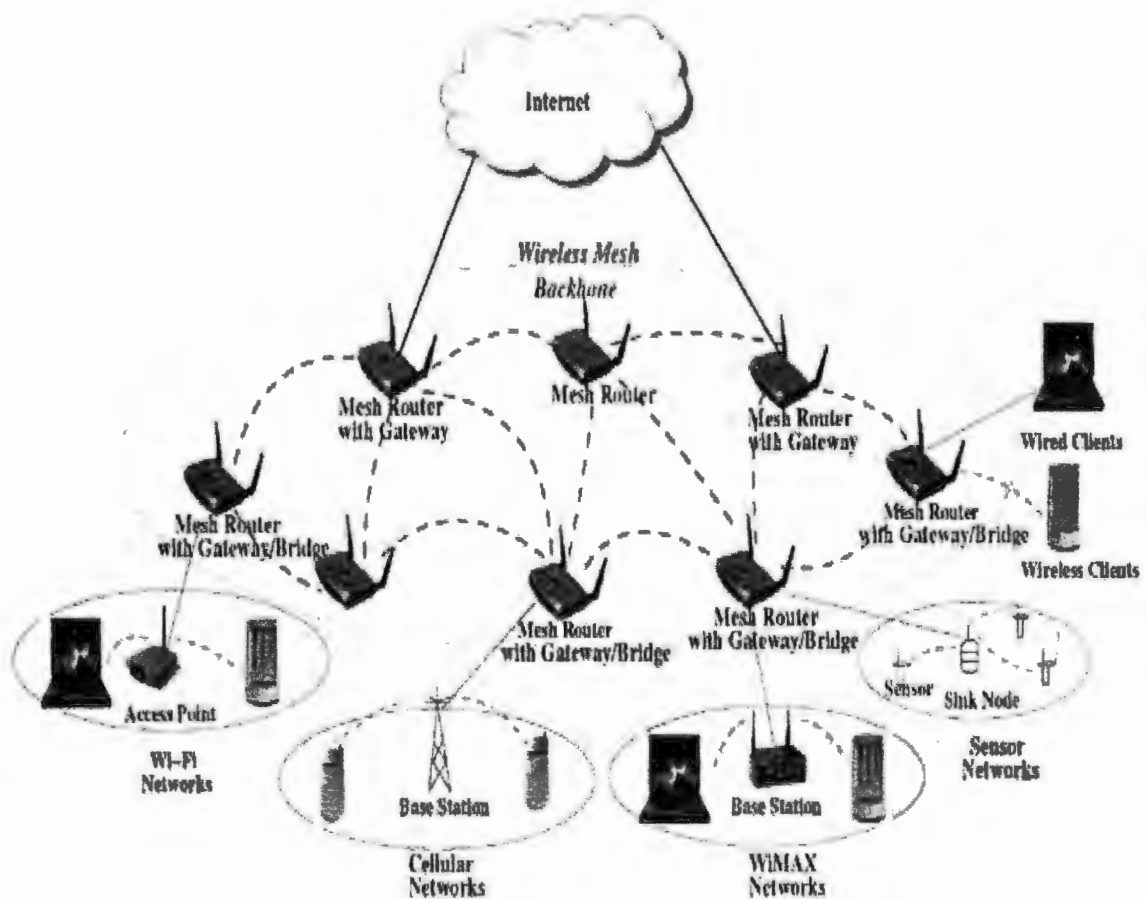


Figure 1.2: Infrastructure WMN Architecture [3]

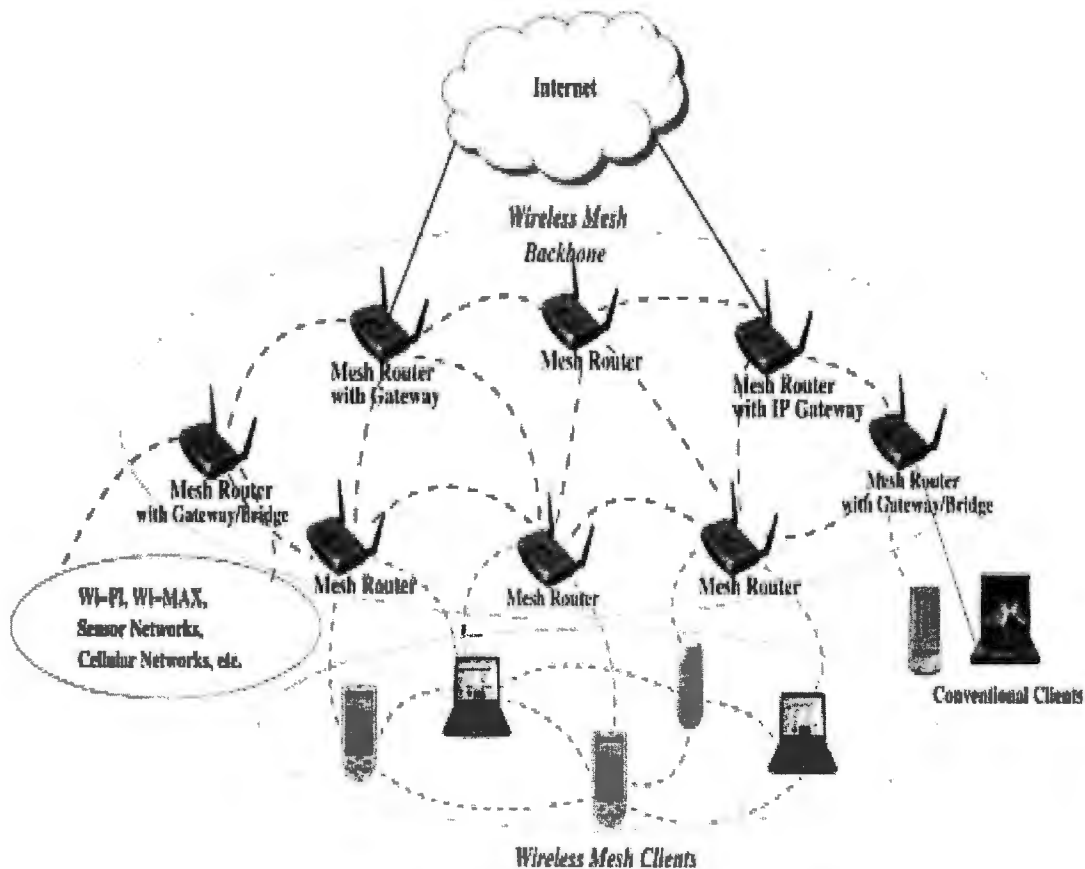


Figure 1.3: Hybrid WMNs architecture [3]

WMNs have the following characteristics [3]:

- (i) Multi-hop capabilities: to extend the coverage of traditional wireless networks without compromising the capacity and quality of the link;
- (ii) Self-forming, self-healing and self-organization capabilities: to make it suitable for tactical networks;
- (iii) Support for multiple type of networks access: WMNs support both access to internet and peer to peer communications;
- (iv) Power-consumption constraints on the mesh clients: in WMNs, mesh routers do not have strict constraints on power consumption. But mesh clients requires power efficient protocols; and
- (v) Capabilities to be compatible and inter-operable with other wireless networks: WMNs built IEEE (Institute of Electrical and Electronical

Engineers) 802.11 technologies must be compatible with any other IEEE 802.11 standards in supporting both mesh and WIFI clients. WMNs should also be inter-operable with other wireless networks e.g., cellular networks, WIMAX and Zig-Bee networks.

Research and development of WMNs is motivated by their applications in real life and some of these are in:

- (i) enterprise networking;
- (ii) metropolitan area networking;
- (iii) broadband networking;
- (iv) transportation systems; and
- (v) health and medical systems.

Internet Protocol (IP) has been approved as the network layer for many wireless networks including WMNs [3]. So routing protocols for wireless networks can be applied to WMNs.

1.2 Problem Statement

Generally, routing protocols are very important in the field of networks, and particularly in wireless networks. Ramadhan and Davis [4] stated that: *"One of the important issues for wireless networks is the choice of the routing protocol as it plays an important role in managing the formation, configuration, and maintenance of the topology of the network."* The routing protocol's strengths and weaknesses have a direct impact in the WMN's characteristics [5, 6] and WMNs are analogous to Ad-hoc networks in terms of wireless multi-hop communication [5, 7, 8]. As a result, routing protocols for Ad-hoc networks can be applied to WMNs. Initially, routing protocols used in WMNs were based on Ad-hoc networks routing protocols e.g. Ad-hoc On demand Distance Vector (AODV) and Dynamic Source Routing (DSR). These routing protocols select the best path based on the minimum hop count metric. However, due to the WMN's specific characteristics, it has been shown that the hop count metric alone does not produce the expected results in WMNs as the hop count metric does not take into consideration the link quality and the interferences. Therefore, new metrics for

WMNs are critically needed [1]. Some routing metrics such as Expected Transmission Count (ETX) and Expected Transmission Time (ETT) have been proposed in an attempt to take into consideration the link quality during the best path selection. However, these metrics still do not take into considerations the intra-flow and inter-flow interferences. More so, modeling link quality, capacity and the effect of interference can be an extremely difficult task for WMNs [9]. Consequently, based on the specific characteristics and architecture of WMNs, more realistic routing metrics and routing protocols must be proposed and used to select routes in WMNs [10] and that is why Catalan-Cid et al. [11] argued that: "*...a comprehensive solution is still needed.* " for routing metrics in WMNs.

1.3 Research Rationale

WMNs are analogous to Ad-hoc networks, therefore routing protocols for Ad-hoc networks can be implemented in WMNs. Unfortunately, these traditional routing performs poorly in WMNs because they use minimum count as routing metric. Minimum hop count metric does not consider the fact that Mesh routers are equipped with multi radio interfaces which could produce interferences during the operation of the WMN. Furthermore, minimum hop count does not take into consideration the link quality during the best route selection from source node to destination node. Reported in this thesis is a solution to enhance traditional routing protocols to efficiently perform in WMNs by designing a routing metric which considers link quality and interference during best path selection.

1.4 Research Questions

The research conducted and reported in this thesis provides answers to the following questions:

- (i) Can an efficient and simple routing metrics be developed with an attempt to accurately capture link quality and interferences in WMNs?
- (ii) Can the developed routing metrics be incorporated into existing routing protocols resulting to new routing protocols?

- (iii) Do the novel protocols perform better than existing routing protocols in WMNs?

1.5 Research Goal

The main goal of this research is to design, implement and simulate a novel Wireless Mesh Routing protocol with link quality and interference awareness, for optimal performance improvement of typical routing protocols used in WMNs.

1.6 Research Objectives

In order to achieve the goal for this research, the following objectives were formulated:

- (i) To propose routing metrics which take into consideration the link quality and the interference in WMNs;
- (ii) To implement the proposed metrics in existing routing protocols, resulting to novel protocols; and
- (iii) To compare the performance of the novel protocols with existing reactive routing protocols.

1.7 Research Methodology

In order to achieve the main goal of this research through the objectives specified, the following research methodology was employed:

- (i) Literature survey of the existing routing metrics and routing protocols in WMNs;
- (ii) Design of routing metrics which are link quality and interference aware.
- (iii) Implementation of the designed routing metric in existing routing protocols resulting to novel protocols.
- (iv) Modeling and simulation of WMNs using a discrete event simulator such as Optimized Network Engineering Tools (OPNET) modeler version 14.0; and
- (v) Evaluating the comparative advantages of the proposed routing protocol against a traditional routing protocol to ascertain the best performance metrics.

1.8 Research Contribution

The main contribution of the research reported in this thesis to the research community, academia and network engineers is the development and implementation of two new routing protocols called AODV-*i*MHEB and DSR-*Xi*MHEB which accurately capture link quality and interference in multi-interface, multi-radio and multi-rate hybrid WMNs.

1.9 Research Limitation

The Research work reported in this thesis concentrates mainly on link quality and interference modeling and measurement in WMNs. Others WMNs problems such as power control, topology control, link scheduling, and channels assignment are out of the scope for this research. This research is also limited to a simulation implementation; hence no physical or test bed was performed.

1.10 Included Publications

Part of the research deliverable reported in this thesis has been published in a conference proceedings and, also another submitted and under review by an accredited journal. Therefore, we have:

- (i) A paper presented and published in the proceedings of the Southern Africa Telecommunication Networks and Application Conference (SATNAC 2012. 2-5 September, 2012. Frankfort, George, Western Cape). The Title of the paper is: *"Towards Improved Multi Hop Effective Bandwidth in Wireless Mesh Networks."* Pages 151-156.
- (ii) A paper submitted and currently under review in the Turkish Journal of Electrical Engineering and Computer Sciences. The title of the paper is : *"Implementation of an Efficient Link Quality and Interference Aware Dynamic Source Routing Metric for Wireless Mesh Networks."*

1.11 Chapter Summary

This chapter presented an overview and general background for this thesis. The remainder of this thesis is organized as follows:

Chapter 2: Presents the literature review of this thesis. This chapter gives a description on routing metrics and routing protocols.

Chapter 3: Reports on the methodology used for this thesis. It shows the routing metrics design and implemented. It also presents the simulation setup used in this thesis.

Chapter 4: Presents and discusses the results obtained from the simulation setup used in this thesis.

Chapter 5: This is the concluding chapter of this research. A summary of the whole thesis is firstly presented followed by concluding remarks and finally, the area future work.

CHAPTER 2

LITERATURE REVIEW

2.1 Chapter Overview

This chapter firstly, presents routing metrics overview; this includes their requirements and their design issues. Secondly, routing metrics for WMNs are discussed with an emphasis on routing metrics such as ETX, ETT, WCETT, EPL and MHEB. Finally, an overview on routing protocols is presented; the popular reactive routing protocols AODV, DSR and TORA are discussed.

2.2 Routing Metrics Overview

This section presents routing metrics design requirements. It then discusses some popular routing metrics.

2.2.1 Routing Metrics Requirements

Designing routing metrics has a significant impact on performance in wireless mesh networks. Based on their characteristics, WMNs do not perform well using existing solutions e.g. minimum hop count metric from both wired and wireless networks and impose unique requirements on designing routing metrics for mesh networks. Amongst these requirements are :

- (i) **Route stability:** Unstable path weights can considerably reduce the performance of any network. A high number of route update messages creation is caused by constant changes. They can also disturb normal network functions as routing protocols may not congregate under frequent route updates. The stability of route costs is defined by the nature of path properties captured by the routing metrics, which can be either load-sensitive or topology-dependent. Load-sensitive metrics allocate a cost to a route-based on the traffic load on the route. The departure and arrival of traffics can lead to a constant change of route under load-sensitive metrics.

On the other hand, topology-dependent metrics assign a cost to a route based on the topological characteristics of the route, such as the hop count and link capacity of the path. Consequently, topological-dependent metrics are usually more stable, especially for static networks where the topology does not constantly change. Topology-dependent routing metrics can be used with both on-demand and proactive routing protocols; as a result, they are more desirable for mesh networks. Topology-dependent and load-sensitive metrics can be used with different routing protocols, because routing protocols have different levels of tolerance of route cost instability.

(ii) **Good performance for minimum weight paths:** For efficient mesh networks resources utilization, the minimum weight paths selected by the routing protocols should have good performance in terms of high throughput and low packet end-to-end delay. To accomplish this, the routing metrics must be able to take into account the characteristics of mesh networks that impact the performance of paths. These characteristics are as follows [12]:

- **Paths length:** each hop generates additional delay and probably more packets drop; as a result, a longer path generally increases the end-to-end delay and decreases the overall throughput of the network. Thus, a routing metric should increase the weight of a path when the path's length increases.
- **link capacity:** Since current wireless cards are able to adapt their transmission rates based on the quality of the channel by changing their modulation schemes, the link capacity decreases when the distance between the nodes increases.
- **Packet loss ratios:** Different wireless links can have different packet loss ratios. A node might have to retransmit a packet several times on a link with a high packet loss ratio; this has an effect on both the throughput and the delay of any flow that goes through the link. For this reason, a routing metric should be able to capture the packet loss ratios to ensure good performance for the minimum weight path.
- **Interferences:** interference can be classified into two categories in WMNs; inter-flow and intra-flow interferences. Inter-flow interference refers to the

interference which occurs when flows operating on the same channels are competing for the medium. So it is the interference suffered amongst concurrent flow [13]. Figure 2.1 shows an example of inter-flow interference between the node D and the node E. On the other hand, intra-flow interference refers to interference which takes place when adjacent nodes compete for the same channel bandwidth. In other word, it is the interference which occurs when different nodes transmitting packets from the same flow interfere with one another [13]. Figure 2.2 shows an example of intra-flow interference between the node A and the node B on channel 1 (CH=1). So, to find minimum weight paths with good performance, routing metrics should be able to consider both intra-flow and inter-flow interferences.

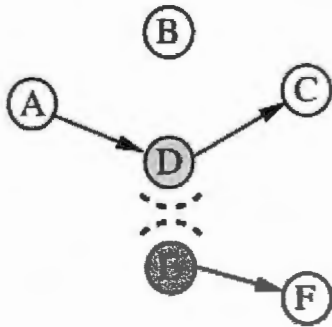


Figure 2.1: Inter-flow interference between two nodes

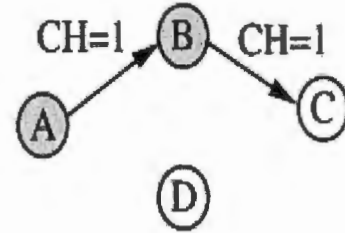


Figure 2.2: Intra-flow interference between two nodes

(iii) **Efficient algorithms to calculate minimum weight paths:** Even though a given routing metric guarantees that its minimum weight paths have good performance, there is no guarantee that a routing protocol can have good performance if there is no efficient algorithm to calculate the minimum weight paths based on the routing metric. The fundamental and adequate condition for the existence of such efficient algorithms is that the routing metrics should have a property called isotonicity. The isotonic property basically refers to the fact that a metric ought to guarantee that the order of the weights of two paths is preserved if they are appended by a similar third path.

(iv) **Loop-free outing:** Sobrinho pointed out that a metric must be isotonic to ensure that no routing loops can be formed when hop-by-hop routing is combined with Dijkstra's algorithm [12, 14].

2.2.2 Routing Metrics for Wireless Mesh Networks

This sub-section describes some of the routing metrics for WMNs proposed by the research community. These routing metrics are: ETX, ETT, WCETT, EPL, and MHEB.

2.2.2.1 Expected Transmission Count (ETX).

The ETX routing metric was proposed by De Couto et al. [15]. The ETX routing metric takes into consideration the loss rates of the links [16] as well as the path length and the impact of the link lost ratio in the selected path. ETX discovers the path with the smallest number of expected number of transmissions (including retransmissions) required for a source node to send a packet to its destination. The metric predicts the number of retransmissions required using per-link measurements of packet loss ratios in both directions of each wireless link. The main goal of the ETX routing metric is to find paths with high throughput, in spite of losses. To achieve this goal, ETX has the following characteristics:

- ETX is based on delivery ratios, which directly have an impact on the throughput;
- ETX identifies and handles asymmetry by integrating loss ratios in each direction;
- ETX uses accurate link loss ratio measurements to make intelligent decisions for routes selection;
- ETX discourages the selection routes with more hops, which have lower throughput due to interference between different hops of the same path;
- ETX have a tendency of minimizing the spectrum use, resulting in an overall system capacity improvement; and
- ETX may decrease the energy consumption per packet, as each transmission or retransmission may increase the energy consumed by a node.

The total ETX of a route is found by adding up the ETX of each link in the route. The forward delivery ratio is the measured probability that a data packet sent by the source node has been successfully received by the destination node; the reverse delivery ratio is the measured probability that the ACK packet sent by the destination node has been successfully received by the source node. ETX selects the path by using the following equation:

$$ETX = \frac{1}{d_f \times d_r} \quad (1)$$

Where:

- d_f is the forward delivery ratio, and
- d_r is the reverse delivery ratio.

In addition, ETX is also an isotonic metric, which guarantees easy calculation of minimum weight paths and loop-free routing under all routing protocols. However, the ETX routing metric have some shortcomings, some of these are described below:

- ETX does not consider interference [17];
- ETX does not take into account the fact that different links may have different transmission rates [1, 18];
- ETX does not consider the packets size [1];
- ETX does not specifically consider the mobility of nodes [15]; and
- ETX does not accurately capture the asymmetry of traffic on the wireless link [10] resulting in a poor link quality evaluation.

2.2.2.2 Expected Transmission Time (ETT).

The ETT routing metric was proposed by Draves et al. [19] in an attempt to address some of the limitations of ETX. ETT captures the effect of heterogeneity on the performance of the chosen path. It captures the multiple transmission rates at which packets are transmitted over the network. The ETT routing metric allocates lower

weights to a high-bandwidth link compared to a low-bandwidth link, as long as the loss rates on the two links are equal. Hence, ETT prioritizes fast links with minimal error rates [11]. It computes the transmission expected value in the MAC layer by reflecting link bandwidth and packet size; this is probably why the authors in [11] define ETT as a “*bandwidth-adjusted ETX*”. The ETT is defined as follows:

$$ETT = ETX \times \frac{PacketSize}{Bandwidth} \quad (2)$$

This metric is also isotonic. However, the major drawback of ETT is that it does not consider the channel diversity, leading to inter-flow and intra-flow interference in the networks [20]. Another disadvantage of ETT is that it does not take into account the link quality.

2.2.2.3 Weighted Cumulative Expected Transmission Time (WCETT).

The WCETT routing metric was proposed by Drave et al. [19] in an attempt to reduce the intra-flow interference. The WCETT was designed to take into account the link bandwidth and the loss rates of links in the presence of multiple non-overlapping channels [17]. It was designed using the following assumptions [19]:

- All nodes in the network are stationary.
- Each node must be equipped with one or more IEEE 802.11 radios. These can be a combination of IEEE 802.11a, IEEE 802.11b or IEEE802.11 g radios. Each node is not required to have the same number of radios.
- If a node has multiple radios, they are tuned to different, non-interfering channels.

The three main design goals of the WCETT routing metric are as follows [19]:

- WCETT should be able to take into account the bandwidth and the loss rate of a link when considering it for inclusion in a route.
- The route metric combining the cost of individual links should be increasing. That is, if a hop is added to an existing route, the cost of the route should never decrease but should increase.

- A route built up of hops on various channels is better than a route where all the hops are on the same channel.

WCETT minimizes the intra-flow interference by allowing a sender node to optimize the entire route taken by its flow. The WCETT of the path p is defined as follows:

$$WCETT_p = (1 - \beta) \sum ETT + \beta MAXX_j \quad (3)$$

Where:

- β is a tunable parameter between $0 \leq \beta \leq 1$ which controls the preferences over path length versus channel diversity, and
- X_j represents the number of times the channel j is used on links in the end-to-end path.

The WCETT metric balances channel diversity and path length, by varying the value of the control parameter β . It attempts to address some of the shortcomings of the ETT routing metric. However, some of the weaknesses experienced by WCETT are that it is not isotonic and it does not solve the problem of inter-flow interference.

2.2.2.4 Expected Link Performance

The EPL metric was proposed by Ashraf et al. [10]. EPL uses both link quality and cross-layering to gather wireless channel information in order to estimate link performance. EPL also takes into account the asymmetry of traffic on the wireless link. The authors in [10] proposed to solve the problem of asymmetry by assigning a higher weight to the forward link in order to moderate the asymmetry of the packets. This is because the reverse link is meant for the ACK packets which are loss resistant and would probably be successfully received almost regardless of estimated reverse delivery ratio. The authors in [10] calculate the link delivery ratio P as follows:

$$P = \alpha d_f + (1 - \alpha) d_r \quad (4)$$

Where α is the weight assigned to the forward delivery ratio d_f and is subject to $0.5 < \alpha < 1$.

ELP is a hybrid link metric which attempts to correctly approximate the link performance by combining link interference information along with link quality information. For optimal performance improvement of the whole network, the main goal is to choose the routes with minimal interference. The key idea is that routes with minimal interference should be preferred to achieve global optimum. However, Equation (4) only takes into account the link quality. The authors in [10] used a cross layer mechanism to incorporate and estimate interference on the link in the metric for optimal routes selection. They define Expected Link Performance as follow:

$$EPL = \frac{1}{\alpha d_f + (1 - \alpha) d_r} \quad (5)$$

EPL takes into consideration the inter-flow and intra-flow interferences as well as the asymmetry of the packets to estimate the link quality. However, EPL was only evaluated in a single-radio scenario. Another drawback of EPL is that, it uses cross layering which usually lead to loss of the protocol layer abstraction. Cross layer design can lead to a "spaghetti system" which is usually difficult to maintain [21, 22]

2.2.2.5 Multi-Hop Effective Bandwidth (MHEB)

The MHEB routing metric was proposed by LI et al. [23]. MHEB provides a generic approach to calculate the achievable bandwidth along a path, taking the impacts of inter-flow and intra-flow interference and channel diversity into account. Li et al. developed an approach to compute the achievable bandwidth under intra-flow interference (ABIRF). In order to accurately capture the inter-flow interference, the authors in [23] combined the ETX metric with a newly proposed interference degree ratio to evaluate the achievable bandwidth under the inter-flow interference (ABITF). The routing metric MHEB is defined as the weighted average of ABIRF and ABITF. The Researchers in [23] evaluated the ABITF at a link i as follows:

$$ABITF_i = (1 - IDR_i) \times \frac{Bandwidth_i}{ETX_i} \quad (6)$$

Where IDR_i is the Interference Degree Ratio for a link i , $Bandwidth_i$ is the original bandwidth at a link i , and ETX_i denotes the expected transmission attempts for a successful transmission over link i .

