

ANALYSIS ON WORMHOLE ATTACK IN AD HOC NETWORK

SYAZWANI BT MOHD SOBRI



This report is submitted in partial fulfilment of the requirements for the Bachelor of
Computer Science (Networking)

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY
UNIVERSITI TEKNIKAL MALAYSIA MELAKA
2016

BORANG PENGESAHAN STATUS TESIS*

JUDUL: ANALYSIS ON WORMHOLE ATTACK IN AD HOC NETWORK

SESI PENGAJIAN: 2015/2016

Saya SYAZWANI BT MOHD SOBRI
(HURUF BESAR)

mengaku membenarkan tesis (PSM/Sarjana/Doktor Falsafah) ini disimpan di Perpustakaan Fakulti Teknologi Maklumat dan Komunikasi dengan syarat-syarat kegunaan seperti berikut:

1. Tesis dan projek adalah hakmilik Universiti Teknikal Malaysia Melaka.
2. Perpustakaan Fakulti Teknologi Maklumat dan Komunikasi dibenarkan membuat salinan untuk tujuan pengajian sahaja.
3. Perpustakaan Fakulti Teknologi Maklumat dan Komunikasi dibenarkan membuat salinan tesis ini sebagai bahan pertukaran antara institusi pengajian tinggi.
4. ** Sila tandakan (/)



 SULIT

 TERHAD

(Mengandungi maklumat yang berdarjah keselamatan atau kepentingan Malaysia seperti yang termaktub di dalam AKTA RAHSIA RASMI 1972)

(Mengandungi maklumat TERHAD yang telah ditentukan oleh organisasi/badan di mana penyelidikan dijalankan)

اونيور سیتی تیکنیکل ملیسیا ملاک

 TIDAK TERHAD

(TANDATANGAN PENULIS)

Alamat tetap: Pt 1115 Taman Kemahiran
Pauh Panji, 15200
Kota Bharu
Kelantan.

(TANDATANGAN PENYELIA)

Nama Penyelia: PN HANIZA BT
NAHAR

Tarikh: 19 DISEMBER 2016

Tarikh: 19 DISEMBER 2016

CATATAN: * Tesis dimaksudkan sebagai Laporan Akhir Projek Sarjana Muda (PSM)
** Jika tesis ini SULIT atau TERHAD, sila lampirkan surat daripada pihak berkuasa.

ANALYSIS ON WORMHOLE ATTACK IN AD HOC NETWORK



UNIVERSITI TEKNIKAL MALAYSIA MELAKA

DECLARATION

I hereby declare that this project report entitled
ANALYSIS ON WORMHOLE ATTACK IN AD HOC NETWORK
is written by me and is my own effort and that no part has been plagiarized
Without citations.



STUDENT :

Syazwani

Date : 19/12/2016

(SYAZWANI BT MOHD SOBRI)

اونيور سیتی تکنیکل ملیسیا ملاک

I hereby declare that I have read this project report and found
this project report is sufficient in term of the scope and quality for the award of
Bachelor of Computer Science (Computer Networking) With Honours.

SUPERVISOR :

PN Haniza

Date : 19/12/2016

(PN HANIZA BT NAHAR)

DEDICATION

First of all I thank to Allah for His guidance at a time when I get stuck, need help and solve the problem. Thanks to Allah because He smooth all the works for this project. I also dedicate my dissertation work to my family and all my friends who's always support me. A special feeling of gratitude to my loving parents who always give me encouragement whenever I have depressed.

I also dedicate this dissertation to my friends who supported me throughout the process that always give an idea to make this project more relevant. I would like to thank my supervisor Pn Haniza Bt Nahar for her help and support through this project and always support all my idea and give guidance to complete this project.

Thank you very much.

ACKNOWLEDGMENT

First of all, I would like to thanks to my parents Mohd Sobri Bt Ali and Zubaidah Bt Harun for their support and give me more motivation to me during this research. I'm very appreciating for all the support from my parents and family.

For my supervisor, Pn Haniza Bt Nahar many thanks for your guidance and sharing the knowledge step by step through this research. Besides, that always spent your time to improve my research from time to time.