Authors in [23] also computed the ABIRF as follows:

$$ABIRF = \frac{Bandwidth_x \times Bandwidth_y}{Bandwidth_x + Bandwidth_y} \quad (7)$$

Where $Bandwidth_x$ is the maximum bandwidth for a channel x , and $Bandwidth_y$ is the maximum channel for the channel y .

Using Equations (6) and (7), the authors in [23] compute MHEB as follows:

$$MHEB = \beta \times minABITF_i + (1 - \beta) \times ABIRF \quad (8)$$

Where β is a tunable parameter and subject to $0 \leq \beta \leq 1$.

The MHEB routing metric effectively captures the effects of inter-flow and intra-flow interference, and channel diversity along a path. However, it does not guarantee optimal path selection. Authors in [23] also evaluated the performance of MHEB using fixed mesh nodes. These could cause a bias in the results as mesh client nodes in WMNs are usually mobile.

2.3 Routing Protocols

The challenges and flexibility of Wireless Networks have generated a lot of research in routing protocols for such networks. The network research community has been intensively working on modeling, designing and implementing new routing protocols for wireless networks. Researchers in [24] classify Wireless networks routing protocols into four categories proactive protocols, reactive protocols, hybrid protocols and cluster-based protocols. Three popular reactive routing protocols, DSR, AODV and TORA and a popular proactive routing protocol, OLSR, will be discussed in the next section.

2.3.1 Reactive Routing Protocols.

Reactive routing protocols establish the route only when needed and there is no need for a node to maintain routes to destinations that are not in active communication.

Whenever a source node wish to know the route to a destination node, it floods the network with a route request message. This algorithm has considerable low routing overhead for light traffic. The main disadvantages of such protocols are longer delays in establishing the routes for forwarding the data and the excessive flooding of the control messages may lead to network clogging.

2.3.1.1 Ad hoc On Distance Vector (AODV)

AODV routing protocol was first proposed by an Internet Engineering Task Force (IETF) Internet draft in 1997 [25]. According to Belding-Royer and Perkins [26], AODV was designed to achieve the following objectives:

- Minimal control overhead: this is done because resources are scarce in Mobile Ad hoc Networks (MANETs). AODV tries to minimize the control overhead by utilizing only on-demand messaging and by eliminating periodic routing updates.
- Minimal processing overhead: this is done by ensuring that AODV messages are simple and require little computation.
- Multi-hop path routing capability: In Wireless networks, source nodes and destination nodes may be out of direct communication range with each other due to the limited transmission range of the wireless medium. Thus, AODV provides nodes with the ability to discover multi-hop paths to destinations.
- Dynamic topology maintenance: AODV gives the nodes the ability to maintain the multi-hop paths even when the network topology is dynamically changing.
- Loop prevention: routing loop very expensive in MANETs; so, AODV uses the destination sequence numbers to prevent routing loops.

One of the main characteristics of AODV is that it uses a destination sequence number created by the destination and used for ensuring loop freedom. Given a choice of multiple routes to a destination node, the source node selects the one with the greatest sequence number. Also, the most recent destination sequence number associated with

the destination and lifetime of the route is stored in the table. If during the lifetime, the route is not used, the routing table entry is discarded. When errors occur or there is a change in the selected route, AODV has the ability to create a new route for the rest of transmission, establishment and maintenance [27].

AODV defines three types of message types: Route Request (RREQ), Route Reply (RREP) and Route Error (RERR) [25, 26, 27]. It uses these message types to discover the route from the source node to the destination node via the User Datagram Protocol (UDP). The RREQ format, RREP format and RERR format are respectively shown in Figure 2.3, Figure 2.4 and Figure 2.5. When a route to a new destination is required, a node broadcasts the RREQ message to find a route. A route is found when the RREQ reaches the destination itself, or when it reaches an intermediate node that has an updated route to the destination. The route is made available by unicasting the RREP message back to the source of the destination. Since each node that receives the RREQ caches the route back to the source, the RREP can be unicasted to the origination of the RREQ. The link status of active routes is continuously monitored for any link breakage. When a link breaks, RRRER message is propagated down the route to notify the affected nodes about the loss of link. The purpose of the RRRER message is to indicate which destinations are now unreachable because of the link breakage. Each node keeps a list containing the Internet Protocol (IP) address of each of its neighbours that are likely to use it as a next hop towards each destination.

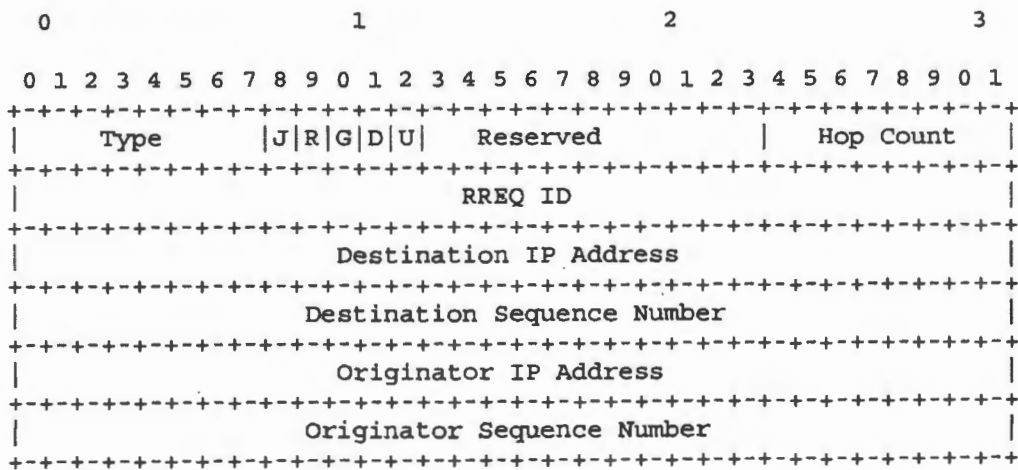


Figure 2.3: RREQ message format [26]

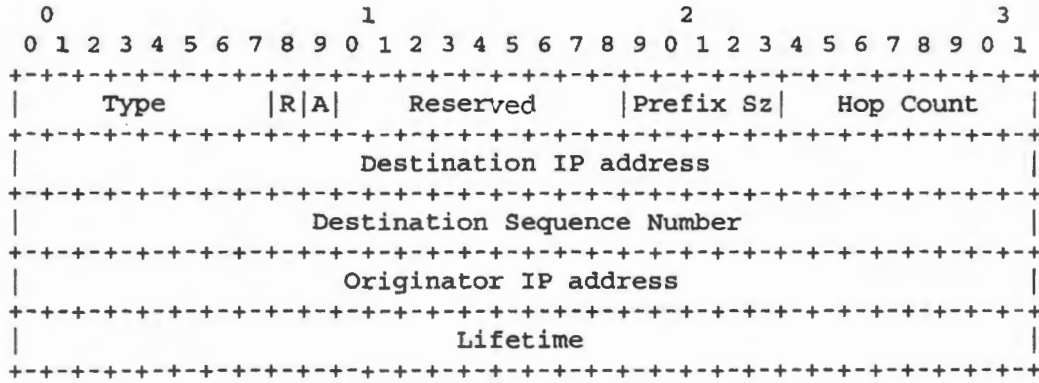


Figure 2.4: RREP message format [26]

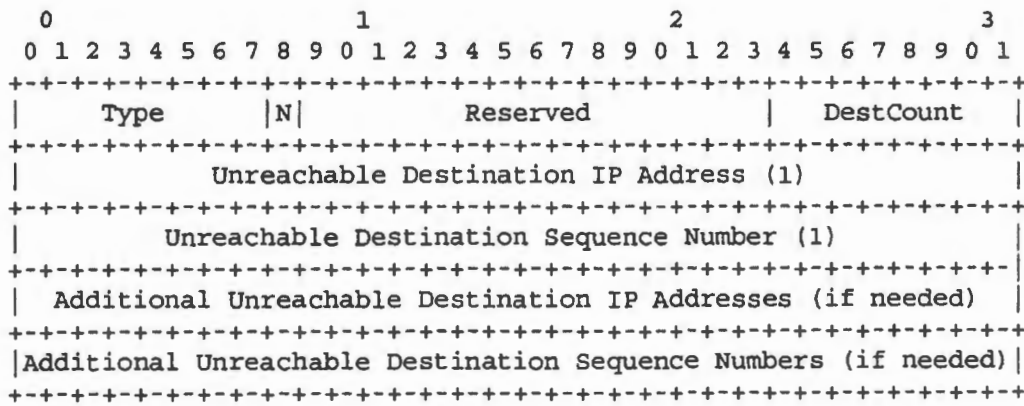


Figure 2.5: RRER message format [26]

The operation of AODV is done using the following two mechanisms: route discovery and route maintenance [25, 26, 27].

a) Route discovery

Route discovery is a mechanism whereby a source node wishing to send a packet to a destination node obtains dynamically a source route when it does not have a route in its routing table. Route discovery on its own is divided into two mechanisms:

1) RREQ broadcasting

When a destination is previously unknown to the source node or the route to the destination is no longer valid, the node disseminates a RREQ. The Destination Sequence Number (DSN) is the last known DSN for this destination. If no sequence number is known, the unknown sequence number flag is set. After broadcasting a RREQ, the node source waits for a RREP. When a mobile node receives a RREQ, it creates or updates the route to the previous hop without a

valid sequence number and then checks if it has received the RREQ with the same originator IP address and RREQ ID. If such a RREQ has been received, the node silently discards the newly received RREQ. If the RREQ received has not been discarded, the intermediate node searches for a reverse route to the originator IP address. If the reverse route already exists, it is updated only if the originator sequence number is higher than the destination sequence number of the originator IP address in the routing table or if the sequence numbers are equal but the hop count in RREQ is smaller than the existing hop count in the routing table. The RREQ is rebroadcast if the active route does not exist in its routing table or if the existing DSN is smaller than the DSN field of the RREQ. The Time to Live (TTL) in the outgoing RREQ is decremented by one and the hop count field is incremented by one to account for the new hop through the intermediate node.

2) RREP generation and forwarding

A node generates a RREP when itself is the destination or when it has an updated route to satisfy the request. The RREP is unicast to the next hop towards the originator of the RREQ. As the RREP is propagated, the hop count field in RREP is incremented by one at each hop. When a node receives a RREP, it searches for a route to the previous hop. Once the route to the destination is created or updated in the route table, the route is marked active and the next hop is assigned to the node from which RREP was received. Consequently, the node can use this route to forward the data to the destination. If the node is not the node indicated by the Originator IP address in RREP, then the RREP packet is forwarded towards the next hop, selected based on the route table entry. When a node forwards a RREP, the antecedent list for the corresponding destination node is updated by adding to it the next hop node to which the RREP is forwarded. Since the RREP is forwarded along the route established by the RREQ, AODV only supports the use of symmetric links [28]. Examples of RREQ messages from a source node and RREP from a destination node are respectively illustrated in Figure 2.6 and Figure 2.7

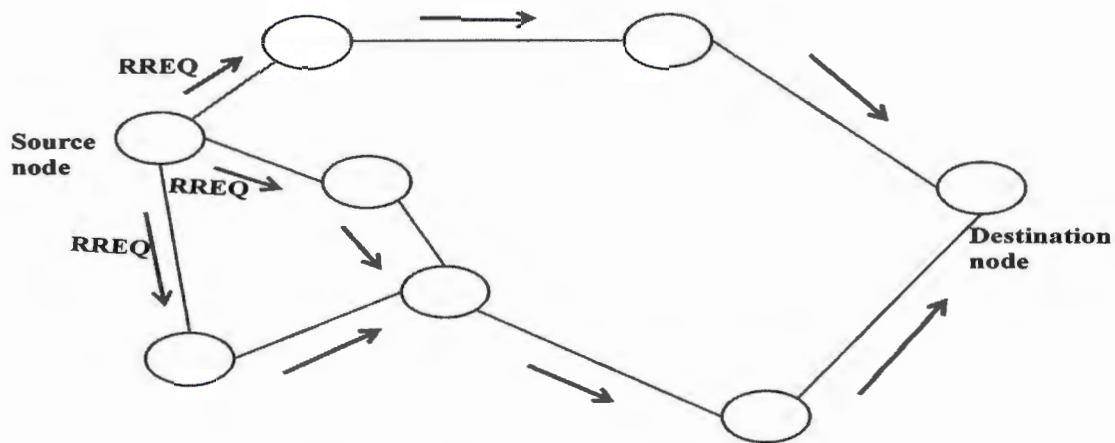


Figure 2.6: RREQ message broadcasted by the source node

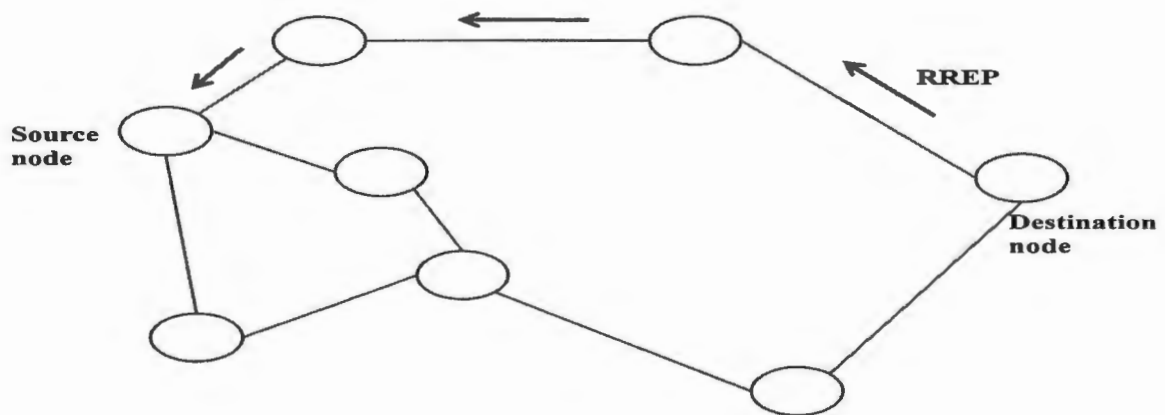


Figure 2.7: RREP message unicasted to the next hop towards the source node

b) Route maintenance

Once a route has been established, the source node will maintain the route for as long as it needs it. The movement of nodes not lying along the active route does not affect the routing to that path's destination. If a source node moves during an active session, it can reinitiate a new route discovery to the destination. However, if either the destination or an intermediate node moves, a RERR message is sent to the source node. To detect link failures, HELLO messages are used. AODV also uses HELLO messages broadcasted periodically to immediate neighbours to detect whether the links are broken. If HELLO messages stop coming from a particular neighbour, a node notifies the affected set of nodes by sending a RERR message.

Once the next hop becomes unreachable, the RERR message is initiated by the node upstream of the broken link (i.e., node closer to the source). The node inserts in the RERR message a fresh sequence number that is greater than the previously known sequence number, changes the hop count to ∞ , and then propagates the packet to all upstream neighbours. They in turn relay that message to their neighbors and so on. This process continues until the affected source node is notified. Upon receiving the RERR message of a broken link, the source node can reinitiate route discovery if it still needs it.

2.3.1.2 Dynamic Source Routing (DSR)

DSR is a reactive routing protocol developed at Carnegie Mellon university Pittsburgh USA for use of multi-hop wireless MANET [29]. DSR allows the network to be completely self-organizing and self-configuring [30]. The source node specifies a complete path in the header of a data packet which contains an ordered list of nodes through which the packet must traverse. The DSR header format is shown in figure 2.8. Nodes cooperate by forwarding packets for each other to allow communication over multiple hops between nodes that are outside the wireless transmission range of each other. As the network topology changes by nodes coming or leaving the network, or existing nodes moving, all routing is automatically determined by the DSR protocol. Nodes are required to maintain route caches that can contain multiple source routes to any destination. Entries in the route cache are continually updated as new routes are learned.



Figure 2.8: DSR header format [29]

The operation of DSR is done using the following two mechanisms: route discovery and route maintenance [24, 29].

a) Route discovery

Route discovery is a mechanism whereby a source node wishing to send a packet to a destination node obtains dynamically a source route. Route discovery is used only when the source node does not know a route to the destination. When a source node has a packet addressed to some destination node, it places in the header of the packet a source route containing the sequence of hops that the packet should follow on its way to the destination. Usually, the source node will obtain a suitable source route by searching its Route Cache (a table of paths previously learned). However, if no route is found in its cache, the source node will initiate a Route discovery to dynamically find a new route to the destination node.

When initiating route discovery, the initiating node time stamps and saves a copy of the original packet in a local buffer called Send Buffer. The Send Buffer contains a copy of each packet that cannot be transmitted by this node due to the lack of source route to the packet's destination.

Occasionally, the source node initiates a new route discovery for the packet's destination address, but at limited rate (to prevent a large number of unproductive Route Requests) for it is possible that the destination node is not reachable due to limited wireless transmission range plus the possibility that the network might be partitioned. In order to reduce the overhead from such route discoveries, the source node should use an exponential back-off algorithm to limit the rate at which it initiates new route discoveries for the same destination.

b) Route maintenance

Route maintenance is performed when there is an error with an active route. When a node of the network that is part of some route notices that it cannot send packets to the next hop, it will create a RERR and send it to the source node. The RERR message contains the addresses of the node that sent the packet and of the next hop that

is unreachable. When the RERR message reaches the source node, the source node removes all routes from its route cache that have the erroneous node address. It then initiates route discovery for a new route if needed.

DSR protocol guarantees loop-free routing and very rapid recovery when routes in the network change [16, 30].

2.3.1.3 Temporally-Ordered Routing Algorithm (TORA)

According to Park and Corson [31], TORA is a reactive routing protocol that possesses the following properties:

- Distributed execution;
- Loop-free routing;
- Multipath routing;
- Reactive or proactive route establishment and maintenance; and
- Minimization of the communication overhead via algorithmic reaction to topological changes.

TORA is an efficient, highly adaptive, and scalable routing protocol based on link reversal algorithm. TORA provides multiple routes to transmit data packet between source and destination nodes of the MANET.

TORA defines four types of control packets: query packet (QRY), update packet (UPD), clear packet (CLR), and optimization packet (OPT) [31]. QRY packet format, UPD packet format, CLR packet format and OPT packet format are respectively shown in Figure 2.9, Figure 2.10, Figure 2.11 and Figure 2.12.

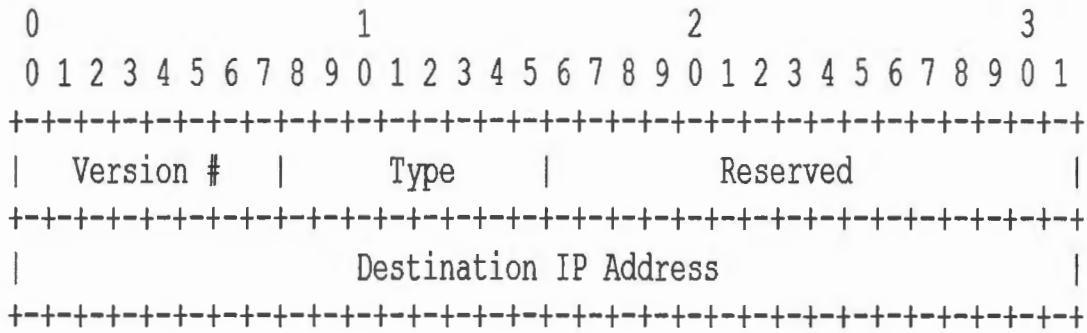


Figure 2.9: Query (QRY) Packet Format [31]

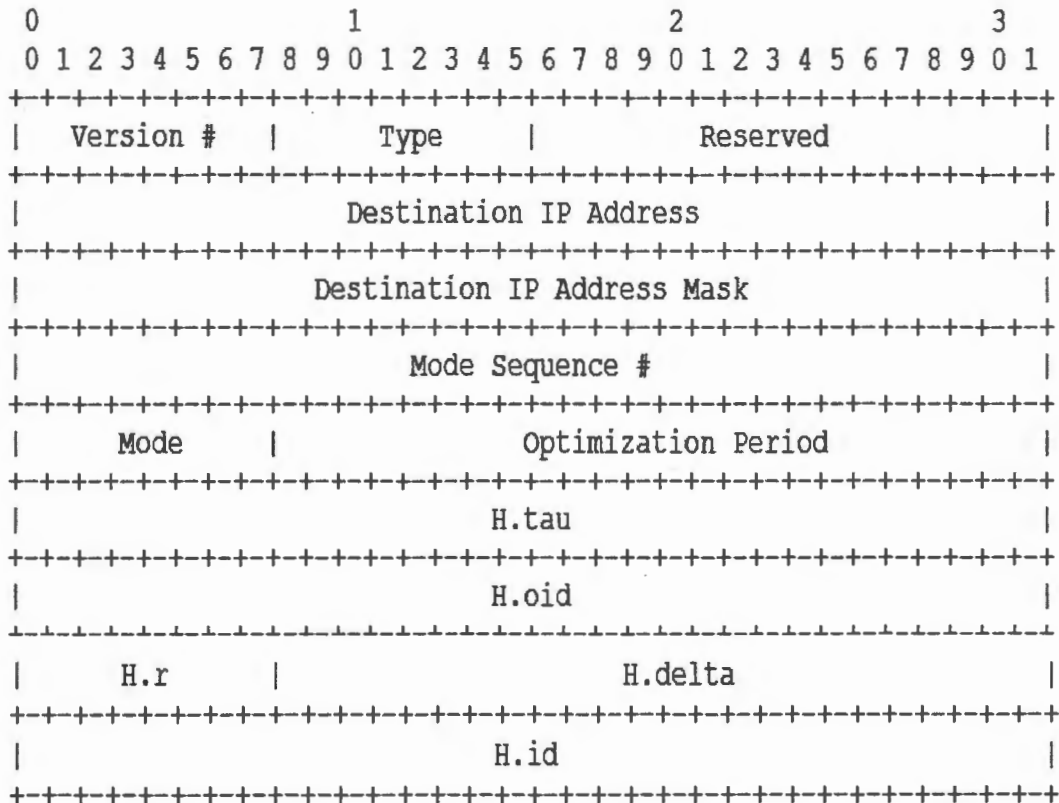


Figure 2.10: UDP packet format [31]

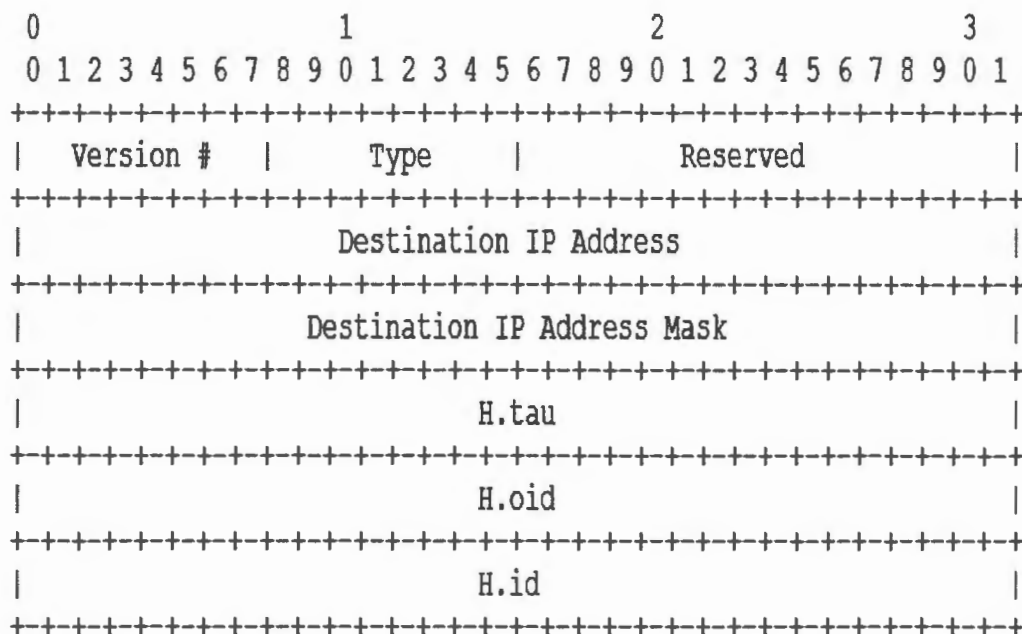


Figure 2.11: CLR packet format [31]

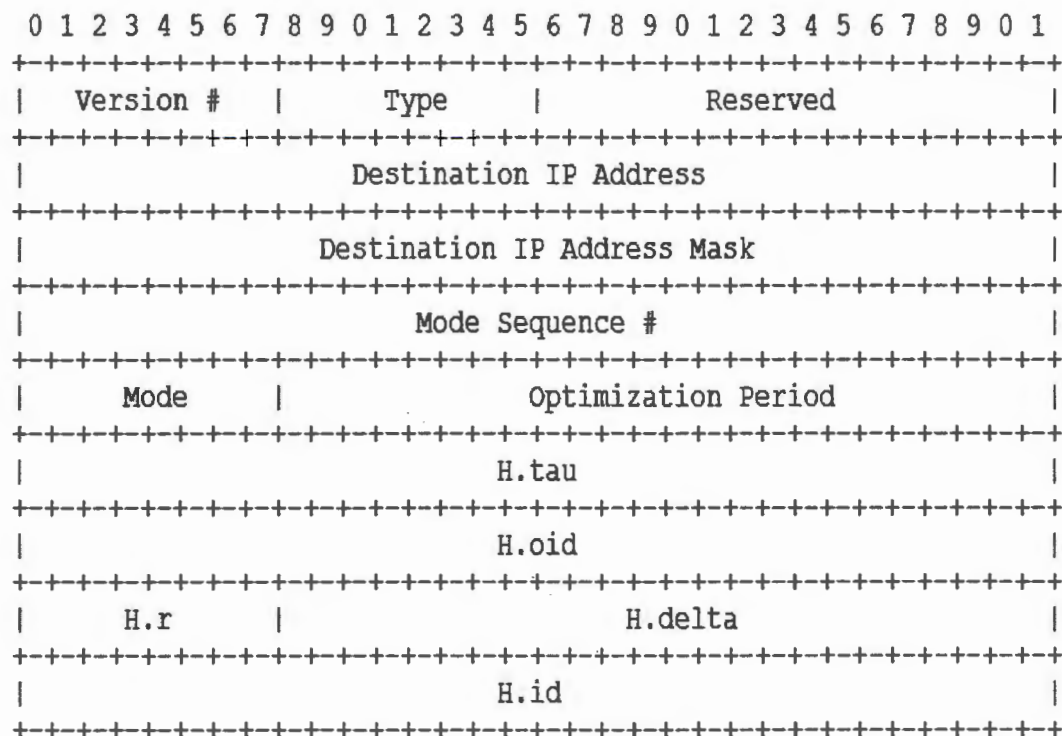


Figure 2.12: OPT packet [31]

As stated earlier, TORA minimizes communication overheads by localizing algorithmic reaction to topological changes when possible. At each node in the network, a logically separate copy of TORA is run for each destination. When a source node needs a route to a destination, it broadcasts a QRY packet containing the address of the destination node for which it requires a route. This packet propagates through the network until it reaches either the destination, or an intermediate node having a route to the destination. The node receiving the QRY packet then broadcasts an UPD packet listing its height with respect to the destination. The node receiving the UPD packet sets its height to a value greater than the height of the neighbour from which the UPD packet was received. This has the effect of creating a series of directed links from the original sender of the QRY packet to the node that initially generated the UPD packet. In the case of a route failure, nodes adjust their height to a local maximum with respect to its neighbours and transmit an UPD packet. If the node has no neighbors of finite height with respect to this destination, then the node instead attempts to discover a new route. When a node detects a network partition, it generates a CLR packet that resets the routing state and removes invalid routes from the network.