Finally, I would like to thanks to my entire member that always support me and sharing the idea in my research and also being co-operative and helpful.

ABSTRACT

The recent development in the wireless technology have made remarkable enhancement in productivity in the corporate and industrials sector. However, these development also introduced new security threats. Wormhole attack is one of the security threats in wireless. In this project AODV has been chosen for implementation of this attack for mobile ad hoc network. The effect of wormhole attack on AODV routing protocols is analysed on behalf of parameters like throughput, end to end delay and normalized routing load.

ABSTRAK

Pembangunan dalam teknologi tanpa wayar merupakan suatu peningkatan produktiviti dalam sektor industri dan korporat. Walaubagaimanapun, pembangunan ini juga telah memperkenalkan banyak ancaman sekuriti dalam teknologi tanpa wayar. Serangan “Wormhole ” merupakan salah satu daripada ancaman sekuriti tersebut. Untuk projek ini, AODV telah dipilih untuk melaksanakan serangan “wormhole”. Kesan serangan tersebut akan dianalisis dalam tiga parameter iaitu “throughput”, “end to end delay” dan “normalised routing load”.

TABLE OF CONTENTS

CHAPTER	SUBJECT	PAGE
	DECLARATION	i
	DEDICATION	ii
	ACKNOWLEDGMENT	iii
	ABSTRACT	iv
	ABSTRAK	v
	TABLE OF CONTENTS	vi
	LIST OF TABLES	x
	LIST OF FIGURES	xi
CHAPTER 1	INTRODUCTION	1
	1.1 Introduction	1
	1.2 Research Problem	2
	1.3 Project Question	3
	1.4 Research Objectives	3
	1.5 Project Scope	4
	1.6 Research Contribution	5
	1.7 Project Organisation	5
	1.8 Expected Output	6
	1.9 Conclusion	7
CHAPTER II	LITERATURE REVIEW	8
	2.1 Introduction	8

2.2 Ad hoc wireless network.	9
2.3 Classification of wireless networks.	11
2.3.1 Cellular network	12
2.3.2 Wireless Mesh Networks	12
2.3.3 Wireless Sensor Networks	12
2.4 Classification of routing protocols.	13
2.5 Ad hoc On Demand Routing (AODV)	14
2.5.1 Route Discovery	14
2.5.2 Route Maintenance	16
2.6 Security Attacks in Ad Hoc Networks	17
2.6.1 Passive Attack	17
2.6.2 Active Attack	17
2.7 Wormhole Attack	20
2.8 Implication of wormhole attack to Ad hoc network.	21
2.9 Related Work	22
2.10 Summarization	23
2.11 Conclusion	23
CHAPTER III METHODOLOGY	24
3.1 Introduction	24
3.3 Project Tool and Requirement	27
3.4 Project Gant Chart and Milestone.	28
3.5 Conclusion	30
CHAPTER IV DESIGN	31
4.1 Introduction	31
4.2 Network System Architecture	31
4.2.1 Logical design	32
4.2.1 Simulation parameter from previous work	32

4.2.2 Project simulation parameter	34
4.3 Possible Scenarios	35
4.5 Metric Measurement	35
4.6 Conclusion	36
CHAPTER V IMPLEMENTATION	37
5.1 Introduction	37
5.2 Simulation Setup	38
5.2.1 Simulation Overview	38
5.2.2 Simulation Parameter	38
5.2.3 Wormhole Attack Implementation	39
5.2.4 Recompilation of NS-2	41
5.2.5 Generate NAM Network Animator	42
5.3 Conclusion	44
CHAPTER VI TESTING AND ANALYSIS	45
6.1 Introduction	45
6.2 Result and Analysis	46
6.2.1 Metric measurement 1: Throughput (bits per second)	46
6.2.2 Metric measurement 2: End To End Delay (second)	49
6.2.3 Metric measurement 3: Normalised Routing Load	51
6.3: Conclusion	52
CHAPTER VII CONCLUSION	53
7.1 Introduction	53
7.2 Project Summarization	53
7.3 Project Contribution	54
7.4 Project Limitation	55
7.5 Future Work	55
7.6 Conclusion	55

REFERENCES

56

APPENDICES

57



LIST OF TABLES

TABLES	TITLE	PAGE
1.1	Project Question	3
1.2	Project Objectives	4
2.1	RREQ Field	15
2.2	RREP Field	15
2.3	Types of attacks in ad hoc	19
2.4	Justification from previous works	22
3.1	Hardware and software requirements	27
4.1	previous work of simulation parameter	33
4.2	Simulation parameter	34
5.1	AODV Parameter for 20 nodes	39
6.1	Data collected for throughput	47
6.2	Data collected for delay	49
6.3	Data collected for Normalised Routing Load	51

LIST OF FIGURES

DIAGRAM	TITLE	PAGE
2.1	Ad hoc Network	10
2.2	Taxonomy of wireless network	11
2.3	Taxonomy of routing protocol	14
2.4	Route Discovery in AODV	15
2.5	Route Maintenance in AODV	16
2.6	Taxonomy of attacks in ad hoc	18
2.7	Network affected by wormhole attack	20
3.1	Waterfall model	25
3.2	Project Gant Chart	29
4.1	Ad hoc topology	32
4.2	process in NS2 simulator	35
5.1	NS2 Overview	38
5.2	Mobile node schematic diagram	40
5.3	Wormhole attack implementation on Tcl script	40
5.4	NS2 recompilation command	41
5.5	NAM animator without wormhole attack	42
5.6	Path discovered	42
5.7	NAM animator with wormhole attack	43
5.8	Path discovery with wormhole attack	43
6.2	Graph of throughput	48
6.3	AWK Script for delay	49
6.4	Graph of delay	50

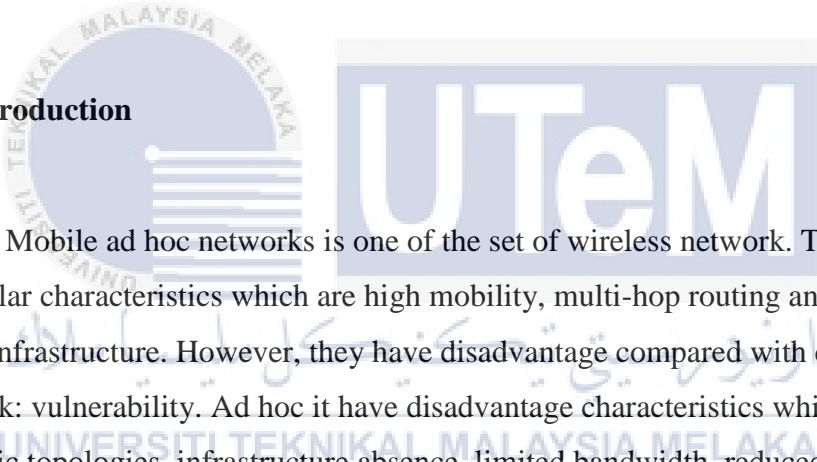
6.5	AWK Script for Normalised Routing Load	51
6.6	Graph of Normalised Routing Load	52



CHAPTER 1

INTRODUCTION

1.1 Introduction



Mobile ad hoc networks is one of the set of wireless network. They have particular characteristics which are high mobility, multi-hop routing and the absence of fix infrastructure. However, they have disadvantage compared with classic network: vulnerability. Ad hoc it have disadvantage characteristics which are dynamic topologies, infrastructure absence, limited bandwidth, reduced autonomy and auto-configuration. Because of these particular characteristics, ad hoc networks are much more vulnerable to attacks and naturally offer more faults than other types of network when faced with a potential attacker.

The term “attack” specifies an action that negotiate the reliability and confidentiality of the network’s information. In ad-hoc, generally distinguish two categories which are passive attacks where the attacker only listens the traffic without effecting the overall process of routing while active attacks is effecting the overall process of routing in the network

In this research, will concentrate on wormhole attack which is categorised as active attack. Wormhole attack is one of the Denial-of-Service attack effective on the

network layer (layer 3 of OSI model). In wormhole attack, the nodes of attacker creates the illusion tunnel to transmit and replies data packet through the network. This attack give huge effects against routing of network.