The TORA routing protocol consists of three basic functions: creating routes, maintaining routes, and erasing routes [30, 31]. Creating routes corresponds to the selection of heights to form a directed sequence of links leading to the destination in a previously undirected network or portion of the network. Maintaining routes refers to adapting the routing structure in response to network topological changes. For example, following the loss of some router's last downstream link, some directed paths may temporarily no longer lead to the destination. This event triggers a sequence of directed link reversals (caused by the re-selection of router heights), which re-orient the routing structure such that all directed paths again lead to the destination. In cases where the network becomes partitioned, links in the portion of the network that has become partitioned from the destination must be marked as undirected to erase invalid routes. During this erasing routes process, routers set their heights to null and their adjacent links become undirected.

2.3.2 Proactive routing protocols

Proactive routing protocols maintain the routing information of all the participating nodes and update their routing information frequently irrespective of the routing requests. Proactive routing protocols transmit control messages to all the nodes and update their routing information even if there is no actual routing request. This makes proactive routing protocols bandwidth deficient, though the routing itself is simple having this prior updated routing information. A lot of proactive routing protocols have been proposed and implemented. According to Guo et al. [32], OLSR is the most popular proactive routing protocol. The next section discusses a well known proactive routing protocol, OLSR.

2.3.2.1 Optimized Link State Routing (OLSR)

OLSR is an MANET proactive routing protocol that uses the concept of Multi Point Relays (MPRs) [33]. MPR is an optimized flooding control protocol used by OLSR to construct and maintain routing tables by diffusing partial link state information to all nodes in the network [16, 33]. OLSR uses two kinds of control messages: HELLO and TOPOLOGY CONTROL (TC) [33]. This control traffic is communicated using the User Datagram Packet (UDP) of the Transport Layer defined by a general packet format, as shown in Figure 2.13, omitting UDP and IP headers where each packet encapsulates one or more messages. HELLO messages are used to find information about the link status of the host's neighbours. With the HELLO message, the MPR Selector set is constructed which lists neighbours that have chosen this host as a MPR and from this information, the host can calculate its own set of the MPRs. The HELLO messages are forwarded only one-hop away while the TC messages, on the other hand, are forwarded like usual broadcast messages in the entire network. TC messages are used for broadcasting information about own advertised neighbours which includes at least the MPR Selector list. The TC messages are broadcasted periodically and only the MPR hosts can forward them.

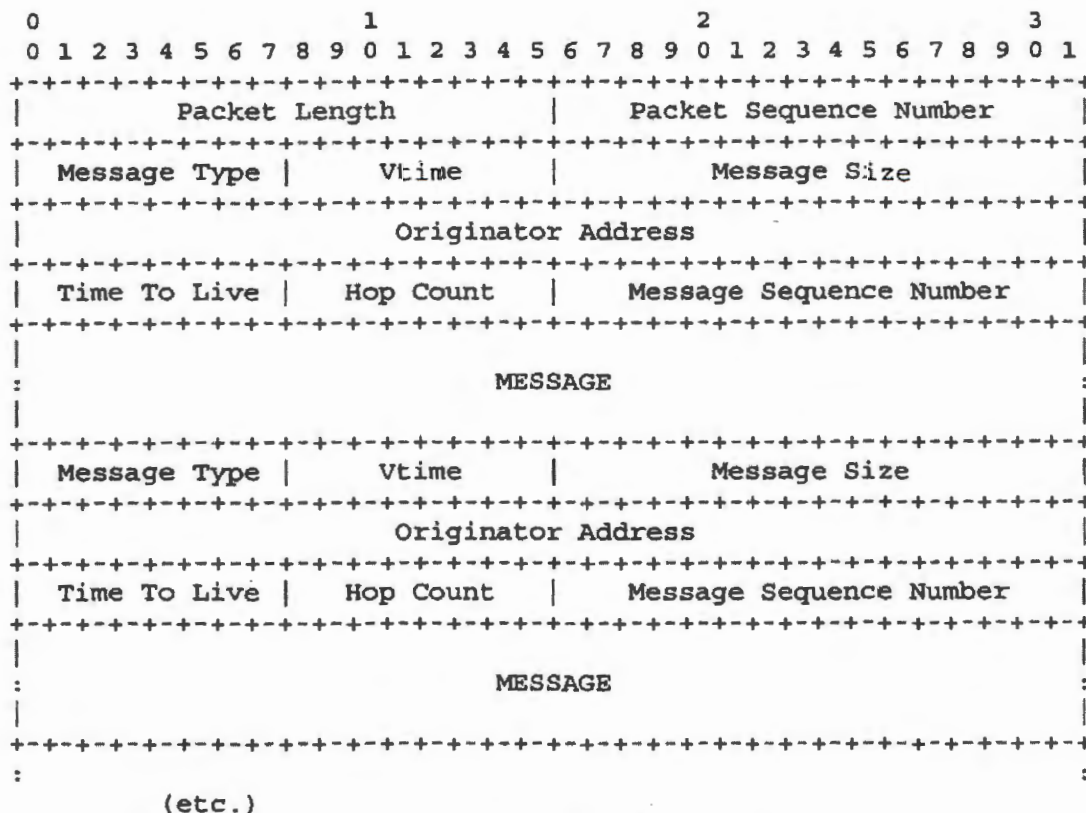


Figure 2.13: OLSR packet format [33]

The functioning of OLSR can be divided into the following three mechanisms:

- Neighbour/Link Sensing;
- Efficient control flooding using MPR; and
- Optimal route calculation.

a) Neighbor/Link Sensing.

A link in the ad hoc network can be either uni-directional or bi-directional and a host node must know this information about its neighbours. Each node periodically broadcasts its HELLO messages, as shown in Figure 2.12, with information about its neighbours and their link status. These control messages are transmitted in a broadcast mode. They are received only by one-hop neighbors and are not relayed to further nodes. When the first host receives a HELLO message from the second host, it sets the second host status to asymmetric in the routing table. When the first host sends a HELLO message and includes information about the link to the second host as

asymmetric, the second host sets the first host status to symmetric in its routing table. Finally, when the second host sends again a HELLO message, where the status of the link to the first host is indicated as symmetric, the first host changes the status from asymmetric to symmetric. In the end both hosts know that their neighbour is alive and the corresponding link is bi-directional. The HELLO messages allow each node to learn information about its neighbours up to two hops away. The HELLO messages periodic broadcasting is used for link sensing, neighbour's detection and MPR selection process. The message contains information on how often the host sends HELLO messages, willingness of host to act as a Multipoint Relay, and information about its neighbour. Information about the neighbours contains: interface address, link type and neighbour type. The link type indicates whether the link is symmetric, asymmetric or simply lost. The neighbour type is either symmetric, MPR or not a neighbour.

b) Efficient control flooding using MPR

MPR is the key concept used in OLSR to reduce flooding of broadcast packets [16, 33]. OLSR uses MPRs to reduce the number of hosts which broadcast the information throughout the network. The MPR is a host's one-hop neighbour which may forward its messages. The MPR set of a host is kept small in order for the protocol to be efficient. In OLSR, only the MPRs can forward control traffic throughout the network. Each host must have information about the symmetric one-hop and two-hop neighbours in order to calculate the optimal MPR set. Figure 2.14 shows an example of MPR. Information about the neighbours is taken from the HELLO messages. The two-hop neighbours are found from the HELLO message because each HELLO message contains all the hosts' neighbours. Selecting the minimum number of one-hop neighbours which cover all the two-hop neighbors is the goal of the MPR selection algorithm. Also, each host has the Multipoint Relay Selector set which indicates which host has selected the current host to act as a MPR. When the host gets a new broadcast message, which is needed to be spread throughout the network and the message's sender interface address is in the MPR Selector set, the host must forward the message. Due to the possible changes in the MANET, the MPR Selectors sets are updated continuously using HELLO messages.

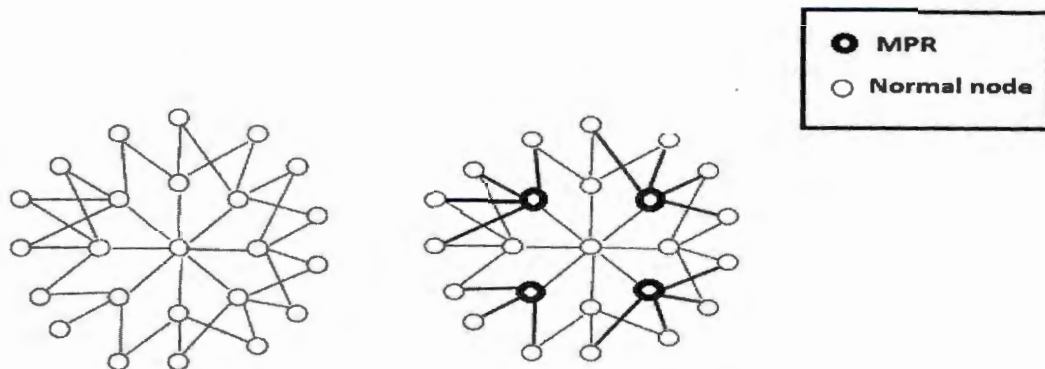


Figure 2.14: Normal broadcast (left) and optimized broadcast using MPRs (right).

The good functioning of MPRs in OLSR depends on two main factors: MPR selection and topology information [33].

1) MPR selection

Each node of the network selects its own set of MPRs. The host must have information about one- and two-hop symmetric neighbours in order to start needed calculations for the MPR set. All exchange of information is broadcast using HELLO messages. The MPR set is calculated in a manner that includes a minimum number of one-hop symmetric neighbors from which it is possible to reach all the symmetric two-hop neighbours. Only neighbours that are symmetric and willing to act as MPRs are selected. In subsequent HELLOs, a given node declares multipoint relays and transmits them so that the information reaches the multipoint relays themselves. The MPR set gets recalculated when:

- There is a change in the neighbourhood that detected due to a failure of a bi-directional link with a neighbour, or an addition of a neighbour with a bidirectional link; or
- There is a change in the two-hop neighbour where a bi-directional link is detected.

2) Topology Information

Each node within the network keeps topology information about the network. This information is acquired from TC messages and used for building routing tables. In order to exchange the topological information and build the topology information base, nodes that were selected as MPRs need to send a TC message. The format of the TC message is shown in Figure 2.15. TC messages are broadcast throughout the network and only MPRs are allowed to forward them. The TC messages are generated and broadcast periodically in the network. The TC message is sent by a host in order to advertise its own links. The host must send at least the links of its MPR selector set. The TC message includes its set of advertised links and the sequence number of each message. The sequence number is used to avoid loops of the messages and for indicating the freshness of the message. So, if the host gets a message with the smaller sequence number, it must discard the message without any updates. The host must increment the sequence number when the links are removed from the TC message and also, it should increment the sequence number when the links are added to the message. The sequence numbers are wrapped around. When the host's advertised links set becomes empty, it should still send empty TC messages for a specified amount of time in order to invalidate previous TC messages. This should stop sending the TC messages until it has obtained some information to send.

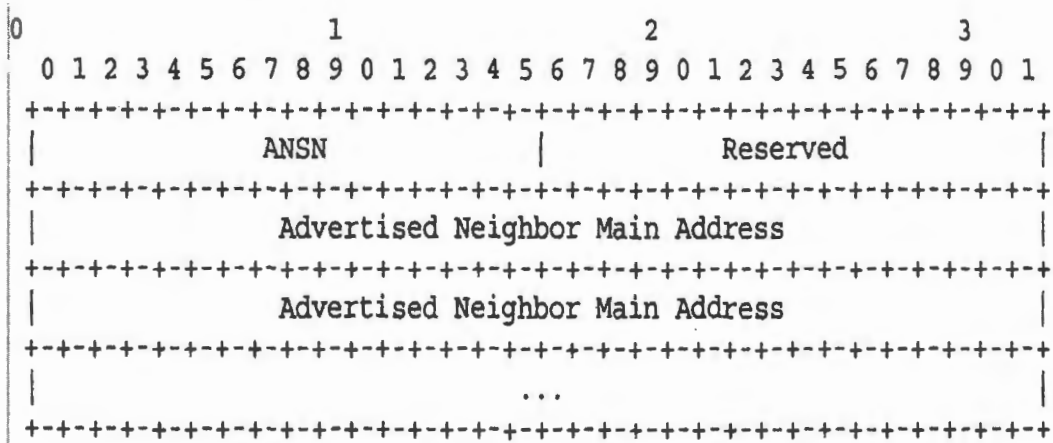


Figure 2.15: TC message format [33]

The size of the TC message can be quite large so the TC message can be sent in parts but then, the receiver must combine all parts during some specified amount of time. A host can increase its transmission rate to become more sensitive to possible link failures. When a change in the MPR Selector set is noticed, it indicates that the link failure has happened and the host must transmit a new TC message as soon as possible.

3) Optimal route calculation

A routing table is maintained by every node, which is then refreshed and updated whenever a change in the topology is detected. To populate a routing table, shortest path algorithm is used on the partial topology graph obtained from TC messages. It is important to note that OLSR is not involved in forwarding data packets.

In this section, three reactive routing protocols, DSR, AODV and TORA and a proactive routing protocol, OLSR were explained. Table 2.1 summarizes and compares them.

Characteristics	AODV	DSR	TORA	OLSR
Routing philosophy	Reactive	Reactive	Reactive	Proactive
Type of routing	Hop-by-hop routing	Source routing	Hop-by-hop routing	Hop-by-hop routing
Frequency of updates	As needed	As needed	Based on the mode of operation	Periodically
Worst case	Full flooding	Full flooding	Full flooding	Pure link state
Multiple routes	No	Yes	No	No

Table 2.1: summary and comparison of the four MANET routing protocols studied earlier [34].

2.4 Chapter Summary

This chapter presented a brief overview on WMNs routing metrics requirements and design considerations. It then briefly discussed some common routing metrics used for WMNs. Popular routing protocols for wireless networks were also presented in this chapter. The next chapter presents the routing metrics designed as well as the novel routing protocols proposed in this thesis. The simulation setup is also briefly described in the next chapter.

CHAPTER 3

ROUTING METRICS DESIGN, SIMULATION AND EXPERIMENTAL SETUP

3.1 Chapter Overview

This chapter presents a report on the design of two (improved and Extended improved) routing metrics and their implementation into two reactive protocols. Firstly, the improved Multi Hop Effective Bandwidth (*i*MHEB) routing metric was designed as an improvement of the MHEB. Secondly, the Extended improved Multi Hop Effective Bandwidth (XiMHEB) routing metric was also design as an enhancement of the *i*MHEB. The *i*MHEB routing metric was implemented into the AODV routing protocol and the XiMHEB routing metric was implemented into the DSR routing protocol.

This chapter also presents the modeling and simulation of two WMNs used for the performance evaluation of the routing metrics. The simulator, as well as the simulation parameters used for simulation in this thesis are also discussed.

3.2 Routing Metrics Design

The proposed routing metrics designed in this thesis are discussed in the following subsection.

3.2.1 Improved Multi-Hop Effective Bandwidth

The new metric, named *i*MHEB [35], is an enhancement of the MHEB routing metric. The *i*MHEB takes into consideration the asymmetry of traffic on the wireless links to improve the MHEB routing metric. As proposed by authors in [23], the ABITF at a link i is given by Equation (5), page 16. The *i*MHEB metric computes an improved Achievable Bandwidth under Inter-Flow Interference (*i*ABITF) by biasing the weight towards the forward link; this is done by assigning a greater weight to the forward link

in order to reduce the asymmetry of packets and thus enhancing the MHEB routing metric. The equation to compute the $iABITF$ at a link i is given as follows:

$$iABITF_i = (1 - IDR_i) \times \frac{Bandwidth_i}{\frac{1}{\alpha d_f + (1-\alpha)d_r}} \quad (9)$$

In reformulating Equation (9), we have the following:

$$iABITF_i = (1 - IDR_i) \times Bandwidth_i \times [\alpha d_f + (1 - \alpha)d_r] \quad (10)$$

Where:

- IDR_i is the Interference Degree Ratio for a link i ;
- $Bandwidth_i$ is the original bandwidth at a link i ;
- α is the weight assigned to the forward delivery ratio d_f and is subject to $0.5 < \alpha < 1$;and
- d_r is the reverse delivery ratio.

The achievable bandwidth under intra-flow interference is the same as in Equation (7). From Equations (7) and (9), the proposed routing metric $iMHEB$ can be defined as follows:

$$iMHEB = \beta \times \min [(1 - IDR_i) \times Bandwidth_i \times [\alpha d_f + (1 - \alpha)d_r]] + (1 - \beta) \times ABIRF \quad (11)$$

Where:

- $ABIRF$ is as defined in Equation (7); and
- β is a tunable parameter and is subject to $0 \leq \beta \leq 1$.

The $iMHEB$ routing metric achieves link quality, inter-flow and intra-flow interference awareness. However, $iMHEB$ does not take into consideration actual packets size. As a solution to this, the $XiMHEB$ routing metric is proposed; the $XiMHEB$ routing metric is described in the next subsection.

3.2.2 Extended improved Multi-Hop Effective Bandwidth

The proposed routing metric called $XiMHEB$ is an extension to the $iMHEB$ [35] routing metric. The routing metric is a hybrid multi-capability routing metric which takes into

account the link quality, inter-flow and intra-flow interferences as well as the size of packets during transmission and the raw data rate. It also incorporates the actual time spent using a particulate channel. The proposal to compute an Extended improved Achievable Bandwidth under Inter-Flow Interference (XiABITF) which differs from *i*ABITF is due to the fact that XiABITF consistently estimate the bandwidth (raw data rate) rather than using the rates recovered from the firmware. It also accurately captures the packets size transmitted. The computation of XiABITF is given as follows:

$$XiABITF_i = (1 - IDR_i) \times \frac{\frac{Bandwidth_i}{1} \times \frac{Packet Size}{ad_f + (1-\alpha)d_r}}{Bandwidth_i} \quad (12)$$

It follows therefore that:

$$XiABITF_i = (1 - IDR_i) \times [ad_f + (1 - \alpha)d_r] \times \frac{Bandwidth_i^2}{Packet Size} \quad (13)$$

where the ABIRF is defined as in Equation (7). From Equations (7) and (13), the XiMHEB is defined routing metric using the following equation:

$$XiMHEB = \beta \times \min [(1 - IDR_i) \times [ad_f + (1 - \alpha)d_r] \times \frac{Bandwidth_i^2}{Packet Size}] + (1 - \beta) \times ABIRF \quad (14)$$

Where:

- *ABIRF* is as defined in Equation (7);
- β is a tunable parameter and is subject to $0 \leq \beta \leq 1$.

The XiMHEB routing metric easily and simultaneously achieve link quality, inter-flow and intra-flow awareness and also takes into account the effect of the heterogeneity, space diversity, and link capacity on the performance of the chosen path.

3.3 Routing Metrics Implementation

As stated earlier, the *i*MHEB routing metric was implemented into the AODV reactive routing protocol resulting to a novel WMN routing protocol named AODV-*i*MHEB. The XiMHEB routing metric was implemented into the DSR reactive routing protocol leading to new WMN routing protocol called DSR-XiMHEB. The following subsection shows how the routing metrics *i*MHEB and XiMHEB routing metrics were implemented into the corresponding routing protocols using a flow chart.

3.3.1 Improved Multi Hop Effective Bandwidth Implementation

The step on how the *i*MHEB was implemented into the AODV routing mechanism is shown in Figure 3.1. An additional field was used on the Route Request (RREQ), Route Reply (RREP) and Route Error (RERR) messages format. The *i*MHEB routing metric was computed from Equation (11).

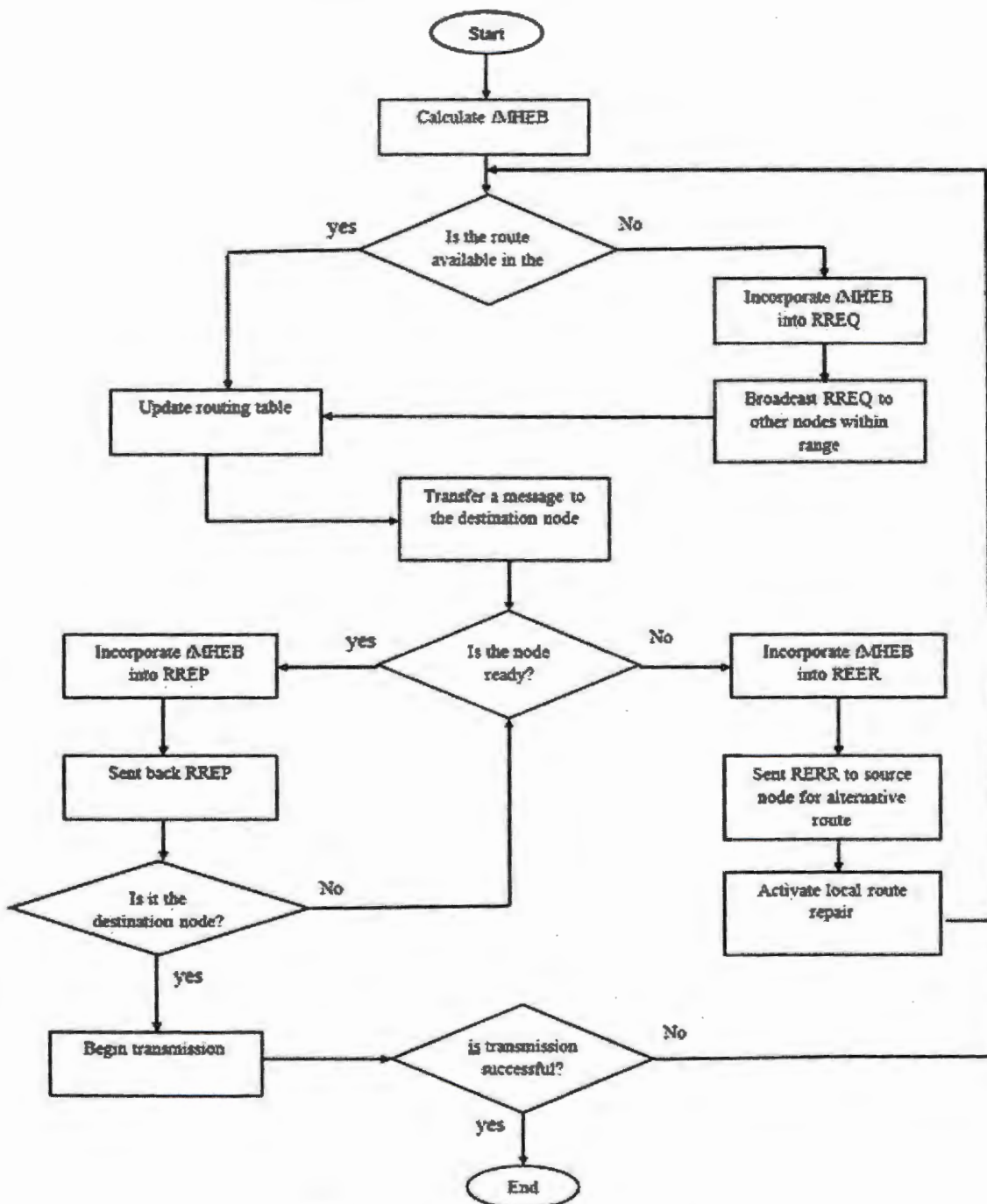


Figure 3.1: Flow chart of the AODV-*i*MHEB

When a source node has data to transmit to an unknown destination node, it incorporates the *i*MHEB routing metric into the RREQ and then broadcasts a RREQ for that destination. At each intermediate node, when a RREQ is received, a route to the source is created. If the receiving node has not received this RREQ before, it is not the destination and if does not have an up to date route to the destination, it rebroadcasts the RREQ. If the receiving node is the destination or has an up-to-date route to the destination, it generates a RREP. The *i*MHEB is incorporated in the RREP; which is then unicast to the source node. As the RREP propagates, each intermediate node creates a route to the destination. When the source node receives the RREP, it keeps a record of the route to the destination in its routing table before it can begin to transmit. If the intermediate node is not ready to receive an RREP, it then generates an RERR. The *i*MHEB is then incorporated into the RERR before the local route repair mechanisms start.