The wormhole attack that consists of two malicious node shorter than the other route within the network so the network will choose the shortest path to transfer data from source to destination.

In the end of this project, the results of with and without wormhole attack will be presented in graph. The expression by combining the result of three metrics which are throughput, end-to-end delay and normalised routing load also will be created.



1.2 Research Problem

Wormhole attack put severe threats to the ad hoc networks. This attack is very dangerous because it can also still be performed even if the network communication provides authentication and confidentiality. The wormhole attack is able to confuse the clustering procedure and lead to a wrong topology and it can partition the network through control links between two cluster heads of the routing hierarchy. Many previous works proposed or implemented various techniques but it is still very severe attack

Security against wormhole attack is a challenging task since various techniques had been proposed. It may caused by the lacking of infrastructure in ad hoc networks.

1.3 Project Question

In fact what are the properties of wormhole attack? An initial study must be done before any deployment takes place. Once the environment of ad hoc network is understood then what are activities of wormhole attack do in ad hoc network? Last but not least what is the purpose of analysing the wormhole attack behaviour in ad hoc network?

Table 1.1: Project Question

RQ	Research Questions
RQ1	What is wormhole attack in Ad hoc network?
RQ2	How to conduct activities of wormhole attack in Ad hoc network?
RQ3	What is the effect of wormhole attack on performance of Ad hoc network?



1.4 Research Objectives

According to the research question, this project will focus on three objectives so that the expected output can be achieved. The initial study about impact of wormhole attack on Ad hoc network will be understood. After that the stimulation of wormhole attack in ad hoc using NS-2 will be completed. Last but not least, the discussion will be covered by comparing the results with and without wormhole attack based on metric packet loss, end-to-end delay and throughput.

Table 1.2: Project Objectives

RQ	RO	Research Objectives
RQ1	RO1	To study wormhole attack on Ad hoc network
RQ2	RO2	To analyse the wormhole attack behaviour in ad hoc network by using NS2 simulator.
RQ3	RO3	To discuss the impact on Ad hoc network by comparing the results with and without wormhole attack.

RO 1: To study wormhole attack on Ad hoc network.

Depth study will be performed to completely understand the behaviour of wormhole attack in ad hoc network

RO2: To analyse the wormhole attack behaviour in ad hoc network by using NS2 simulator.

The existence of wormhole attack in ad hoc will be detected by using NS-2.

RO3: To discuss the impact on Ad hoc network by comparing the results with and without wormhole attack.

In the end of this project, the discussion will be cover the comparing result of with and without wormhole attack.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

1.5 Project Scope

Scope of the project will be conducted as follow:

- i. Thoroughly research on active attack which is wormhole attack on wireless ad hoc network.
- ii. Focusing on the metrics used to measure the performance of the network which are delay, throughput, normalised routing load.
- iii. Analysis of the wormhole attack is evaluated using NS-2

1.6 Research Contribution

As the wormhole attack can have serious impact. This paper contributes in the study of the behaviour of the wormhole attack in depth so that better prevention mechanisms can be employed against this attack. Also, in the end of this research will present a graph theoretic model for characterising the wormhole attack by using network simulation-2 (NS-2).

1.7 Project Organisation

This project report consists of seven chapter which are Chapter 1: Introduction, Chapter 2: Literature review, Chapter 3: methodology, Chapter 4: design, Chapter 5: implementation, Chapter 6: result and finding and lastly Chapter 7: conclusion

Chapter I: Introduction

This chapter will concentrate on introduction, project background, research problem, research question, research objectives, scopes, project contribution and report organisation.

Chapter II: Literature Review

This chapter will focus on details about this project. It will supported by studying previous research paper and other reading materials.

Chapter III: Methodology

This chapter will focus on method that will be handle this project. Waterfall method will be used in this project so that this project can be organised smoothly.

Chapter IV: Design

This chapter will describe about the design that will be handle this project. This chapter also will focus on possible scenario and metric measurement used for this project.

Chapter V: Implementation

This chapter will explain and focus about the environment setup. It will explain briefly for every phase that need to be done in this project.