3.3.2 Extended improved Multi Hop Effective Bandwidth Implementation

The step on how the *Xi*MHEB was implemented into the DSR routing mechanism is illustrated in Figure 3.2. An additional field was used on the Route Request (RREQ), Route Reply (RREP) and Route Error (RERR) messages format. The *Xi*MHEB routing metric is computed from Equation (14). When a source node wishes to send a packet to a destination node previously unknown to the source node, it must first incorporate the *Xi*MHEB into its RREQ and then broadcast the RREQ to all its neighbours within range before updating its routing table. If the destination node is ready, it incorporates the *Xi*MHEB into the RREP and then sends the RREP back to the source node via the chosen path; otherwise, the routing metric is integrated into the RERR and sent back to the source node for alternative route. RREP has all the information about the path it uses back to the source node. The route selected is the route with the least inter-flow and intra-flow interference, as well as the route with the best link quality. The modified routing protocol with the *Xi*MHEB routing metric is named DSR-*Xi*MHEB.

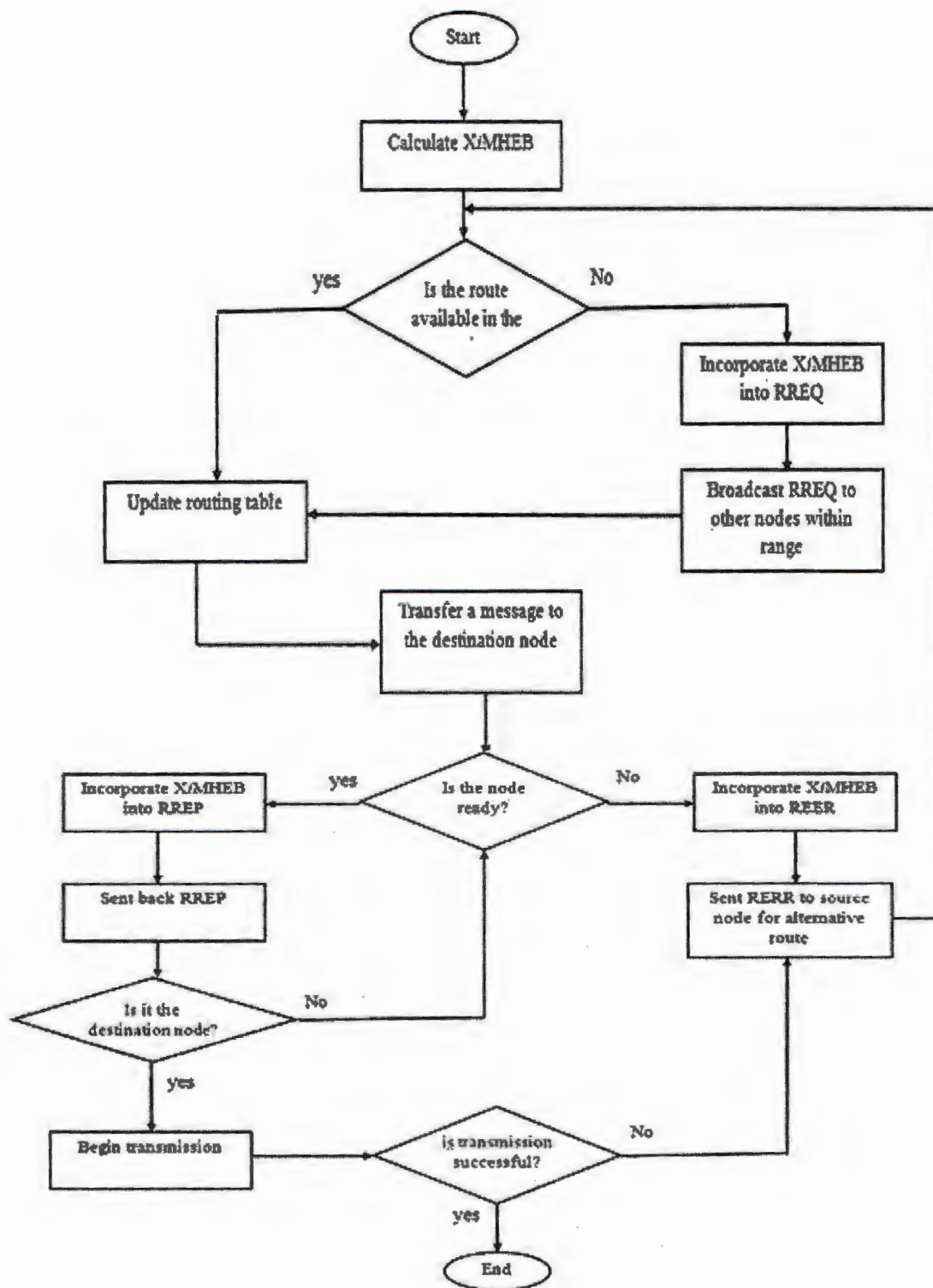


Figure 3.2: Flowchart of the DSR-X/MHEB

3.4 Simulation Software Tool

In WMNs research, several simulation software tools exist. Amongst this simulation software tools are:

- (i) **NS-2**: This is a discrete event simulator primarily designed to simulate wired networks. Several extensions of the simulator have been implemented to support wireless networks. The core modules of NS-2 are written and implemented in C++ paradigm.
- (ii) **OMNET++**: This is a discrete event simulation simulator. It supports wireless communications simulation. Its modules are written and implemented in C++.
- (iii) **OPNET**: This is a discrete event simulator written using the C++

For the purpose of this research, we used OPNET Modeler 14.0 for modeling and simulations.

OPNET (Optimized Network Engineering Tools)

OPNET is a simulation tool used for simulation and analysis for network communication and distributed systems. The preference place of this tool is because of its the obvious merits over others existing simulation tools. Some of these merits are as follows:

- (i) OPNET provides a very good graphical user interface and is very easy to learn.
- (ii) OPNET can be used to model the entire network, including its routers, switches, protocols, servers, and the individual applications they support. OPNET can also support a large range of communication systems from a single local area network to global inter-networks.
- (iii) OPNET software has a free version for students called OPNET IT Guru Academic Edition at home which can be downloaded and installed.
- (iv) The OPNET's discrete event engine for network simulations is the fastest and most scalable commercially available solution.

OPNET consists of hierarchical models and these are divided into three main domains: the network domain, the node domain, and the process domain.

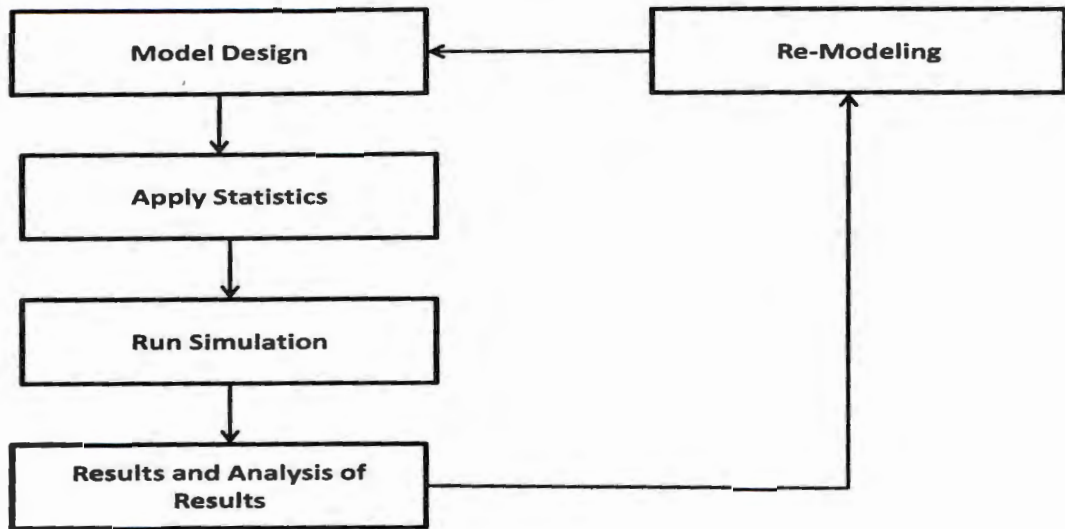


Figure 3.3: OPNET workflow

The modeling of network in OPNET usually involves four steps: model design, applying statistics, run simulation, view results and to analyze the results. For different scenarios, using for example different metrics or different protocols, the network has to be re-modeled. Figure 3.3 shows the basic OPNET workflow as implemented in this thesis.

3.5 Simulation Setup

In this thesis, two major topologies were used for diversity namely; the random topology and the grid topology. The random topology was used for the implementation of *i*MHEB and the grid topology was used for the implementation of *Xi*MHEB. Both topologies consisted of mesh clients, mesh routers and Wireless Local Area Network (WLAN) server for traffic generation. The Mesh clients and mesh routers both used the data rate adaptive channel assignment algorithm (DR-CA) [36] for multi-channel multi-rate wireless communication.

3.5.1 Performance Metrics

The evaluation of the network performance can be done using different metrics such as the throughput, the load, the delay, the jitter, the packet delivery ratio, the routing overhead, the number of transmission per packet and many others. However, in this thesis, and based on the profiles modeled, we used the throughput, the routing overhead

and the delay as a function of the load; that is under light and heavy HTTP loads. The summary of the performance metrics used in this thesis is summarize in Table 3.1

Performance Metric	Definition	Desired function
Throughput	It is the sum of data packets generated by every source in the network. It is expressed in bits per seconds	High throughput is desirable
Delay	it is the time it takes for a packet to be transmitted from the source node to the destination nodes. It is expressed in seconds	Low delay is desirable
Routing Overhead	It is the total number of control packets generated by a routing protocol. It is expressed in packets. Low routing overhead is desirable.	Low Routing Overhead is desirable

Table 3.1: Routing metrics used for performance evaluation

HTTP light load is characterized by the following parameters [37]:

- Page Rate (Pages/hour): 5
- Page Size (Objects/page): 10
- Average Object Size (bytes/object): 12 000.

HTTP heavy load is characterized by the following parameters [37]:

- Page Rate (Pages/hour): 60
- Page Size (Objects/page): 10
- Average Object Size (bytes/object): 12,000

3.5.2 Random Topology

The WMN modeled consists of 20 mesh clients, a Wireless Local Area Network (WLAN) server and 4 mesh routers randomly distributed in $1000m \times 1000m$ square area. The mesh nodes have applications running over TCP/IP and UDP/IP. They support wireless multi-rate communication with rates of up to 11Mbps. The mesh nodes implemented moved using the Random Waypoint mobility model with a speed of 10 meters per second and a pause time of 100 seconds. The WLAN server has applications running over TCP. The interference range was set to 250 meters. The MAC protocol used was the IEEE 802.11b. The traffic sources followed the HTTP traffic with packet sizes of 1,024 bytes. As stated earlier, the OPNET modeler consists of network model, node model and process model.

Network model: A network model defines the overall scope of a system to be simulated. It is a high-level description of the objects contained in the system. The network model specifies the objects in the system, their physical locations, interconnections and configurations. It consists of network topologies described in terms of subnetworks, nodes, links, and geographical coordinates. The network model used for the random topology is shown in Figure 3.4

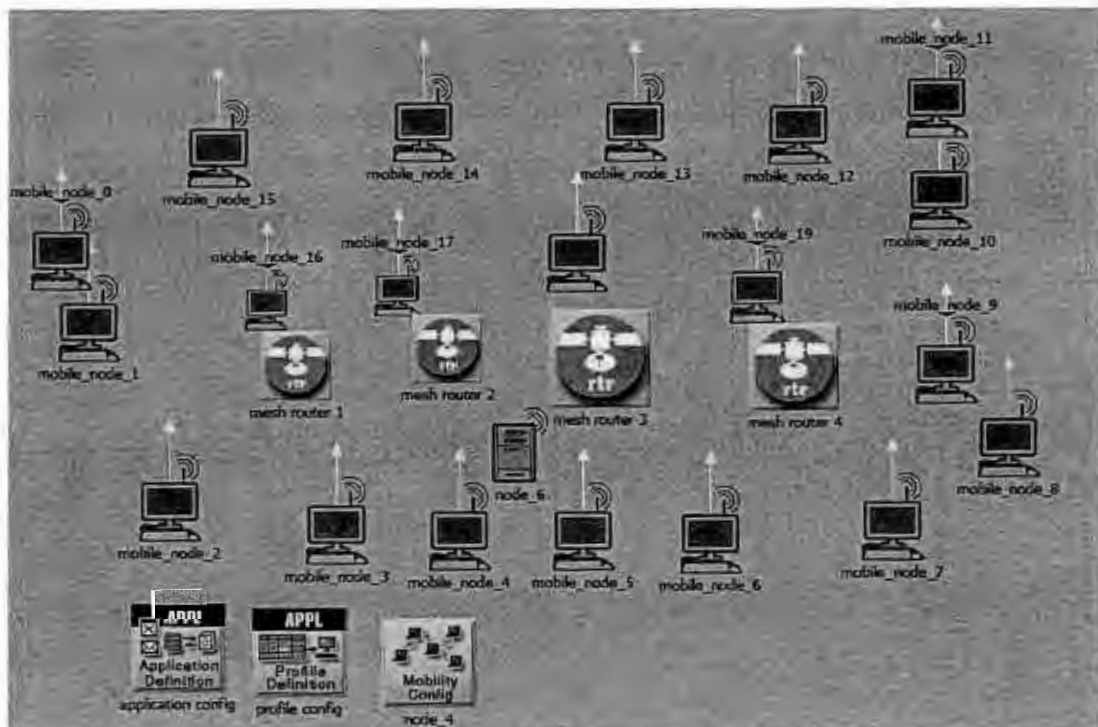


Figure 3.4: Random topology network model

Node model: A node model is composed of a series of connected blocks called modules. Each module contains a set of inputs and outputs, some state memory, and a method for computing the module's outputs from its inputs and its state memory. The node domain consists of single network nodes such as routers, workstations and mobile devices. The node model used in this random topology is illustrated in Figure 3.5 below.

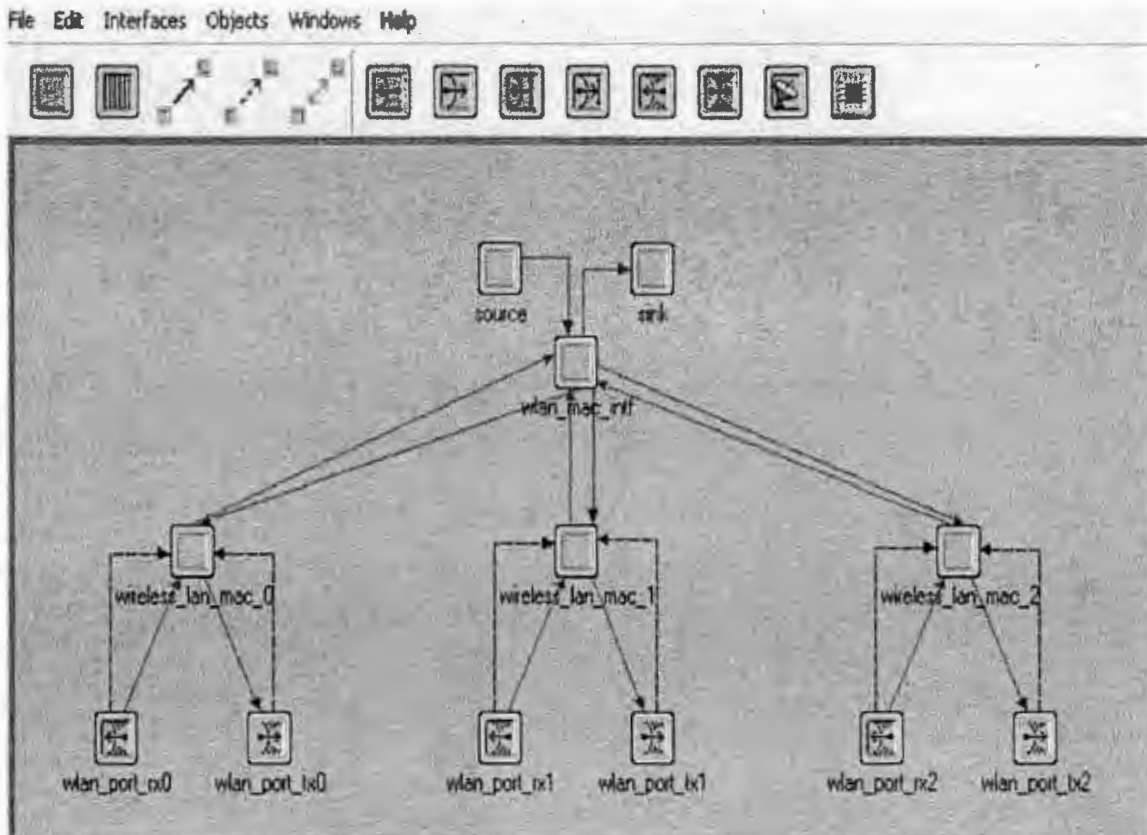


Figure 3.5: Simulation node model

Process model: The process model for the *i*MHEB used for the random topology is represented as a Finite State Machine (FSM) shown in Figure 3.6. The FSM machine consists of two states: the initial (init) state and the wait state. The initial state consists of all the state variable initializations and the registration used to collect various statistics during simulation. The wait state handles the computation of the *i*MHEB as shown in Figure 3.2. The computation of *i*MHEB is based on the number of packets received in 10 seconds and then finding forward and reverse delivery ratios.

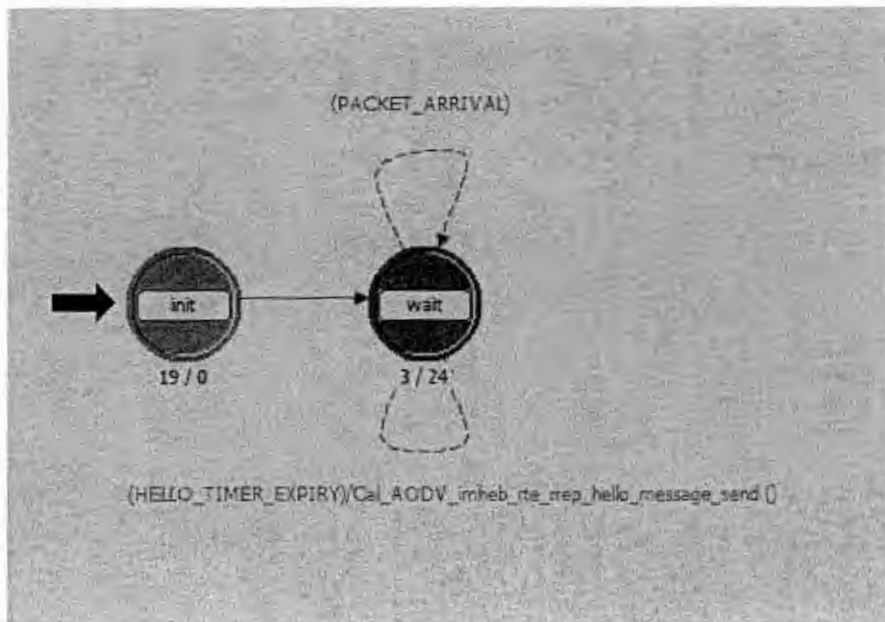


Figure 3.6: Process model of *i*MHEB in random topology

The routing metrics WCETT and MHEB were also incorporated into the reactive routing protocol AODV for performance comparison. An additional field was added to the AODV protocol to calculate ABIRF, ABITF and *i*ABTF, hence update the routing metrics MHEB and *i*MHEB when a route request or a route reply is received by a node. The calculation is based on the number of packets received in 10 seconds. The value of α was set to 0.75 and the value of β was set to 0.5.

3.5.3 Grid Topology

The WMN modeled consists of 20 mesh clients, a Wireless Local Area Network (WLAN) server and 4 mesh routers distributed in a grid (5×5) $1000m \times 1000m$ square area. The mesh nodes have applications running over TCP/IP and UDP/IP. They support wireless multi-rate transmission with rates of up to 11Mbps. The mesh clients implemented moved using the Random Waypoint mobility model with a speed of 10 meters per second and a pause time of 100 seconds. The WLAN server had applications running over TCP. The interference range was set to 250 meters. The IEEE 802.11b was used as The IMAC protocol. The traffic sources generated followed the HTTP

traffic with packet sizes of 1,024 bytes. For this topology, each scenario was run for 3600 seconds.

Network model: The network model for the grid topology is shown in Figure 3.7

Node model: The node model for this 5×5 grid topology was the same as the one used for the random topology.

Process model: The process model for the XiMHEB routing metric implemented into DSR in the grid topology is represented by a FSM shown in Figure 3.8. The FSM also consists of two states: the init state and the wait state. The init state consists of all the state variable initializations and the registration used to collect various statistics during simulation. The wait state handles the arrival of packet and the calculation of the XiMHEB as illustrated in Figure 3.2. The computation of *i*MHEB is based on the number of packets received in 10 seconds and then finding forward and reverse delivery ratios.