Chapter VI: Result and Finding

This chapter will concentrate and explain all the result from the project. This chapter also include the recommendation for the project.

Chapter VII: Conclusion

This chapter will explain and summarize about the project summarization, project contribution, project limitation and future work of this project.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

1.8 Expected Output

In the end of this project, the graphs will be presented using NS-2 by comparing the results with and without wormhole attack on ad hoc network. Discussion will be made by comparing the performance of Ad hoc network based on three metric which are throughput, end-to-end delay and normalised routing load.

1.9 Conclusion


As conclusion, through this chapter research objectives and scopes have been determined based on research question. Project contribution for in the end of this project also has been determined based on expected output. The next chapter will concentrate on literature review which will be covered on materials reading.



CHAPTER II

LITERATURE REVIEW

2.1 Introduction



This chapter mainly will concentrate on the previous work related to wormhole attack on ad hoc network. This chapter will answer the objectives stated in chapter 1. This chapter will come out with several issues related to this research such as ad hoc wireless network, security, wormhole attacks and routing protocols. It also include the analysis techniques, pattern, simulation and scenario from previous research.

This chapter will help to understand in-depth the behaviour of wormhole attacks in ad hoc networks. It also will help in determine the techniques, pattern and simulation that will be used in this project by comparing several research from previous works.

2.2 Ad hoc wireless network.

For a past few decades, wireless communication is one of the fastest communication technology that has been developing exponentially. We have seen so many advance wireless devices such as laptops, cell phones and PDA (Goldsmith, 2004). Until today, wireless network in planned and sized infrastructure and its operations especially for personal and local networks in order to full fill the need of self-organisation, adaptability, cost reduction and independence.

For the next generation of wireless communication system, the fast implementation is needed for the independent mobile users. This is because there are some network scenario that cannot depends on centralised and organised connectivity such as military networks, disaster recovery and communication foe emergency operations (Goldsmith, 2004).

Wireless mobile networks is categorised into two types. The first one “one-hop” wireless network which are wired gateways and infrastructure networks with fixed. Wireless local area networks (WLANs) is one of the application for this type. The second one is Ad hoc wireless network where the infrastructure is less mobile network.

Ad hoc wireless network is one of the wireless mobile node that form a network without established infrastructure as shown in Figure 2.1 (Goldsmith, 2004). Ad hoc wireless network handle all the essential control and networking task by themselves.

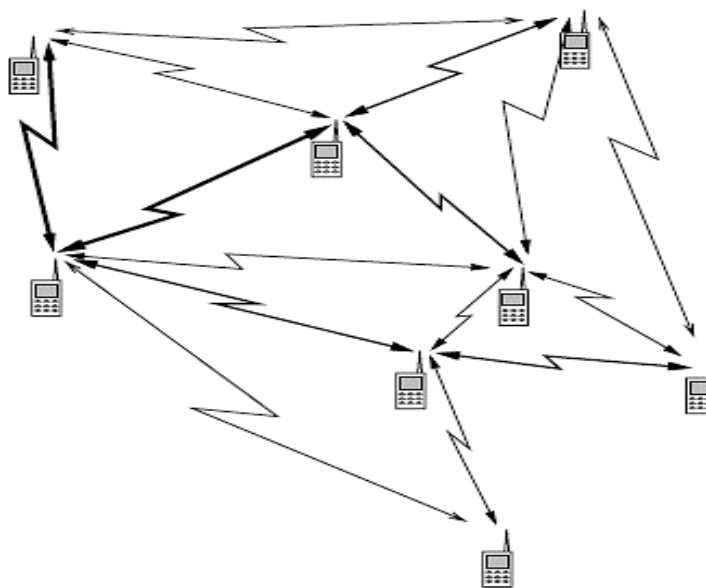


Figure 2.1: Ad hoc Network

Ad hoc is characterise as being inherent to bandwidth, reduced autonomy and auto-configuration. Because of these characteristics, ad hoc network is exposed to attack. So, when faced with attacker, those characteristics makes ad hoc network exposed to attack and offer more faults than other types of network.

2.3 Classification of wireless networks.

This section will briefly explain the types of wireless networks to show the difference between ad hoc networks and the others existing networks.