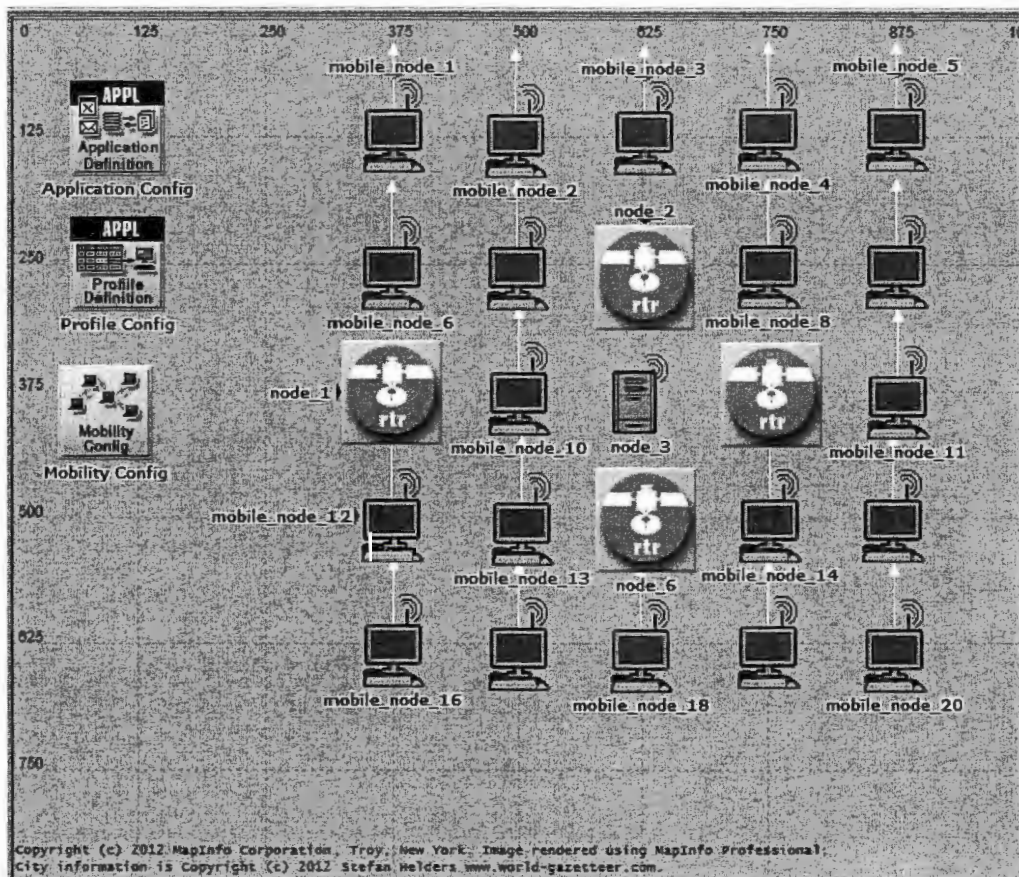


Figure 3.7: Grid Topology network model

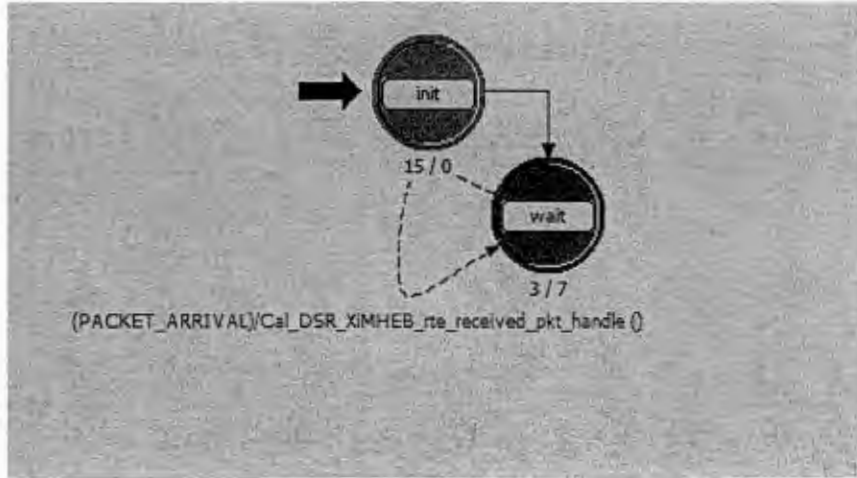


Figure 3.8: Process model for the grid topology

The routing metrics EPL and *i*MHEB were also incorporated into DSR for performance comparison. An additional field was added to the DSR protocol to calculate ABIRF, *i*ABTF and XiABTF hence update the routing metrics *i*MHEB and XiMHEB when a RREQ, RREP or a RERR is received by a node. The calculation was based on the number of packets received in 10 seconds. The value of α was set to 0.75 and the value of β was set to 0.5.

3.6 Chapter Summary

In this chapter, the design of two routing metrics named *i*MHEB and XiMHEB were presented. The *i*MHEB was implemented into the AODV routing protocol resulting to a new AODV based routing protocol. The XiMHEB routing metric was implemented into the DSR reactive routing protocol.

This chapter also presented the modeling, design and simulation setup of two WMNs with different topologies: the random topology and the grid topology. The simulation parameters and the performance metrics used in the modeling were also defined and discussed.

In the next chapter, the analysis, evaluation and discussion of the results obtained from the modeling and simulation of the WMNs will be presented.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Chapter Overview

In this chapter, the simulation results are presented and discussed. The results obtained are based on both the random and grid topologies. The performance metrics used for analysis are throughput, delay and routing overhead as a function of the load; that is under light and heavy load HTTP traffic loads.

4.2 Simulation Results for *i*MHEB

These results are presented under the random topology; each discussion based on throughput, delay and routing overhead.

4.2.1 Random Topology

4.2.1.1 Throughput Comparisons Under Light and Heavy HTTP loads

In terms of throughput, the performance of the AODV-*i*MHEB, AODV-MHEB and AODV-WCETT routing protocols for light and heavy load is respectively illustrated in Figures 4.1 and 4.2. Under light and heavy HTTP traffic load, Figures 4.1 and 4.2 respectively show that the routing protocol AODV-*i*MHEB outperforms AODV-MHEB and AODV-WCETT routing protocols. This discrepancy is firstly due to the fact that AODV-*i*MHEB attempts to choose the path with the least interference and the least latency from the source node to the destination node, resulting in a higher throughput. Secondly, it is also due to the fact that AODV-*i*MHEB takes into account the asymmetry to improve the link quality estimation by assigning a higher weight to the forward delivery ratio. Hence, the optimal path selection used by the routing protocol AODV-*i*MHEB is improved. Under light load, Figure 4.1 shows that the AODV-MHEB routing protocol performs better than the AODV-WCETT protocol since the AODV-MHEB routing protocol considers both inter-flow and intra-flow interference while AODV-WCETT only takes into account intra-interference; that is, the AODV-WCETT

routing protocol always considers all links in the computation of intra-flow interference even if they are not within the other nodes interference ranges. Consequently, the routes created sometimes suffer from high levels of interferences; resulting to a low throughput. Under heavy load traffic, Figure 4.2 shows that the AODV-MHEB routing protocol competes with AODV-WCETT for the best second routing protocol in terms of the network throughput. Again, this is due to the fact that under heavy load, AODV-MHEB does not necessarily choose the best route to the destination node, because it does not properly estimate the link quality of the network.

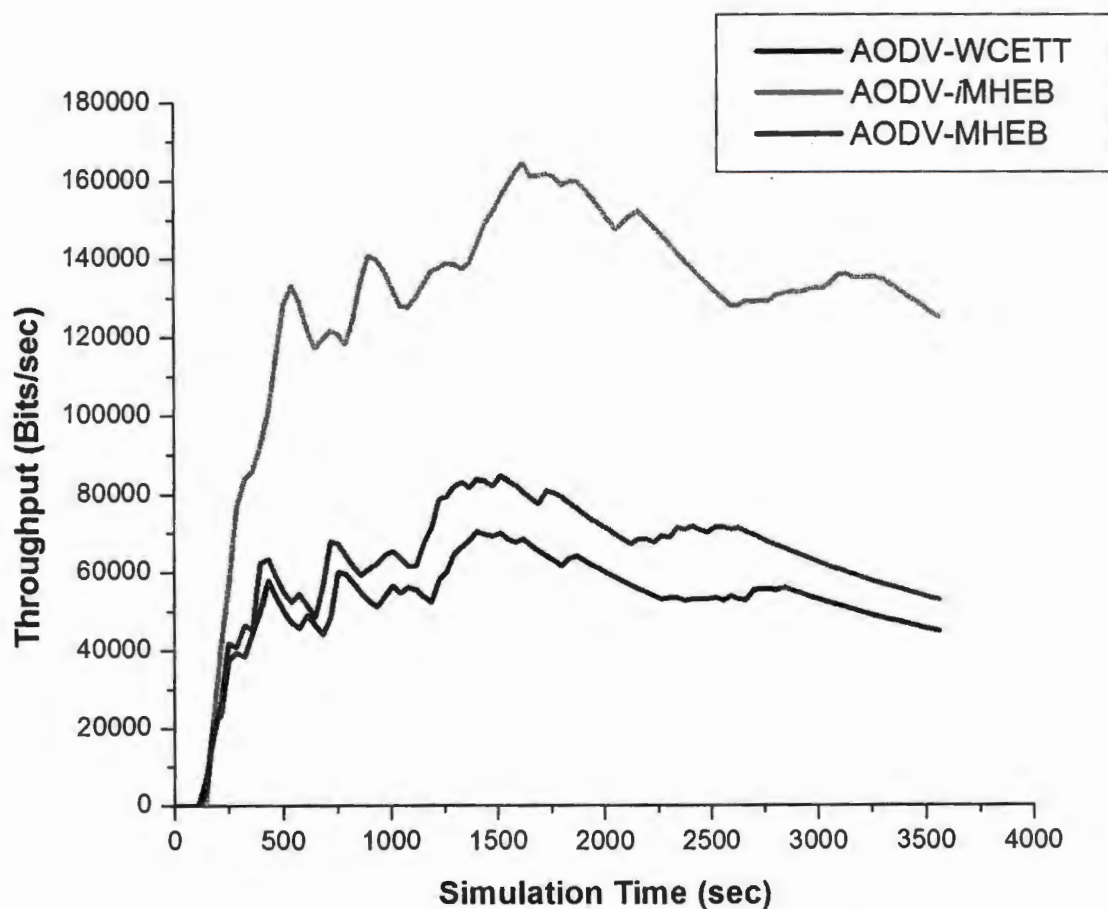


Figure 4.1: Throughput under light HTTP traffic

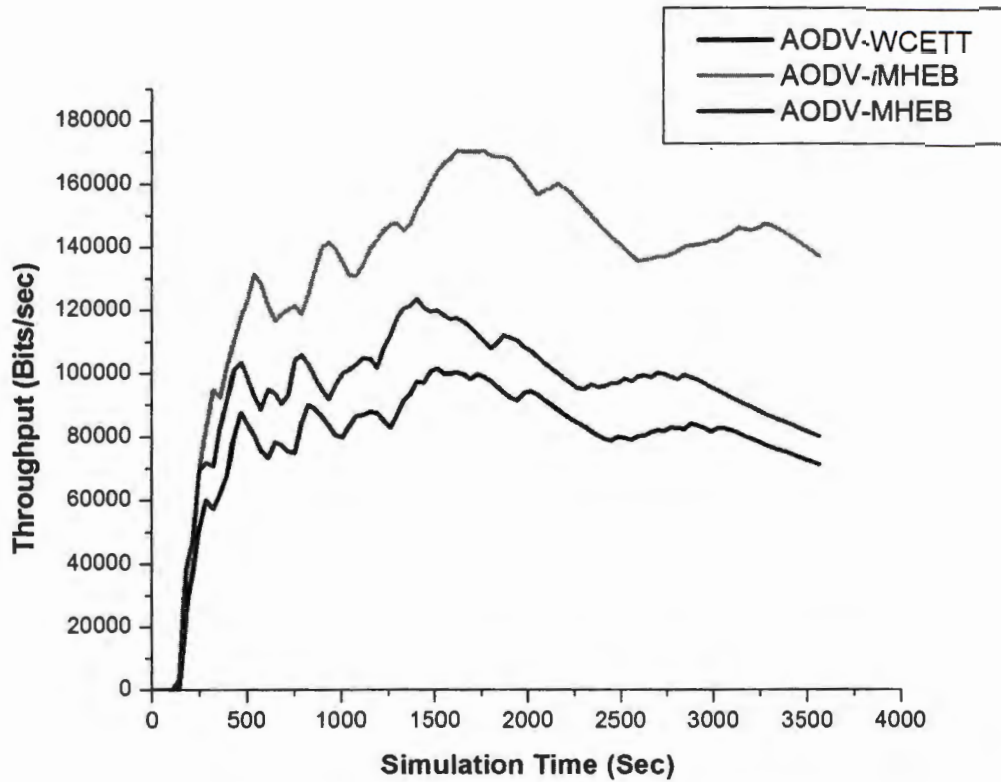


Figure 4.2: Throughput under heavy HTTP traffic

4.2.1.2 Delay Comparisons Under Light and Heavy HTTP Loads

In terms of delay, Figures 4.3 and 4.4 respectively show the delay comparison under light and heavy load of the routing protocols AODV-WCETT, AODV-MHEB and AODV-iMHEB. Under light load, Figure 4.3 shows AODV-MHEB and AODV-iMHEB routing protocols are competing for the shortest delay. But under heavy load, Figure 4.4 shows that AODV-iMHEB has a better delay than AODV-MHEB. That is due to the fact that the AODV-MHEB routing protocol under light load attempt to accurately estimate the link quality and interference of the path but under heavy load, the AODV-MHEB suffers from approximate estimations leading the protocol not to optimally reduce its delay under heavy load. Figures 4.3 and 4.4 respectively illustrate that the routing protocol experiences the longest delay under light and heavy load. This is due to the fact that the AODV-WCETT routing protocol does not choose the best route because it does not explicitly capture the effect of inter-flow interference amongst the links leading to its poor delay.

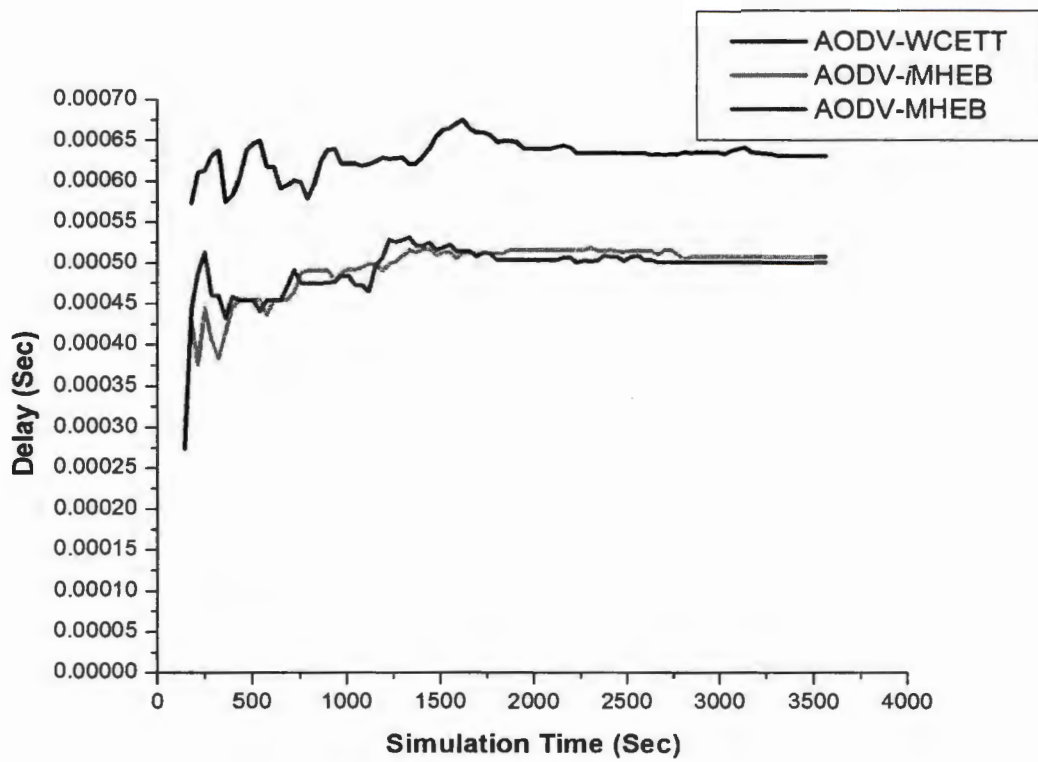


Figure 4.3: Delay under light HTTP traffic

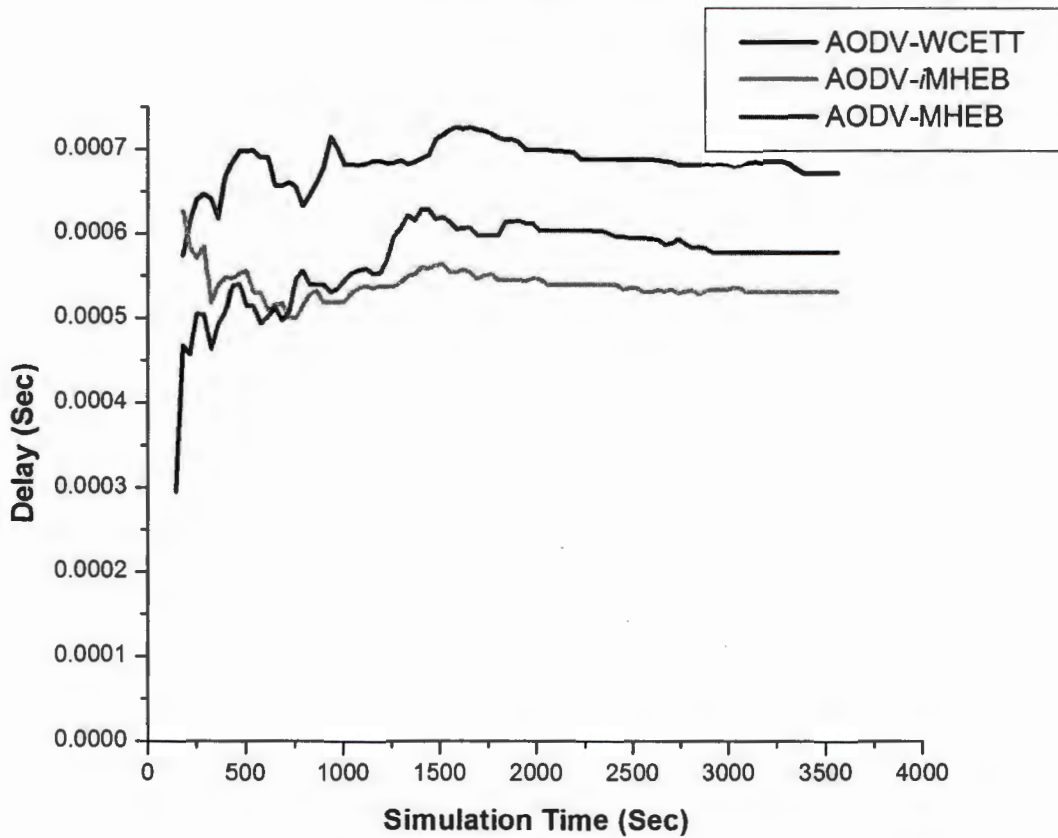


Figure 4.4: Delay under heavy HTTP traffic

4.2.1.3 Routing Overhead Comparisons Under Light and Heavy HTTP Loads

In terms of routing overhead, the performance of the AODV-*i*MHEB, AODV-MHEB and AODV-WCETT routing protocols for light and heavy load is respectively illustrated in Figures 4.5 and 4.6. The results showed that under light and heavy HTTP loads, the AODV-*i*MHEB routing protocol experiences the highest routing overhead. This is firstly due to the fact that AODV-*i*MHEB constantly sends control packets to update its routes for optimal path selection; this creates more traffic in the network resulting in high routing overhead. Secondly, the constant transmission of control packets done by the *i*MHEB routing metrics results in a longer computational time for the route selection calculation. This leads to a high routing overhead. Thirdly, the complexity of the routing metric as shown in Equation (9), causes the nodes to take longer to compute the AODV-*i*MHEB routing metric. This could also result in a high overhead.

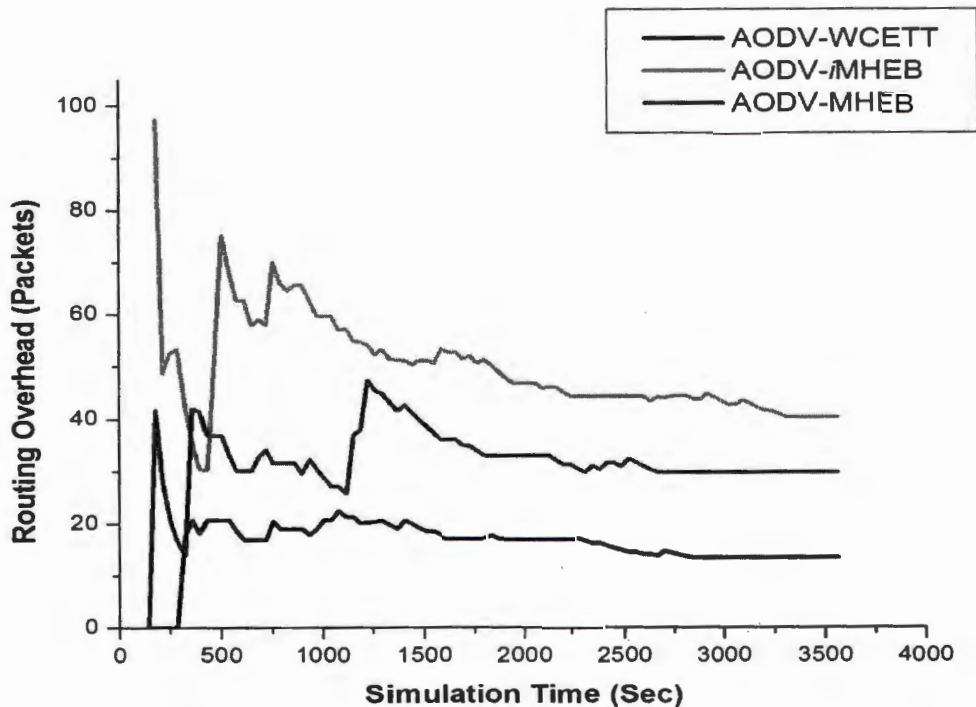


Figure 4.5: Routing overhead under light load HTTP traffic

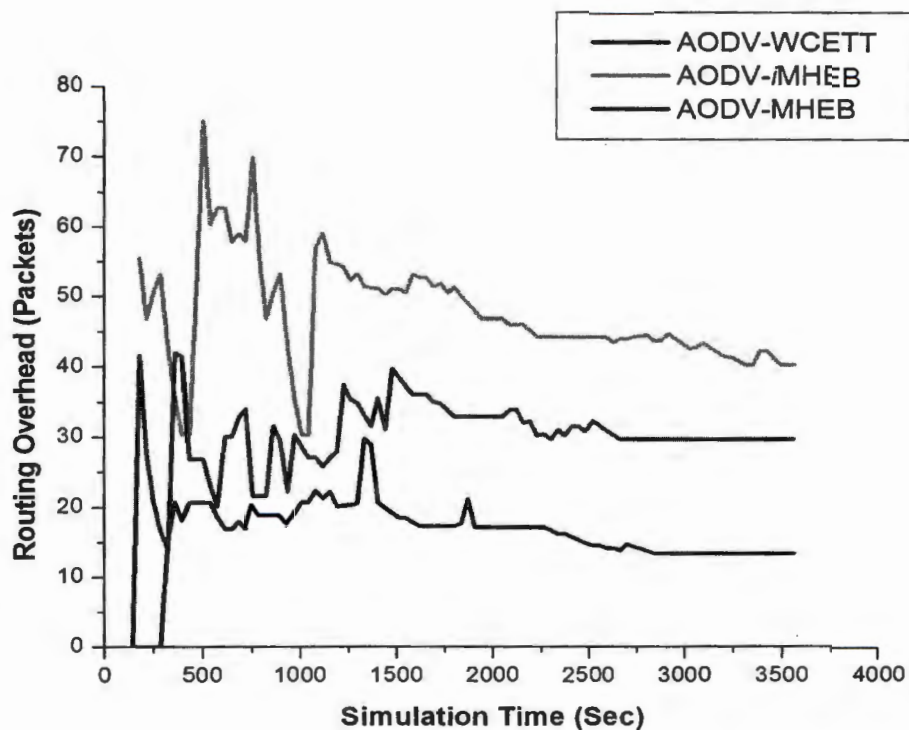


Figure 4.6: Routing overhead under heavy load HTTP traffic

4.2.2 Grid Topology

4.2.2.1 Throughput Comparisons Under Light and Heavy HTTP Loads

In terms of throughput, Figures 4.7 and 4.8 show the DSR-XiMHEB, DSR-iMHEB and DSR-EPL routing protocols under light and heavy load respectively. Figures 4.7 and 4.8 show that DSR-XiMHEB has a better throughput than DSR-iMHEB and DSR-EPL under light and heavy load respectively. This is due to fact that DSR-XiMHEB handles well multiple rates conditions because it takes into account raw data rates rather than the rates from the firmware. This also due to the fact that DSR-XiMHEB takes into account the effect of space diversity during route selection. Figures 4.7 and 4.8 also show that DSR-EPL has the lowest throughput under light and heavy load respectively; this is due to the fact that even though DSR-EPL captures the link quality for different routes, it does not capture channel diversity, leading to low throughput.

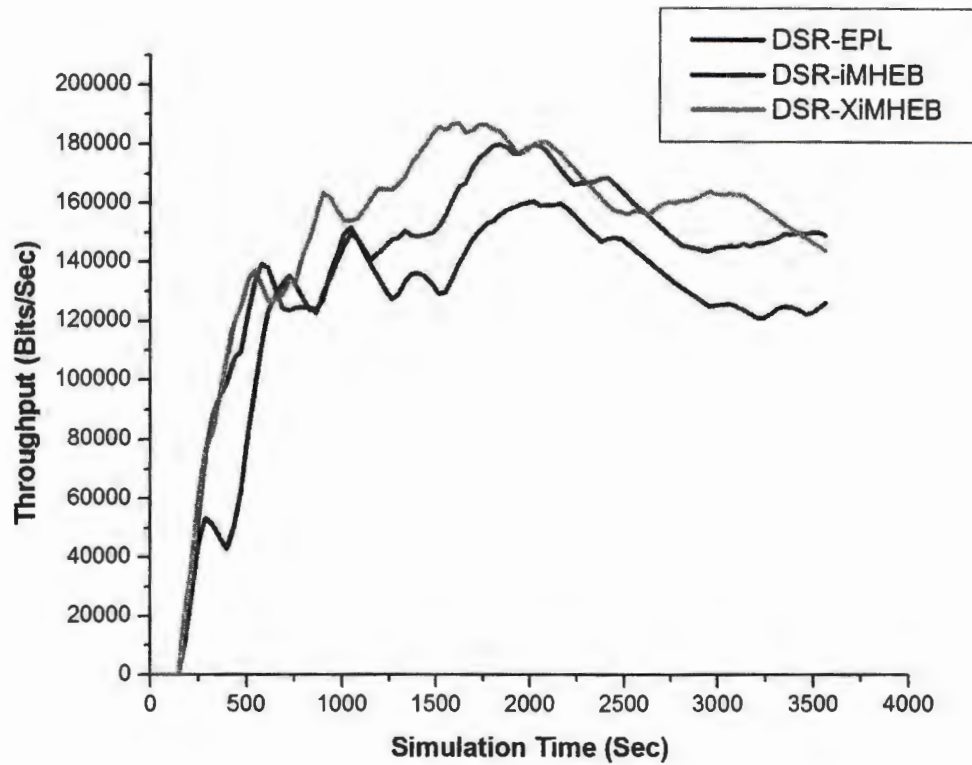


Figure 4.7: Throughput under light http load

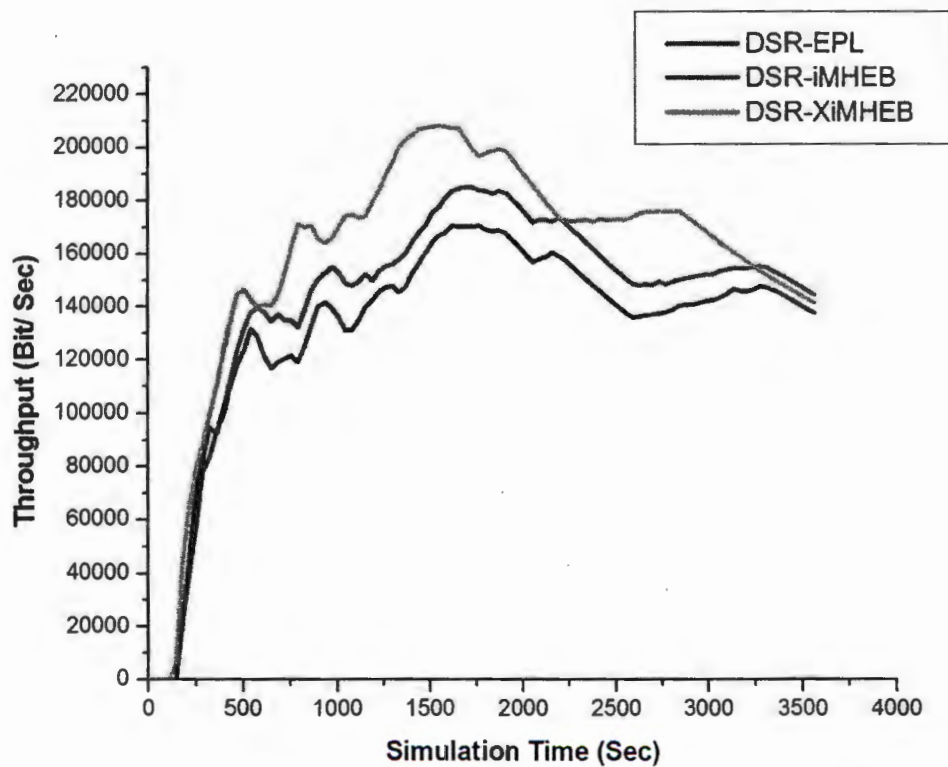


Figure 4.8: Throughput under heavy HTTP load

4.2.2.2 Delay Comparisons Under Light and Heavy HTTP Loads

In terms of delay, Figures 4.9 and 4.10 show that DSR-XiMHEB has the shortest delay. This is primarily because DSR-XiMHEB chooses routes with the least inter-flow and intra-flow interferences. It chooses the paths with the best link quality; thus leading to minimal delay during transmission between source a node to a destination node. This is also due to the fact that DSR-XiMHEB efficiently evaluates the expected MAC layer transmission time for successful sending of packet from the source node to destination node. The better delay experienced by DSR-XiMHEB with respect to DSR-iMHEB and DSR-EPL could also be due to the fact that DSR-XiMHEB exploits the channel diversity more efficiently than DSR-iMHEB and DSR-EPL.

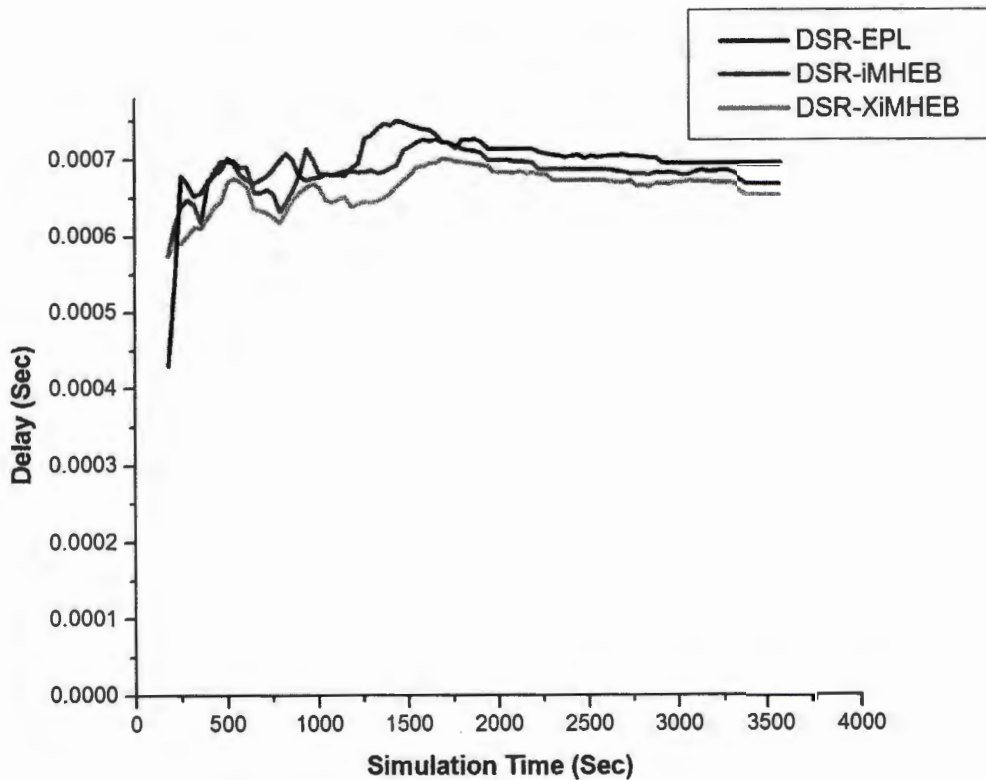


Figure 4.9: Delay under light load

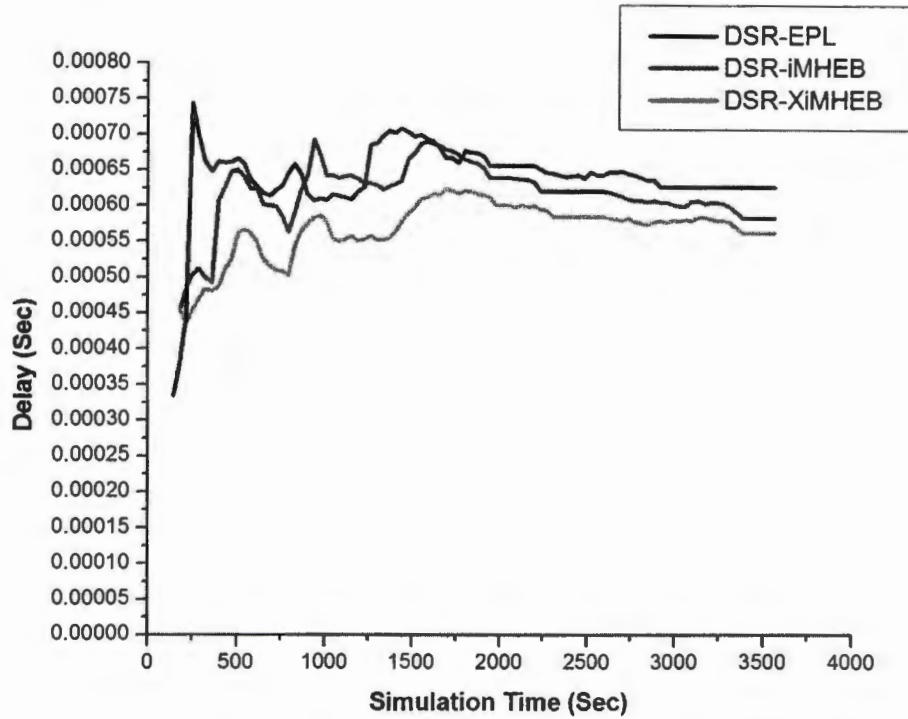


Figure 4.10: Delay under heavy load

4.2.2.3 Routing Overhead Comparisons Under Light and Heavy HTTP Loads

In terms of routing overhead, Figures 4.8 and 4.9 show that DSR-XiMHEB has the highest routing overhead. This is due to the fact that, the DSR-XiMHEB routing protocol sends more control packets to update its routes for accurate route selection than the DSR-iMHEB and DSR-EPL; this leads to more traffic being created in the network resulting in high routing overhead. This is also due the fact, repeated broadcast of control packets performed by the DSR-XiMHEB routing protocol results in a longer computational and processing time for the route selection calculation and estimation than the DSR-iMHEB and DSR-EPL. This leads to a high routing overhead. The complexity of the routing metric as shown in Equation (13), causes the nodes to take longer to compute the XiMHEB routing metric than the EPL and iMHEB routing metrics. This also results in a high overhead.

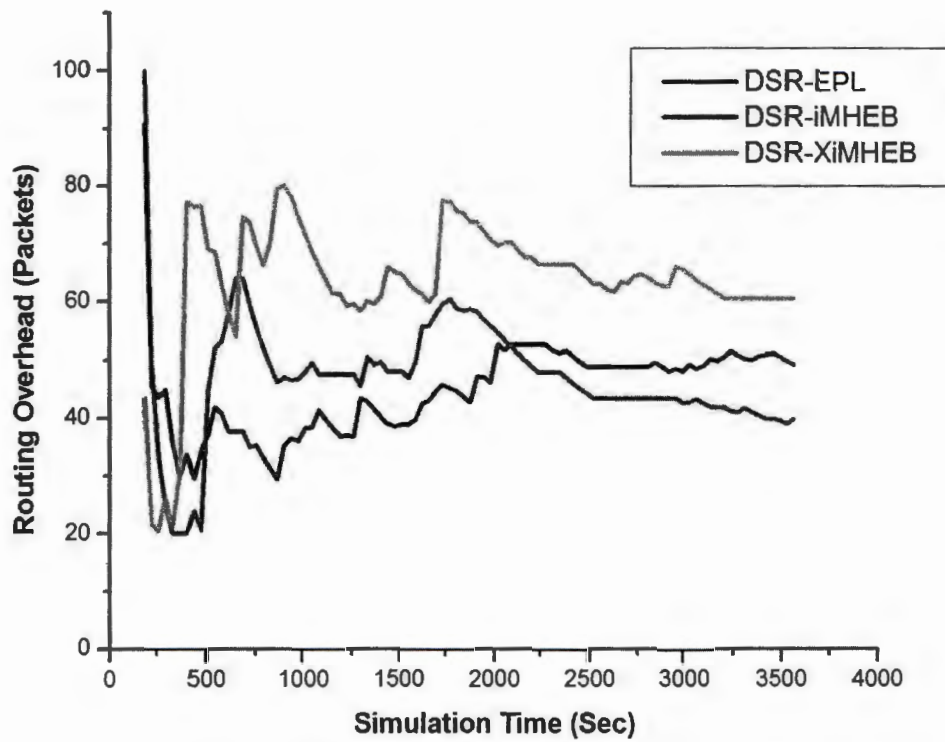


Figure 4.11: Routing Overhead under low load

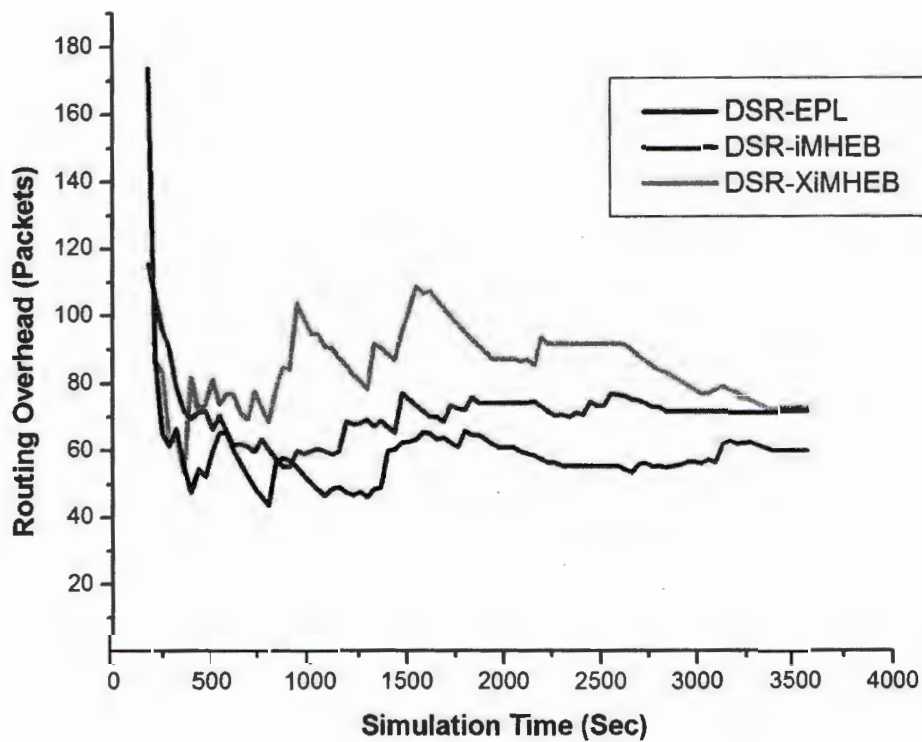


Figure 4.12: Routing Over head under heavy load

4.3 Chapter Summary

This chapter presented the results and performance evaluation based on the throughput, delay and routing overhead obtained from the simulation. The results show that the proposed routing metric *i*MHEB and *Xi*MHEB performed better than the other routing metrics. From the random topology, the overall results show that the AODV-*i*MHEB routing protocol has a better throughput and better delay than the AODV-WCETT and AODV-MHEB routing protocols. However, it experiences the worst routing overhead. From the grid topology, the overall results show that the DSR-*Xi*MHEB has better throughput and delay but worst routing overhead than the DSR-*i*MHEB and DSR-EPL routing protocols.

CHAPTER 5

SUMMARY, CONCLUSIONS AND FUTURE WORK

5.1 Summary

WMNs are a special case of wireless multi-hop ad-hoc networks with the possibility of playing a critical role as a substitute tool for broadband communication due to their unique characteristics of providing multi-hop routing, reliable services, self-healing, easy maintenance, self-configuration, self-organization, high scalability, bandwidth fairness, low cost, easy deployment, and because they can deliver wireless services for a wide range of applications. WMNs are analogous to Ad-Hoc networks in terms of wireless multi-hop communication. Traditional routing protocols such as AODV and DSR applied in Ad Hoc Networks do not perform well in WMN environments, due to the fact that these protocols used the minimum hop count routing metric for route selection. Minimum hop count metric does not consider the fact that Mesh routers are equipped with multiple network interfaces cards have the possibility of generating interferences during the functioning of the WMN. Moreover, minimum hop count does not take into consideration the link quality during the best path selection. This research therefore reports on ways of improving conventional routing protocols (AODV and DSR) by enhancing them to efficiently perform in WMNs through designing routing metrics which consider link quality and interferences during the route selection processes.

The main goal of this research was to design, implement and simulate novel Wireless Mesh Routing protocols with link quality and interference awareness, for optimal performance improvement of typical routing protocols such as AODV and DSR used in WMNs. This goal was achieved by realizing the three key objectives as specified in chapter one through:

- (i) Designing routing metrics which take into consideration the link quality and the interference in WMNs.

Consequently, two link quality and interferences aware routing metrics were designed namely; the *i*MHEB and the XiMHEB routing metrics. The *i*MHEB routing metric uses the asymmetry of packets to improve the conventional MHEB. The *i*MHEB compute an improved Achievable Bandwidth under Inter-Flow Interference (*i*ABITF) by biasing the weight towards the forward link. That is, assigning a greater weight to the forward link in order to reduce the asymmetry of packets during transmission from a source node to destination node. The XiMHEB routing metric is a hybrid metric which extents the *i*MHEB by taking into account the data rate transmission and the raw data packets size to improve the *i*MHEB routing metric. The XiMHEB routing metric also considers the actual time spent during transmission from a source node to a destination node to enhance the *i*MHEB routing metric.

- (ii) Implementing the designed routing metrics into existing routing protocols, resulting to novel routing protocols.

As a proof of concept, the designed routing metrics were implemented into existing routing protocols resulting to novel routing protocols. The *i*MHEB routing metric was implemented into the reactive AODV routing protocol resulting in a novel link quality and interference aware routing protocol named *i*MHEB-AODV. Additionally, the XiMHEB routing metric was also implemented into the DSR routing protocol leading to a new routing protocol called DSR-XiMHEB.

- (iii) Comparing the performance of the novels routing protocols with other routing protocols

As a comparative analysis, and to validate the results obtained from the two novel routing protocols developed, comparison of the performance in terms of throughput, delay and routing overhead of the novel protocols with others

routing protocols was carried out. The results obtained from simulation shows that, AODV-iMHEB significantly performs better in terms of throughput and delay than the routing protocols AODV-WCETT and AODV-MHEB under light and heavy HTTP loads. The novel DSR-XiMHEB routing protocol also performs better than the DSR-iMHEB and DSR-EPL routing protocols under light and heavy HTTP loads in terms of throughput and delay.

5.2 Concluding Remarks

Within the scope of the main goal of this research and reported in this thesis is the development of two novel link quality and interference aware routing protocols for hybrid multi-channel, multi-radio and multi-rate WMNs.

Conclusively, we introduced the background information for this research, pointing out the main goal and its associated objectives as well as the methodology used to achieve this goal.

Moreover, an overview of existing works in the area of WMNs and routing protocols was explored, surveyed and review.

Consequently, we reported on how the specified objectives to achieve the main goal are practically implemented. We also explained the experimental design process as well as the simulation setup.

Finally, we reported on the results obtained from the simulation and discussed the results in relation to existing research, while showing the contributing approach achieved by this research to already existing works.

As concluding and contributory remarks, this research endeavors to provide networks engineers and the research community the opportunity to utilize the novel routing protocols in improving networks functionalities and research.

The results obtained and reported in this thesis are encouraging to warrant further focus.

5.3 Future Work

The most imminent future work of this research would be:

- (i) An attempt to minimize the complexity of the routing metric to reduce the high routing overhead experienced by the AODV-*i*MHEB and DSR-*Xi*MHEB routing protocols.
- (ii) To implement *i*MHEB and *Xi*MHEB into a proactive routing protocol such as OLSR.
- (iii) To implement the *i*MHEB and *Xi*MHEB routing metrics in very fast networks and using different mobility models such Reference Point Group Mobility model or Manhattan mobility Model.

References

- [1] A. H. Mogaibel and M. Othman. "Review of Routing Protocols and it's Metrics for Wireless Mesh Networks," in *Proc. IACSIT Spring Conference*, 2009, pp. 62-67.
- [2] S.Khan, A. A. Pirzada, M.Portmann. "Performance Comparison of Reactive Routing Protocols for Hybrid Wireless Mesh Networks," in *Proc. 2nd International Conference on Wireless Broadband and Ultra Wideband Communications (AusWireless)*, 2008, pp. 78-84.
- [3] I. F. Akyildiz, X. Wang, W. Wang. "Wireless mesh networks: a survey." *Computer Networks*, vol. 47, pp. 445–487, 2005.
- [4] M. Ramadhan and M.Davis. "Performance Study of a Cross-layer Optimization to the DSR Routing Protocol in Wireless Mesh Networks," in *Proc. International Conference on Network and Service Security*, 2009, pp. 1-5.
- [5] J. Jun and M. L.Sichitiu. "MRP: Wireless mesh networks routing protocol." *Computer Communications*, vol.31, pp. 1413–1435, 2008.
- [6] C. Y. Bian, X. Jin, C. Liu, X. M. Li, Y. Wei. "Relative Link Quality Assessment and Hybrid Routing Scheme for Wireless Mesh Networks," in *Proc. IEEE ICC*, 2011, pp. 1-6.
- [7] M. Kim, I. Ra, J. Yoo, D. Kim, H. Kim. "QoS Mesh Routing Protocol for IEEE 802.16 based Wireless Mesh Networks," in *Proc. ICACT*, 2008, pp. 812 - 817.
- [8] A. Zakrzewska, L. Koszałka, I. Poźniak-Koszałka. "Performance Study of Routing Protocols for Wireless Mesh Networks," in *Proc. 19th International Conference on Systems Engineering*, 2008, pp. 331 - 336.
- [9] P. H. Pathak and R. Dutta. "A Survey of Network Design Problems and Joint Design Approaches in Wireless Mesh Networks." *IEEE Communications Surveys & Tutorials*, vol. 13, pp. 396 – 428, 2011.
- [10] U.Ashraf, S.Abdellatif, G.Juanole. "An Interference and Link-Quality Aware Routing Metric for Wireless Mesh Networks," in *Proc. Vehicular Technology Conference*, 2008, pp. 1-5.
- [11] M. Catalan-Cid, J. L. Ferrer, C. Gomez, J. Paradells. "Contention and Interference-Aware Flow-Based Routing in Wireless Mesh Networks: Design and Evaluation of a Novel Routing Metric." *Journal on Wireless Communications and Networking*, vol. 2010, pp. 1-20, 2010.
- [12] Y. Yang, J. Wang, R. Kravets. "Designing Routing Metrics for Mesh Networks," in *Proc. IEEE Workshop on Wireless Mesh Networks (WiMesh)*, 2005, pp. 1-9.

- [13] M.E.M., Campista, et al. "Routing Metrics and Protocols for Wireless Mesh Networks." *IEEE Networks*, vol. 22, p.p 6-12, 2008.
- [14] J. L. Sobrinho. "Algebra and algorithms for QoS path computation and hop-by-hop routing in the Internet." *IEEE/ACM Transactions on Networking*, vol. 10, pp. 541 – 550, 2002.
- [15] D. S. J. De Couto, D. Aguayo, J. Bicket, R. Morris. "A High-Throughput Path Metric for Multi-Hop Wireless Routing," in *Proc. 9th annual international conference on Mobile computing and networking*, 2003, pp. 134-146.
- [16] S. Roy, D. Koutsonikolas, S. Das, Y. C. Hu. "High-throughput multicast routing metrics in wireless mesh networks." *Ad Hoc Networks* vol. 6, pp. 878-899, 2008.
- [17] S. Kim, O. Lee, S. Choi, S.J. Lee. "Comparative analysis of link quality metrics and routing protocols for optimal route construction in wireless mesh networks". *Ad Hoc Networks*, vol. 9, pp. 1343-1358, 2011.
- [18] S. Lee et al. "A Cross Layer Routing Metric To Recognize Traffic Interference in Wireless Mesh Networks," In *Proc. IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*, 2010, pp. 1-6.
- [19] R. Draves, J. Padhye, B. Zill. "Routing in Multi-Radio, Multi-Hop Wireless Mesh Networks," in *Proc. 10th annual international conference on Mobile computing and networking*, 2004, pp. 114-128.
- [20] F. Li, Y. Fang, F. Hu, X. Liu. "Load-aware multicast routing metrics in multi-radio multi-channel wireless mesh networks." *Computer Networks*, vol. 55, pp. 2150-2168, 2011.
- [21] V. Kawadia and P.R. Kumar. "A Cautionary perspective on Cross Layer Design." *IEEE Wireless Communication*, 2005.
- [22] F. Aune. "Cross-Layer Design Tutorial." Norwegian University of Science and Technology, Dept. of Electronics and Telecommunications, Trondheim, Norway, 26.11. 2004. Available at: <http://www.iet.ntnu.no/projects/cuban/archive/1812041.pdf>.
[Last accessed on: 16 July 2012].
- [23] H. Li, Y. Cheng, C. Zhou. "Multi-Hop Effective Bandwidth Based Routing in Multi-Radio Wireless Mesh Networks," In *Proc GLOBECOM*, 2008, pp 1-6.