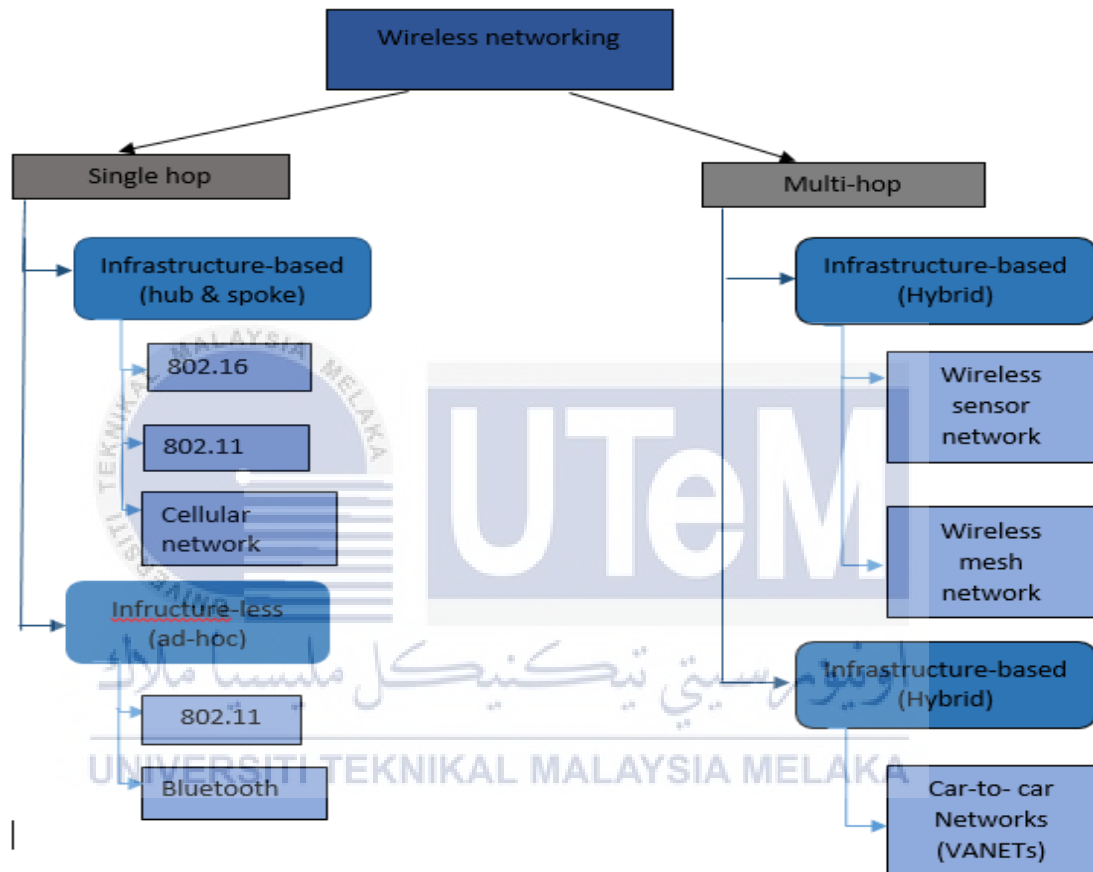


Figure 2.2: Taxonomy of wireless network

2.3.1 Cellular network

Cellular network is a combination of radio cells. Each cell consists of a base station that acts as a fixed location transceiver. This combination of cells offers radio coverage over larger geographical areas. So, we can say that a cellular network works with the existence of a base station. In conclusion, a cellular network is an infrastructure-based network, while an ad hoc network is not an infrastructure-based network.

2.3.2 Wireless Mesh Networks

A wireless mesh network is a communications network that combines radio nodes organized in a mesh topology. Wireless mesh networks usually consist of gateways, mesh clients, and mesh routers.

2.3.3 Wireless Sensor Networks

Wireless sensor networks consist of thousands of distributed autonomous sensors, where each is capable of performing some limited computation, communication, and sensing. These sensors are used to answer any queries about the environment.