- [24] F.De Rango, J. C.Cano, M.Fotino, C.Calafate, P. Manzoni, S. Marano."OLSR vs DSR: A comparative analysis of proactive and reactive mechanisms from an energetic point of view in wireless ad hoc networks." *Computer Communications*, vol. 31,pp. 3843–3854, 2008.
- [25] E. Perkins, S. Belding-Royer, Das, "Ad hoc On-Demand Distance Vector (AODV) Routing." Network Working Group, Request for comments: 3561. (2003).
- [26] E.M. Belding-Royer and C. E. Perkins." Evolution and future directions of the ad hoc on-demand distance-vector routing protocol." *Ad Hoc Networks*, vol. 1,pp. 125–150, 2003.
- [27] J. Khan. "Based on MANET the Correctness and Efficiency Evaluation of DSR, AODV and TORA Routing Protocols for Best QoS." *International Journal of Computer Applications* Special Issue on "Mobile Ad-hoc Networks", pp.111-119, 2010.
- [28] A. Bamis, A. Boukerche, I. Chatzigiannakis, S. Nikolettseas. "A mobility aware protocol synthesis for efficient routing in ad hoc mobile networks." *Computer Networks*, vol. 52, pp. 130–154, 2008
- [29] D. Johnson, Y. Hu, D. Maltz, "The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4." Network Working Group, Request for comments: 4728, 2007
- [30] A K. Gupta, H. Sadawarti, A K.Verma."Performance analysis of AODV, DSR & TORA Routing Protocols." *International Journal of Engineering and Technology*, vol.2, pp. 226-231, 2010.
- [31] V. Park, and S. Corson. " Temporally-Ordered Routing Algorithm (TORA) Version 1 Functional Specification." IETF MANET Working Group, INTERNET-DRAFT, draft-ietf-manet-tora-spec-04.txt
- [32] Z. Guo, S. Malakooti, S. Sheikh, C. Al-Najjar, B. Malakooti. "Multi-objective OLSR for proactive routing in MANET with delay, energy, and link lifetime predictions." *Applied Mathematical Modeling*, vol.35, pp.1413-1426, 2011.
- [33] T. Clausen and P. Jacquet, Optimized Link State Routing Protocol (OLSR). Network Working Group, Request for Comments: 3626
- [34] Opnet Modeler 14 Documentation.

[35] M. Mbougni and O.O Ekabua. “ Towards Improved Multi Hop Effective Bandwidth Routing Metric in Wireless Mesh Networks,” in *Proc. Southern Africa Telecommunication Networks and Application Conference (SATNAC)*, 2012, pp.151-156.

[36] Niranjana, S. Pandey, A. Ganz. “Design and Evaluation of Multichannel Multirate Wireless Networks.” *Mobile Networks and Applications*, vol. 11, pp. 697–709, 2006.

[37] M. Thaker. “Performance Evaluation and Design Improvement of Media Access Control Protocols for Broadband Wireless Local Loop.” Master thesis presentations, Department of Electrical Engineering and Computer Science, University of Kansas, Kansas, USA , July 19, 2000. Available at:

“http://www.ittc.ku.edu/research/thesis/documents/mihir_thaker.pdf. [Last accessed on: 24 February 2012].

Appendix A

Comment: Source Code for the main functions used for the process model implementation of the AODV_iMHEB routing protocol in OPNET 14.0

```
#line 1 "C:/Program Files/OPNET/14.0.A/models/std/Michel Mbougni/aodv_imheb_rte.pr.c"
/* Process model C form file: aodv_imheb_rte.pr.c */
/* Portions of this file copyright 1992-2007 by OPNET Technologies, Inc. */
/* Author: Michel Mbougni

/* This variable carries the header into the object file */
const char aodv_imheb_rte_pr_c [] = "MIL_3_Tfile_Hdr_140A 30A op_runsim 7 4FB2E747
4FB2E747 1 mike-HP mike 0 0 none none 0 0 none 0 0 0 0 0 0 0 18a9 3
",
#include <string.h>

/* OPNET system definitions */
#include <opnet.h>

/* Header Block */

/* Includes */
#include <ip_addr_v4.h>
#include <ip_rte_support.h>
#include <ip_rte_v4.h>
#include <ip_dgram_sup.h>
#include <ip_higher_layer_proto_reg_sup.h>
#include <ip_cmn_rte_table.h>
#include <oms_dist_support.h>
#include <aodv_imheb.h>
#include <aodv_imheb_pkt_support.h>
#include <aodv_imheb_ptypes.h>
#include <oms_pr.h>

#define PACKET_ARRIVAL (invoke_mode == OPC_PROINV_INDIRECT)
#define HELLO_TIMER_EXPIRY ((invoke_mode == OPC_PROINV_DIRECT)
&& (intrpt_type == OPC_INTRPT_SELF) && (intrpt_code ==
AODVC_HELLO_TIMER_EXPIRY))

/* Traces */
#define LTRACE_ACTIVE (op_prg_odb_ltrace_active("aodv_imheb_rte"))

/* Prototypes */
static void aodv_imheb_rte_sv_init(void);
static void aodv_imheb_rte_pkt_arrival_handle(void);
static void aodv_imheb_rte_app_pkt_arrival_handle(Packet*,
IpT_Dgram_Fields*, Boolean);
```

```

static void      aadv_imheb_rte_rreq_pkt_arrival_handle (Packet*, Packet*,
                                                         IpT_Dgram_Fields*, IpT_Rte_Ind_Ici_Fields*,
                                                         AadvT_Packet_Option*);
static void      aadv_imheb_rte_rrep_pkt_arrival_handle (Packet*, Packet*,
                                                         IpT_Dgram_Fields*, IpT_Rte_Ind_Ici_Fields*,
                                                         AadvT_Packet_Option*);
static void      aadv_imheb_rte_rerr_pkt_arrival_handle (Packet*, Packet*,
                                                         IpT_Dgram_Fields*, IpT_Rte_Ind_Ici_Fields*,
                                                         AadvT_Packet_Option*);

/* End of Header Block */

/* State variable definitions */
typedef struct
{
/* Internal state tracking for FSM */
    FSM_SYS_STATE
/* State Variables */
    IpT_Rte_Module_Data*   module_data_ptr ; /* Module Data Memory for the IP module*/
    Objid   own_mod_objid ;    /* Objid of this module */
    Objid   own_node_objid ;    /* Objid of this node */
    Prohandle   own_prohandle ;    /* Own process handle */
    Prohandle   parent_prohandle ; /* Process handle of the parent process */
    AadvT_Route_Table* route_table_ptr ; /* Pointer to the AADV_iMHEB route table
                                         where the routes to all known */
                                         /*destinations are stored.*/
    AadvT_Packet_Queue* pkt_queue_ptr ; /* Pointer to the packet queue where the data
    packets are buffered while a route to */
                                         /* the destination is discovered */
}aadv_imheb_rte_state;

/* Function Block */
#if !defined (VOSD_NO_FIN)
enum { _op_block_origin = __LINE__ + 2};
#endif

static void aadv_imheb_rte_sv_init (void)
{
    /** Initialize the state variables **/
#ifdef BIN
#define BIN
#endif
#ifdef BOUT
#define BOUT
#endif
#line 298
FIN (aadv_imheb_rte_sv_init (void));

```

```

/* Access the module data memory */
module_data_ptr = (IpT_Rte_Module_Data*) op_pro_modmem_access ();

/* Obtain own module ID */
own_mod_objid = op_id_self ();

/* Obtain the node's objid. */
own_node_objid = op_topo_parent (own_mod_objid);

/* Obtain own process handle. */
own_prohandle = op_pro_self ();

/* Obtain parent process handle */
parent_prohandle = op_pro_parent (own_prohandle);

/* Own process ID */
own_pro_id = op_pro_id (own_prohandle);

/* Parent process ID */
parent_pro_id = op_pro_id (parent_prohandle);

/* Set up the display string. */
sprintf (pid_string, "aodv_rte PID (%d)", own_pro_id);

/* Initialize the identification values */
route_request_id = 0;
sequence_number = 0;

/* Initialize the variables used to keep track of the rate */
last_route_error_sent_time = 0.0;
num_route_errors_sent = 0;
last_route_request_sent_time = 0.0;
num_route_requests_sent = 0;
last_broadcast_sent_time = 0.0;

/* Obtain a handle to the global statistics*/
global_stat_handle_ptr = aodv_support_global_stat_handles_obtain ();

/* Assign a unique name to this AODV_iMHEB process. */
Aodv_imheb_proto_id = IP_CMN_RTE_TABLE_UNIQUE_ROUTE_PROTO_ID
(IPC_DYN_RTE_AODV, IPC_NO_MULTIPLE_PROC);
Ip_Cmn_Rte_Table_Install_Routing_Proc (module_data_ptr->ip_route_table, aodv_proto_id,
own_prohandle);

/* Create a /32 subnet mask for entries in the route table */
subnet_mask = inet_smask_from_length_create (32);

/* Create a /128 subnet mask for entries in the route table */
subnet_mask_ipv6 = inet_smask_from_length_create (128);

/* External routes are handled only by gateways. */

```

```

if(ip_manet_gateway_enabled(module_data_ptr))
{BIN{
/* Initialize the cache table used to store routes to */
/* external hosts (non- hosts). */
external_routes_cache_table_ptr = manet_ext_rte_cache_table_init();
}BOUT}

FOUT;
}

/***** PACKET ARRIVAL FUNCTIONS *****/

static void aodv_imheb_rte_pkt_arrival_handle (void)
{
Packet*      ip_pkptr;
Packet*      aodv_imheb_pkptr;
IpT_Rte_Ind_Ici_Fields*  intf_ici_fdstruct_ptr;
IpT_Dgram_Fields*        ip_dgram_fd_ptr;
AodvT_Packet_Option*      tlv_options_ptr;
char                    addr_str [INETC_ADDR_STR_LEN];
char                    node_name [OMSC_HNAME_MAX_LEN];
char                    temp_str [256];
ManetT_Info*            info_ptr;
IpT_Interface_Info*      input_intf_info_ptr;

/* A packet has arrived. Handle the packet appropriately based on its type */
FIN (aodv_rte_pkt_arrival_handle (void));

* Access the argument memory to get the packet pointer. */

info_ptr = (ManetT_Info*) op_pro_argmem_access();
ip_pkptr = (Packet*) info_ptr->manet_info_type_ptr;

if (ip_pkptr == OPC_NIL)
aodv_imheb_rte_error ("Could not obtain the packet from the argument memory", OPC_NIL,
OPC_NIL);

/* Check if the packet came from the application */
if ((intf_ici_fdstruct_ptr->instrm == module_data_ptr->instrm_from_ip_encap) ||
(intf_ici_fdstruct_ptr->instrm == IpC_Pk_Instrm_Child) ||
(ip_manet_gateway_enabled(module_data_ptr)
&&!ip_rte_intf_manet_enabled(input_intf_info_ptr)))
{BIN{
if (LTRACE_ACTIVE)
{BIN{
inet_address_print (addr_str, ip_dgram_fd_ptr->dest_addr);
inet_address_to_hname (ip_dgram_fd_ptr->dest_addr, node_name);
sprintf (temp_str, "An application packet has arrived from the higher layer to destination %s
(%s).", addr_str, node_name);
op_prg_odb_print_major (pid_string, temp_str, "No route exists to the destination.",
OPC_NIL);

```

```

}BOUT}

/* This is a packet that came from the application (higher) layer and no */
/* route exists to the destination of this packet. Handle the arrival */
aodv_rte_app_pkt_arrival_handle (ip_pkptr, ip_dgram_fd_ptr, OPC_TRUE);

FOUT;}}

/* If this is an AODV_imheb control packet, get the AODV_iMHEB */
/* packet from the IP datagram */
op_pk_nfd_get_pkt (ip_pkptr, "data", &aodv_imheb_pkptr);

/* Get the option in the AODV_iMHEB packet */
op_pk_nfd_access (aodv_imheb_pkptr, "Options", &tlv_options_ptr);

/* If this is an AODV_imheb signaling packet. Based on the */
/* type of option set in the packet, handle it */
switch (tlv_options_ptr->type)
{
case (AODV_iMHEB_ROUTE_REQUEST):
{BIN{
/* This is a route request packet */
Aodv_imheb_rte_rreq_pkt_arrival_handle (ip_pkptr, aodv_pkptr, ip_dgram_fd_ptr,
intf_ici_fdstruct_ptr, tlv_options_ptr);

{ BOUT break; }
}}

case (AODV_iMHEB_ROUTE_REPLY):
{BIN{
/* This is a route reply option */
Aodv_imheb_rte_rrep_pkt_arrival_handle (ip_pkptr, aodv_pkptr, ip_dgram_fd_ptr,
intf_ici_fdstruct_ptr, tlv_options_ptr);

{ BOUT break; }
}}

case (AODV_iMHEB_ROUTE_ERROR):
{BIN{
/* This is a route error option */
Aodv_imheb_rte_rerr_pkt_arrival_handle (ip_pkptr, aodv_imheb_pkptr, ip_dgram_fd_ptr,
intf_ici_fdstruct_ptr, tlv_options_ptr);

{ BOUT break; }
}}

case (AODV_iMHEB_HELLO):
{BIN{
/* This is a RREP Hello packet */

```

```

Aadv_imheb_rte_rrep_hello_pkt_arrival_handle (ip_pkptr, aadv_pkptr, ip_dgram_fd_ptr,
intf_ici_fdstruct_ptr, tlv_options_ptr);

{ BOUT break; }
}}

default:
{BIN{
/* Invalid option in packet */
Aadv_imheb_rte_error ("Invalid Option Type in AODV_iMHEB packet", OPC_NIL,
OPC_NIL); }BOUT}}
FOUT;}}

static void aadv_imheb_rte_app_pkt_arrival_handle (Packet* ip_pkptr, IpT_Dgram_Fields*
ip_dgram_fd_ptr, Boolean pkt_from_app_layer)
{
int    ttl_value;
int    last_known_imheb;
double ring_traversal_time;
char   addr_str [INETC_ADDR_STR_LEN];
char   node_name [OMSC_HNAME_MAX_LEN];
char   temp_str [256];
AadvT_Route_Entry* route_entry_ptr = OPC_NIL;

FIN (aadv_rte_app_pkt_arrival_handle (<args>));
/* Check if there exists an invalid route for this destination, if yes update it appropriately */
route_entry_ptr = aadv_route_table_entry_get (route_table_ptr, ip_dgram_fd_ptr->dest_addr);

/* Update the expiry time of the route to be delete period */
if ((route_entry_ptr != OPC_NIL) && (route_entry_ptr->route_entry_state
==AadvC_Invalid_Route))
{BIN{
Aadv_imheb_route_table_entry_expiry_time_update (route_entry_ptr, ip_dgram_fd_ptr-
>dest_addr, delete_period, AODV_iMHEB_ROUTE_ENTRY_EXPIRED);
}BOUT}
if (pkt_from_app_layer == OPC_FALSE)
{BIN{
if (LTRACE_ACTIVE)
{BIN{
inet_address_print (addr_str, ip_dgram_fd_ptr->src_addr);
inet_address_to_hname (ip_dgram_fd_ptr->src_addr, node_name);
sprintf (temp_str, "An application packet has arrived from source node %s (%s)", addr_str,
node_name);
op_prg_odb_print_major (pid_string, temp_str, OPC_NIL);
}BOUT}
/* If this packet came from a neighboring node, update the connectivity table */
Aadv_imheb_rte_neighbor_connectivity_table_update (ip_dgram_fd_ptr->src_addr,
OPC_FALSE);

/* Change the status of the external route, it is now not available. */
ext_cache_entry_ptr->available = OPC_FALSE;

```

```

/* Send the RERR for the failed external route.          */
Aadv_imHEB_rte_ext_route_error_send (ip_dgram_fd_ptr->dest_addr, ext_cache_entry_ptr-
>dest_seq_num);

/* Destroy the data packet.          */
manet_rte_ip_pkt_destroy (ip_pkptr);

/* Update the number of packets dropped statistic          */
op_stat_write (local_stat_handle_ptr->num_pkts_discard_shandle, 1.0);
op_stat_write (global_stat_handle_ptr->num_pkts_discard_global_shandle, 1.0);
FOUT; }}

static void aadv_imheb_rte_rreq_pkt_arrival_handle (Packet* ip_pkptr, Packet*
aadv_imheb_pkptr, IpT_Dgram_Fields* ip_dgram_fd_ptr, IpT_Rte_Ind_Ici_Fields*
intf_ici_fdstruct_ptr, AadvT_Packet_Option* tlv_options_ptr)
{
AadvT_Rreq*          rreq_option_ptr;
InetT_Address        prev_hop_addr;
AadvT_Route_Entry*   route_entry_ptr = OPC_NIL;
AadvT_Route_Entry*   rev_path_entry_ptr = OPC_NIL;
IpT_Interface_Info*  iface_elem_ptr;
IpT_Port_Info        in_port_info;
double               min_lifetime, lifetime;
double               imheb;
int                  dest_seq_num, new_ttl_value;
Packet*              ip_rreq_pkptr;
char                 src_node_name [OMSC_HNAME_MAX_LEN];
char                 src_hop_addr_str [INETC_ADDR_STR_LEN];
char                 dest_node_name [OMSC_HNAME_MAX_LEN];
char                 dest_hop_addr_str [INETC_ADDR_STR_LEN];
char                 temp_str [2048];
double               existing_lifetime;
AadvC_Route_Entry_State route_entry_state;
Boolean              dest_seq_flag = OPC_FALSE;
InetT_Subnet_Mask    selected_subnet_mask;
Ici*                 ip_iciptr;
int                  mcast_major_port = IPC_MCAST_ALL_MAJOR_PORTS;
Manet_Rte_Ext_Cache_entry* ext_cache_entry_ptr = OPC_NIL;

/** Handles the arrival of a route request message      */
FIN (aadv_imheb_rte_rreq_pkt_arrival_handle (<args>));
/* if a route request packet has arrived at this node get the request options from the packet*/

rreq_option_ptr = (AadvT_Rreq*) tlv_options_ptr->value_ptr;

/* Get the previous hop from which this packet arrived  */
prev_hop_addr = ip_dgram_fd_ptr->src_addr;

if (rreq_option_ptr->imheb == 0)
rreq_option_ptr->src_addr = inet_address_copy (prev_hop_addr);

```

```

if((op_prg_odb_ltrace_active("trace_rreq") == OPC_TRUE)|| LTRACE_ACTIVE)
{BIN{
inet_address_print(src_hop_addr_str, ip_dgram_fd_ptr->src_addr);
inet_address_to_hname(ip_dgram_fd_ptr->src_addr, src_node_name);
inet_address_print(dest_hop_addr_str, rreq_option_ptr->dest_addr);
inet_address_to_hname(rreq_option_ptr->dest_addr, dest_node_name);
sprintf(temp_str, "from node %s (%s) destined to node %s (%s) with request ID (%d)",
src_hop_addr_str, src_node_name, dest_hop_addr_str, dest_node_name, rreq_option_ptr-
>rreq_id);
op_prg_odb_print_major(pid_string, "Received a Route Request", temp_str, OPC_NIL);
}BOUT}

```

** Update the connectivity table */*

```
Aadv_imheb_rte_neighbor_connectivity_table_update(prev_hop_addr, OPC_FALSE);
```

** Check to see if this node has received a RREQ */*

/ with the same request ID and originator address */*

```
if(aadv_imheb_route_request_forward_entry_exists(request_table_ptr, rreq_option_ptr-
>rreq_id, rreq_option_ptr->src_addr) == OPC_TRUE)
```

```
{BIN{
```

/ A route request with the same request ID has */*

/ already been received. Discard the packet */*

```
if((op_prg_odb_ltrace_active("trace_rreq") == OPC_TRUE)|| LTRACE_ACTIVE)
```

```
{BIN{
```

```
op_prg_odb_print_major(pid_string, "Destroying the Route Request",
```

```
"as same request ID has already been received", OPC_NIL);
```

```
}BOUT}
```

```
manet_rte_ip_pkt_destroy(ip_pkptr);
```

```
op_pk_destroy(aadv_imheb_pkptr);
```

```
FOUT;
```

```
}}
```

/ Insert the route request in the request table */*

```
Aadv_imheb_request_table_forward_rreq_insert(request_table_ptr, rreq_option_ptr->rreq_id,
rreq_option_ptr->src_addr);
```

/ This is a new route request packet that has not */*

/ been received by this node before. Incorporate the imheb routing metric into the rreq*/*

```
rreq_option_ptr->imheb;
```

/ Drop the rreq if it has reached network diameter */*

```
if(rreq_option_ptr->imheb> net_diameter )
```

```
{BIN{
```

/ Destroy the RREQ message */*

```
op_pk_destroy(aadv_imheb_pkptr);
```

```
manet_rte_ip_pkt_destroy(ip_pkptr);
```

```
FOUT;
```

```

}}

/* Calculate the lifetime for the route entry */
min_lifetime = ((2.0 * net_traversal_time) - (2.0 * rreq_option_ptr->imheb *
node_traversal_time));

/* Check if there exists a reverse route to the */
/* source node of the route request. */
route_entry_ptr = aadv_route_table_entry_get (route_table_ptr, rreq_option_ptr->src_addr);

/* Get the interface_info of the interface specified */
iface_elem_ptr = inet_rte_intf_tbl_access (module_data_ptr, intf_ici_fdstruct_ptr-
>intf_rcvd_index);

/* Create the corresponding port information accordingly */
/* with the IP family address. */
if (inet_address_family_get (&rreq_option_ptr->src_addr) == InetC_Addr_Family_v4)
{BIN{
/* Get the port information for the IPv4 case. */
in_port_info = ip_rte_port_info_create (intf_ici_fdstruct_ptr->intf_rcvd_index,
iface_elem_ptr->full_name);

/* Get subnet mask for the IPv4 case. */
selected_subnet_mask = subnet_mask;
}BOUT}
else
{BIN{
/* Get the port information for the IPv6 case. */
in_port_info = ipv6_rte_port_info_create (module_data_ptr, intf_ici_fdstruct_ptr-
>intf_rcvd_index);

/* Get subnet mask for the IPv6 case. */
selected_subnet_mask = subnet_mask_ipv6;
}BOUT}

if (route_entry_ptr == OPC_NIL)
{BIN{
/* No entry exists to the source of the route request. Create a new route entry*/
route_entry_ptr = aadv_imheb_route_table_entry_create (route_table_ptr, rreq_option_ptr-
>src_addr, selected_subnet_mask, prev_hop_addr, in_port_info, rreq_option_ptr->imheb,
rreq_option_ptr->src_seq_num, min_lifetime);

/* Send all packets queued to this destination */
Aadv_imheb_rte_all_pkts_to_dest_send (rreq_option_ptr->src_addr, route_entry_ptr,
OPC_FALSE);

}BOUT}
else if ((rreq_option_ptr->src_seq_num > route_entry_ptr->dest_seq_num) ||
((rreq_option_ptr->src_seq_num == route_entry_ptr->dest_seq_num) && (rreq_option_ptr-
>imheb < route_entry_ptr->imheb)) || ((rreq_option_ptr->src_seq_num == route_entry_ptr-
>dest_seq_num) && (route_entry_ptr->route_entry_state == AadvC_Invalid_Route)))

```

```

{BIN{

/* Update the route table entry */

if ((op_prg_odb_ltrace_active ("trace_rreq") == OPC_TRUE) || LTRACE_ACTIVE)
{BIN{

printf("Route entry already exists: Old dest_seq_num: %d, Old imheb: %d\n",
route_entry_ptr->dest_seq_num, route_entry_ptr->imheb);
printf("Route entry already exists: new dest_seq_num: %d, update imheb value: %d\n",
rreq_option_ptr->src_seq_num, rreq_option_ptr->imheb);
op_prg_odb_print_major (pid_string, "Updating route table with new RREQ", OPC_NIL);
}BOUT}

/* Update the sequence number */
route_entry_ptr->dest_seq_num = rreq_option_ptr->src_seq_num;

/* Do this before adding info to IP Cmn table in case route was invalid */
route_entry_ptr->imheb = rreq_option_ptr->imheb;
route_entry_ptr->next_hop_port_info = in_port_info;

if (!inet_address_equal (route_entry_ptr->next_hop_addr, prev_hop_addr))
{BIN{
/* Update the route table entry */
Aadv_imheb_route_table_entry_next_hop_update (route_table_ptr, route_entry_ptr,
inet_address_copy (prev_hop_addr), rreq_option_ptr->imheb, in_port_info);
}BOUT}

if (route_entry_ptr->route_entry_state != AadvC_Invalid_Route)
{BIN{
existing_lifetime = route_entry_ptr->route_expiry_time - op_sim_time ();
lifetime = aadv_imheb_rte_max_find (existing_lifetime, min_lifetime);
}BOUT}
else
{BIN{
lifetime = min_lifetime;
aadv_imheb_route_table_entry_state_set (route_table_ptr, route_entry_ptr, rreq_option_ptr-
>src_addr, AadvC_Valid_Route);
}BOUT}

/* Set the valid sequence number field to true */
route_entry_ptr->valid_dest_sequence_number_flag = OPC_TRUE;
aadv_imheb_route_table_entry_expiry_time_update (route_entry_ptr, rreq_option_ptr-
>src_addr, lifetime, AODVC_ROUTE_ENTRY_INVALID);
}BOUT}
else if (!inet_address_equal (route_entry_ptr->next_hop_addr, prev_hop_addr))
{BIN{
/* Drop the route request, if the nexthop on reverse path */
/* is different from the previous hop. This is to avoid */
/* loop formation in certain cases */
/* Destroy the RREQ message */

```

```

op_pk_destroy (aadv_imheb_pkptr);
manet_rte_ip_pkt_destroy (ip_pkptr);

FOUT;
}}

/* A node generates a route reply if : */
/* (a) it is the destination of the RREQ */
/* (b) it has an active route to the destination */
/* and the destination only flag is not set. */
/* Determine if this node is the destination of the */
/* route request. If so, generate a route reply */
/* back to the source of the route request */
if (manet_rte_address_belongs_to_node (module_data_ptr, rreq_option_ptr->dest_addr) ==
    OPC_TRUE)
{
    BIN{
    /* This is the destination of the RREQ */
    /* Send a route reply message */
    Aadv_imheb_rte_route_reply_send (rreq_option_ptr, ip_dgram_fd_ptr, OPC_TRUE,
    route_entry_ptr, OPC_NIL);
    /* Update the statistic for the number of route */
    /* replies sent from the destination. */
    Aadv_imheb_support_route_reply_sent_stats_update (local_stat_handle_ptr,
    global_stat_handle_ptr, OPC_TRUE);

    /* Destroy the RREQ message */
    op_pk_destroy (aadv_imheb_pkptr);
    manet_rte_ip_pkt_destroy (ip_pkptr);

    FOUT;
    }}

static void aadv_imheb_rte_rrep_pkt_arrival_handle (Packet* ip_pkptr, Packet* aadv_pkptr,
    IpT_Dgram_Fields* ip_dgram_fd_ptr, IpT_Rte_Ind_Ici_Fields* intf_ici_fdstruct_ptr,
    AadvT_Packet_Option* tlv_options_ptr)
{
    AadvT_Rrep*          rrep_option_ptr;
    InetT_Address        prev_hop_addr;
    InetT_Address        next_hop_addr;
    IpT_Interface_Info*  iface_elem_ptr;
    IpT_Port_Info        in_port_info;
    AadvT_Route_Entry*   route_entry_ptr = OPC_NIL;
    Boolean               route_updated = OPC_FALSE;
    Boolean               reached_src = OPC_FALSE;
    int                   dest_seq_num;
    double               imheb;
    AadvC_Route_Entry_State route_entry_state;
    Packet*              ip_rrep_pkptr;
    char                  src_node_name [OMSC_HNAME_MAX_LEN];
    char                  src_hop_addr_str [INETC_ADDR_STR_LEN];

```

```

char                dest_node_name [OMSC_HNAME_MAX_LEN];
char                dest_hop_addr_str [INETC_ADDR_STR_LEN];
char                temp_str [2048], temp_str1 [2048];
char                next_addr_str [INETC_ADDR_STR_LEN];
char                next_node_name [OMSC_HNAME_MAX_LEN];
char                dest_addr_str [INETC_ADDR_STR_LEN];
double              existing_lifetime, lifetime;
double              route_expiry_time;
ManetT_Nexthop_Info* manet_nexthop_info_ptr = OPC_NIL;
AodvT_Route_Entry*  fwd_path_entry_ptr = OPC_NIL;
InetT_Subnet_Mask    selected_subnet_mask;

/** Handles the arrival of the route reply message      **/
FIN (aodv_rte_rep_pkt_arrival_handle (<args>));

/* A route reply packet has arrived at this node */
/* Get the reply options from the packet */
rrep_option_ptr = (AodvT_Rrep*) tlv_options_ptr->value_ptr;

/* Get the previous hop from which this packet arrived */
prev_hop_addr = ip_dgram_fd_ptr->src_addr;

/* Check if node has received its own route reply */
if(manet_rte_address_belongs_to_node (module_data_ptr, rrep_option_ptr->dest_addr))
{BIN{
/* Drop the route reply */
op_pk_destroy (aodv_imheb_pkptr);
manet_rte_ip_pkt_destroy (ip_pkptr);
FOUT;
}}

/* Update the connectivity table */
Aodv_imheb_rte_neighbor_connectivity_table_update (prev_hop_addr, OPC_FALSE);

/* Update the route to the previous path of the route reply message. */
Aodv_imheb_rte_route_table_entry_update (ip_dgram_fd_ptr, intf_ici_fdstruct_ptr,
tlv_options_ptr);

if (LTRACE_ACTIVE)
{BIN{
inet_address_print (src_hop_addr_str, ip_dgram_fd_ptr->src_addr);
inet_address_to_hname (ip_dgram_fd_ptr->src_addr, src_node_name);
inet_address_print (dest_hop_addr_str, rrep_option_ptr->dest_addr);
inet_address_to_hname (rrep_option_ptr->dest_addr, dest_node_name);
sprintf (temp_str, "from node %s (%s). Originator of this RREP: %s (%s)",
src_hop_addr_str, src_node_name, dest_hop_addr_str, dest_node_name);
op_prg_odb_print_major (pid_string, "Received a route reply packet", temp_str, OPC_NIL);
}BOUT}

/* Drop the rreq if it has reached network diameter */
if (rrep_option_ptr->hop_count > inet_diameter )

```

```

{BIN{
/* Destroy the RREQ message */
op_pk_destroy(aodv_imheb_pkptr);
manet_rte_ip_pkt_destroy(ip_pkptr);
FOUT;
}}
/* Get the interface_info of the interface specified*/
iface_elem_ptr = inet_rte_intf_tbl_access(module_data_ptr, intf_ici_fdstruct_ptr-
>intf_recvd_index);

/* Check for the IP address family.*/
if(inet_address_family_get(&rrep_option_ptr->dest_addr) == InetC_Addr_Family_v4)
{BIN{
/* Get the port information for the IPv4 case.*/
in_port_info = ip_rte_port_info_create(intf_ici_fdstruct_ptr->intf_recvd_index,
iface_elem_ptr->full_name);

/* Get subnet mask for the IPv4 case.*/
selected_subnet_mask = subnet_mask;
}BOUT}
else
{BIN{
/* Get the port information for the IPv6 case.*/
in_port_info = ipv6_rte_port_info_create(module_data_ptr, intf_ici_fdstruct_ptr-
>intf_recvd_index);

/* Get subnet mask for the IPv6 case. */
selected_subnet_mask = subnet_mask_ipv6;
}BOUT}

/* Check if this is the final destination of the RREP message*/
reached_src = manet_rte_address_belongs_to_node(module_data_ptr, rrep_option_ptr-
>src_addr);

/* Check if the route exists to the destination IP address in the RREP message. */
route_entry_ptr = aodv_route_table_entry_get(route_table_ptr, rrep_option_ptr->dest_addr);
if(route_entry_ptr == OPC_NIL)
{BIN{
/* if no route exists to this destination, create a new route table entry.*/
route_entry_ptr = aodv_route_table_entry_create(route_table_ptr, rrep_option_ptr->dest_addr,
selected_subnet_mask, ip_dgram_fd_ptr->src_addr, in_port_info, rrep_option_ptr->imheb,
rrep_option_ptr->dest_seq_num, rrep_option_ptr->lifetime);

/* Send all packets queued to this destination */
Aodv_imheb_rte_all_pkts_to_dest_send(rrep_option_ptr->dest_addr, route_entry_ptr,
reached_src);

/* Indicate that the route has been created */
route_updated = OPC_TRUE;
}BOUT}

```

```

else
{BIN{
dest_seq_num = route_entry_ptr->dest_seq_num;
imhebt = route_entry_ptr->imhebt;
route_entry_state = route_entry_ptr->route_entry_state;

if (LTRACE_ACTIVE)
{BIN{
printf ("Route already Exists: DestSeq: %d, Cal_imhebt:%d \n", dest_seq_num, imhebt);
printf (" NewDestSeq: %d, Updated imhebt %d \n", rrep_option_ptr->dest_seq_num,
rrep_option_ptr->imhebt);
}BOUT}

if ((rrep_option_ptr->dest_seq_num > dest_seq_num) || ((rrep_option_ptr->dest_seq_num ==
dest_seq_num) && (route_entry_state == AodvC_Invalid_Route)) ||
((rrep_option_ptr->dest_seq_num == dest_seq_num) && (rrep_option_ptr->imhebt < imhebt)))
{BIN{

if (LTRACE_ACTIVE)
{BIN{
op_prg_oddb_print_major (pid_string, " Updating Route Table with new RREP", OPC_NIL);
}BOUT}

/* Do this before adding info to IP Cmn table in case route was invalid */
route_entry_ptr->imhebt = rrep_option_ptr->hop_imhebt;
route_entry_ptr->next_hop_port_info = in_port_info;

if (!inet_address_equal (route_entry_ptr->next_hop_addr, prev_hop_addr))
{BIN{
Aodv_imhebt_route_table_entry_next_hop_update (route_table_ptr, route_entry_ptr,
inet_address_copy (prev_hop_addr), rrep_option_ptr->imhebt, in_port_info);
}BOUT}

/* Set the route to be a valid route */
if (route_entry_state != AodvC_Valid_Route)
{BIN{
aodv_route_table_entry_state_set (route_table_ptr, route_entry_ptr, rrep_option_ptr-
>dest_addr, AodvC_Valid_Route);
}BOUT}

/* Mark the destination sequence as valid */
Aodv_imhebt_route_table_entry_param_set (route_entry_ptr,
AODVC_ROUTE_ENTRY_VALID_SEQ_NUM_FLAG, OPC_TRUE);

/* Set the destination sequence number to that in the RREP message */
Aodv_imhebt_route_table_entry_param_set (route_entry_ptr,
AODVC_ROUTE_ENTRY_DEST_SEQ_NUM, rrep_option_ptr->dest_seq_num);

aodv_route_table_entry_expiry_time_update (route_entry_ptr, rrep_option_ptr->dest_addr,
rrep_option_ptr->lifetime, AODVC_ROUTE_ENTRY_INVALID);

```

```

/* Send all packets queued to this destination */
aodv_rte_all_pkts_to_dest_send (rrep_option_ptr->dest_addr, route_entry_ptr, reached_src);

/* Indicate that the route has been created */
route_updated = OPC_TRUE;
}BOUT}
FOUT;
}}

static void
aodv_imheb_rte_rerr_pkt_arrival_handle (Packet* ip_pkptr, Packet* aodv_pkptr,
IpT_Dgram_Fields* ip_dgram_fd_ptr, IpT_Rte_Ind_Ici_Fields*
PRG_ARG_UNUSED(intf_ici_fdstruct_ptr), AodvT_Packet_Option* tlv_options_ptr)

{
AodvT_Rerr*      rerr_option_ptr;
char             addr_str [INETC_ADDR_STR_LEN];
char             node_name [OMSC_HNAME_MAX_LEN];
char             temp_str [256];

/** A route error packet has been received from a neighbor */
/** Process the route error message. */
FIN (aodv_imheb_rte_rerr_pkt_arrival_handle (<args>));

if (LTRACE_ACTIVE)
{BIN{
inet_address_print (addr_str, ip_dgram_fd_ptr->src_addr);
inet_address_to_hname (ip_dgram_fd_ptr->src_addr, node_name);
sprintf (temp_str, "Received a route error packet from node %s (%s)", addr_str, node_name);
op_prg_oddb_print_major (pid_string, temp_str, OPC_NIL);
}BOUT}

/* Get the route error option */
rerr_option_ptr = (AodvT_Rerr*) tlv_options_ptr->value_ptr;

/* Process the route error message */
Aodv_imheb_rte_route_error_process (ip_dgram_fd_ptr->src_addr, OPC_NIL, rerr_option_ptr->unreachable_dest_lptr,
AodvC_Rerr_Received, rerr_option_ptr->no_delete_flag);

/* Destroy the packets */
op_pk_destroy (aodv_imheb_pkptr);
manet_rte_ip_pkt_destroy (ip_pkptr);
FOUT;
}

```

Appendix B

Comment: Source Code for the main functions used for the process model implementation of the DSR_XiMHEB routing protocol in OPNET 14.0.

NB: Due to the length of this source code, only very important code section are included in this appendix

```
#line 1 "C:/Program Files/OPNET/14.0.A/models/std/manet/dsr_rte.pr.c"
/* Process model C form file: dsr_ximheb_rte.pr.c */
/* Portions of this file copyright 1992-2007 by OPNET Technologies, Inc. */
/* Author: Michel Mbougini */

/* This variable carries the header into the object file */
const char dsr_ximheb_rte_pr_c [] = "MIL_3_Tfile_Hdr_140A 30A op_runsim 7 4FB2E747
4FB2E747 1 mike-HP mike 0 0 none none 0 0 none 0 0 0 0 0 0 0 0 18a9 3
";
#include <string.h>

/* OPNET system definitions */
#include <opnet.h>

/* Header Block */
#include <ip_rte_support.h>
#include <ip_rte_v4.h>
#include <ip_dgram_sup.h>
#include <ip_higher_layer_proto_reg_sup.h>
#include <dsr_ximheb.h>
#include <dsr_ximheb_pkt_support.h>
#include <dsr_ximheb_ptypes.h>

/* Transitions */
#define PACKET_ARRIVAL (invoke_mode == OPC_PROINV_INDIRECT)
#define TIMER_EXPIRY ((invoke_mode == OPC_PROINV_DIRECT) && (intrpt_type
== OPC_INTRPT_PROCEDURE) && ((intrpt_code == DSRC_ROUTE_REQUEST_TIMER)
|| (intrpt_code == DSRC_ROUTE_MAINTENANCE_TIMER)))
#define ROUTE_CACHE_DUMP ((invoke_mode == OPC_PROINV_DIRECT) &&
(intrpt_type == OPC_INTRPT_PROCEDURE) && (intrpt_code ==
DSRC_ROUTE_CACHE_EXPORT))

/* Traces */
#define LTRACE_ACTIVE (op_prg_odb_ltrace_active("dsr_ximheb_rte"))

/* Prototypes */
static void dsrx_ximheb_rte_sv_init(void);
static void dsr_ximheb_rte_received_pkt_handle(void);
static void dsr_ximheb_rte_received_route_request_process(Packet*,
DsrT_Packet_Option*);
static void dsr_ximheb_rte_received_route_reply_process(Packet*,
DsrT_Packet_Option*);
```

```

static void      dsr_ximheb_rte_received_route_error_process (Packet*,
DsrT_Packet_Option*);

/* End Prototypes */

/* State variable definitions */
typedef struct
{
/* Internal state tracking for FSM */
FSM_SYS_STATE
/* State Variables */
IpT_Rte_Module_Data* module_data_ptr; /* Module Data Memory for the IP module */
Objid   own_mod_obji; /* Objid of thie module */
Objid   own_node_obji; /* Objid of this node */
Prohandle own_prohandle; /* Own process handle */
Prohandle parent_prohandle; /* Process handle of the parent process (manet_mgr) */
char   proc_model_name [10]; /* Process Model Name */
DsrT_Route_Cache* route_cache_ptr; /* Pointer to the Route Cache */

}dsr_ximheb_rte_state;

/* Function Block */

#if !defined (VOSD_NO_FIN)
enum { _op_block_origin = __LINE__ + 2};
#endif

static void
dsr_ximheb_rte_sv_init (void)
{
/** Initialize the state variables **/
#ifdef BIN
#define BIN
#endif
#ifdef BOUT
#define BOUT
#endif
#line 187
FIN (dsr_ximheb_rte_sv_init (void));

/* Access the module data memory */
module_data_ptr = (IpT_Rte_Module_Data*) op_pro_modmem_access ();

/* Initialize variables used for process registry. */
/* Obtain the tcp2 module's objid. */
own_mod_objid = op_id_self ();

/* Obtain the node's objid. */
own_node_objid = op_topo_parent (own_mod_objid);

```

```

/* Obtain own process handle. */
own_prohandle = op_pro_self ();

/* Obtain parent process handle */
parent_prohandle = op_pro_parent (own_prohandle);

/* Own process ID */
own_pro_id = op_pro_id (own_prohandle);

/* Parent process ID */
parent_pro_id = op_pro_id (parent_prohandle);

/* Obtain the name of the process. It is the "process model" attribute of the node. */
op_ima_obj_attr_get (own_mod_objid, "process model", proc_model_name);

/* Initialize the unique identifiers */
ack_request_identifier = 0;
route_request_identifier = 0;

/* Set up the display string. */
sprintf (pid_string, "dsr_ximheb_rte PID (%d)", own_pro_id);

/* Determine whether to export the routes to a file */
routes_export = dsr_ximheb_support_route_export_sim_attr_get ();

if (routes_export == OPC_TRUE)
{
    BIN{
    /* Initialize the OT package */
    Dsr_ximheb_support_route_print_to_ot_initialize ();

    /* Schedule a call interrupt to close the file */
    op_intrpt_schedule_call (OPC_INTRPT_SCHED_CALL_ENDSIM, 0,
    dsr_ximheb_support_route_print_to_ot_close, OPC_NIL);
    }BOUT}

/* Determine whether to dump the routes for display */
routes_dump = dsr_ximheb_support_route_record_sim_attr_get ();

/* Get a handle to the global statistics */
global_stathandle_ptr = dsr_ximheb_support_global_stat_handles_obtain ();

/* Initialize the print procedure for the options field in the DSR_XiMHEB packet */
op_pk_format_print_proc_set ("dsr_ximheb", "Options",
dsr_ximheb_pkt_support_options_print);

FOUT;
}

```

/****** PACKET ARRIVAL FUNCTIONS *****/

```
static void dsr_ximheb_rte_received_pkt_handle (void)
{
    Packet*      ip_pkptr = OPC_NIL;
    Packet*      copy_pkptr = OPC_NIL;
    Packet*      return_pkptr = OPC_NIL;
    IpT_Rte_Ind_Ici_Fields* intf_ici_fdstruct_ptr = OPC_NIL;
    IpT_Dgram_Fields* ip_dgram_fd_ptr = OPC_NIL;
    Packet*      dsr_ximheb_pkptr = OPC_NIL;
    List*        tlv_options_lptr = OPC_NIL;
    int          num_options, count;
    DsrT_Packet_Option* dsr_ximheb_tlv_ptr = OPC_NIL;
    DsrT_Packet_Type packet_type = DsrC_Undef_Packet;
    char         addr_str [INETC_ADDR_STR_LEN];
    char         node_name [OMSC_HNAME_MAX_LEN];
    char         temp_str [256];
    Compcode     status;
    Boolean       app_pkt_set = OPC_FALSE;

    FIN (dsr_ximheb_rte_received_pkt_handle (void));

    ip_pkptr = (Packet*) op_pro_argmem_access ();

    if (ip_pkptr == OPC_NIL)
        dsr_ximheb_rte_error ("Could not obtain the packet from the argument memory", OPC_NIL,
        OPC_NIL);
    /* Access the information from the incoming IP packet */
    manet_rte_ip_pkt_info_access (ip_pkptr, &ip_dgram_fd_ptr, &intf_ici_fdstruct_ptr);

    /* Determine the packet type */
    packet_type = dsr_ximheb_rte_packet_type_determine (ip_dgram_fd_ptr, intf_ici_fdstruct_ptr);

    /* Check if this IP packet is carrying a DSR header */
    if (packet_type == DsrC_Higher_Layer_Packet)
    {BIN{
    if (LTRACE_ACTIVE)
    {BIN{
    inet_address_print (addr_str, ip_dgram_fd_ptr->dest_addr);
    inet_address_to_hname (ip_dgram_fd_ptr->dest_addr, node_name);
    sprintf (temp_str, "to destination %s (%s)", addr_str, node_name);
    op_prg_odb_print_major (pid_string, "An application packet has arrived at this node", temp_str,
    OPC_NIL);
    }BOUT}

    Dsr_ximheb_rte_app_pkt_arrival_handle (ip_pkptr, intf_ici_fdstruct_ptr, ip_dgram_fd_ptr,
    OPC_FALSE);
    FOUT;

    }}
}
```

```

/* If this packet is received from the MAC layer update the statistic for the total traffic */
Dsr_ximheb_support_total_traffic_received_stats_update (stat_handle_ptr,
global_stathandle_ptr, ip_pkptr);

/* Get the DSR packet from the IP datagram */
op_pk_nfd_get (ip_pkptr, "data", &dsr_ximheb_pkptr);

/* Check if this is an application packet or just a DSR_XiMHEB routing packet */
app_pkt_set = op_pk_nfd_is_set (dsr_ximheb_pkptr, "data");

/* Get the list of options */
op_pk_nfd_access (dsr_ximheb_pkptr, "Options", &tlv_options_lptr);

/* Set the DSR_XiMHEB packet into the IP datagram */
op_pk_nfd_set (ip_pkptr, "data", dsr_ximheb_pkptr);

if (app_pkt_set == OPC_FALSE)
{BIN{

/* If this is a DSR routing packet, update the statistic for routing traffic */
Dsr_ximheb_support_routing_traffic_received_stats_update (stat_handle_ptr,
global_stathandle_ptr, ip_pkptr);
        }BOUT}
else
{BIN{
/* This is an application packet. Decrease the TTL if i am not the destination node */
if (manet_rte_address_belongs_to_node (module_data_ptr, ip_dgram_fd_ptr->dest_addr) ==
OPC_FALSE)
ip_dgram_fd_ptr->ttl--;
        }BOUT}

/* Get the number of options */
num_options = op_prg_list_size (tlv_options_lptr);

for (count = 0; count < num_options; count++)
{BIN{
/* Make a copy of the incoming packet for each option */
copy_pkptr = manet_rte_ip_pkt_copy (ip_pkptr);

/* Get the DSR packet from the IP datagram */
op_pk_nfd_get (copy_pkptr, "data", &dsr_pkptr);

/* Get the list of options */
op_pk_nfd_access (dsr_ximheb_pkptr, "Options", &tlv_options_lptr);

/* Set the DSR_XiMHEB packet into the IP datagram */
op_pk_nfd_set (copy_pkptr, "data", dsr_pkptr);

/* Get each option */
Dsr_ximheb_tlv_ptr = (DsrT_Packet_Option*) op_prg_list_access (tlv_options_lptr, count);

```

```

/* Process the option based on the type */
switch (dsr_ximheb_tlv_ptr->option_type)
{
case (DSRC_ROUTE_REQUEST):
{BIN{
/* The packet contains a route request option */
/* Insert this node into the route request list */
dsr_pkt_support_route_request_hop_insert (copy_pkptr, intf_ici_fdstruct_ptr-
>interface_received);

/* Insert the route in the route cache based on */
/* the requirement for caching overheard information */
dsr_rte_route_cache_update (dsr_tlv_ptr, ip_dgram_fd_ptr);

/* After possibly inserting the route in the route cache */
/* process the received route request option */
Dsr_ximheb_rte_received_route_request_process (copy_pkptr, dsr_tlv_ptr);

{ BOUT break; }
}}

case (DSRC_ROUTE_REPLY):
{BIN{
/* The packet contains a route reply option */

/* Insert the route in the route cache based on */
/* the requirement for caching overheard information */
Dsr_ximheb_rte_route_cache_update (dsr_tlv_ptr, ip_dgram_fd_ptr);

/* After possibly inserting the route in the route cache */
/* process the received route reply option */
Dsr_ximheb_rte_received_route_reply_process (copy_pkptr, dsr_tlv_ptr);

{ BOUT break; }
}}

case (DSRC_ROUTE_ERROR):
{BIN{
/* The packet contains a route error option */
/* Process the received route error */
Dsr_ximheb_rte_received_route_error_process (copy_pkptr, dsr_tlv_ptr);

{ BOUT break; }
}}

case (DSRC_ACK_REQUEST):
{BIN{
/* The packet contains an acknowledgement */
/* request option. Process the option */
status = dsr_ximheb_rte_received_ack_request_process (copy_pkptr);

```

```

if (status == OPC_COMPCODE_SUCCESS)
{BIN{
op_pk_nfd_get (ip_pkptr, "data", &dsr_pkptr);
dsr_ximheb_pkt_support_option_remove (dsr_pkptr, DSRC_ACK_REQUEST);
op_pk_nfd_set (ip_pkptr, "data", dsr_pkptr);
num_options--;
count--;
}BOUT}

{ BOUT break; }
}}

case (DSRC_ACKNOWLEDGEMENT):
{BIN{

/* the requirement for caching overheard information */
Dsr_ximheb_rte_route_cache_update (dsr_tlv_ptr, ip_dgram_fd_ptr);

/* After possibly inserting the route in the route cache */
/* process the received acknowledgement option */
Dsr_ximheb_rte_received_acknowledgement_option_process (copy_pkptr, dsr_tlv_ptr);

{ BOUT break; }
}}

case (DSRC_SOURCE_ROUTE):
{BIN{
/* If the packet contains a DSR source route option */

/* Insert the route in the route cache based on */
/* the requirement for caching overheard information */
Dsr_ximheb_rte_route_cache_update (dsr_tlv_ptr, ip_dgram_fd_ptr);

/* After possibly inserting the route in the route cache */
/* process the received DSR_XiMHEB source route option*/
Dsr_ximheb_rte_received_dsr_source_route_option_process (copy_pkptr);

{ BOUT break; }
}}

default:
{BIN{
/* Invalid option in packet*/
Dsr_ximheb_rte_error ("Invalid Option Type in DSR packet", OPC_NIL, OPC_NIL);
}BOUT}
}
}BOUT}
FOUT;
}

```