2.4 Classification of routing protocols.

In ad hoc networks routing act as transferring data from source to destination. Basically, there are two activities involved in routing concept which are determining the best routing path and transferring the data through ad hoc network (Otmami & Ezzati, 2014).

Routing protocols of MANETs can be categorised into three categories:

- Proactive protocols

In this type of routing protocol, each node in a network maintains one or more routing tables which are updated regularly. Each node sends a broadcast message to the entire network if there is a change in the network topology. The example of proactive protocols are Distance vector (DV) protocol, Destination Sequenced Distance Vector (DSDV) protocol, and Optimized link state routing protocol (OLSR) protocol.

- Reactive protocols

In this type of routing protocol, each node in a network discovers and maintains a route based on-demand. It broadcast control message during discovering a route and bandwidth is used for data transmission once route is discovered. Dynamic Source Routing (DSR), Ad-hoc On Demand Routing (AODV) and Associativity Based Routing (ABR) protocols are examples for this type protocol.

- Hybrid Protocols

Combination of proactive protocols and reactive protocols.

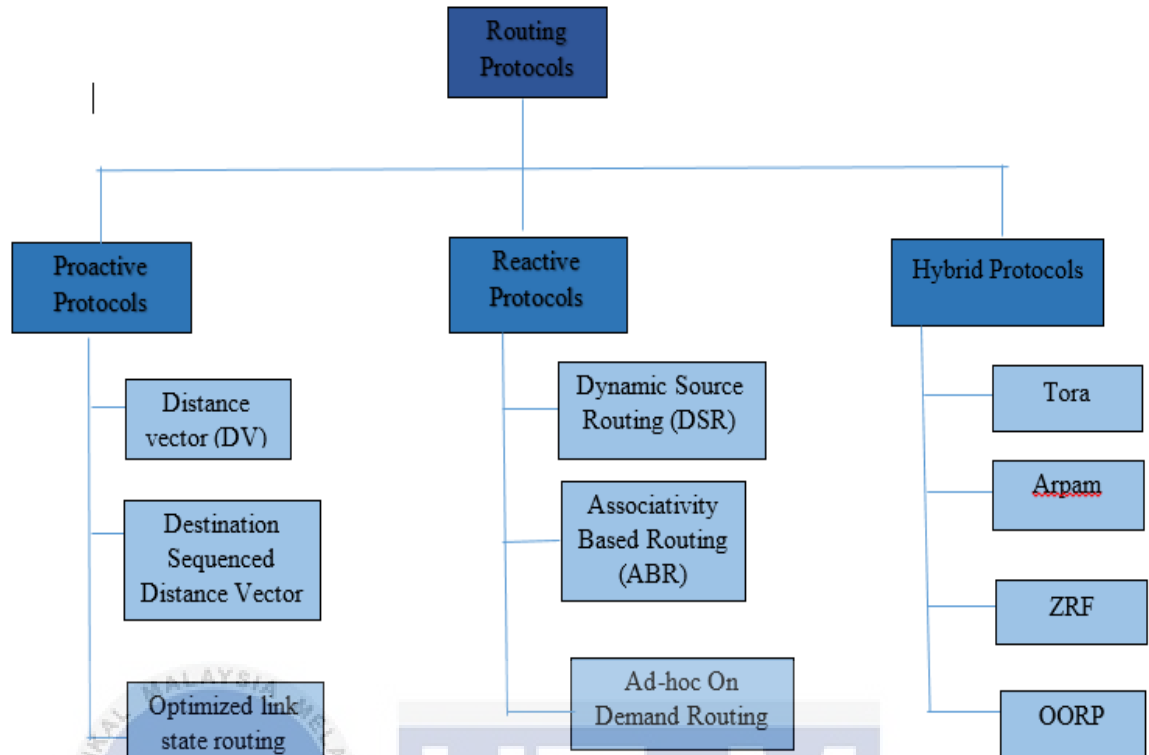


Figure 2.3: Taxonomy of routing protocol

2.5 Ad hoc On Demand Routing (AODV)

2.5.1 Route Discovery

AODV is an environment where any source node wants to send packet to destination node. Source node will broadcast route request packets to all its accessible neighbours. During the process the intermediate node will check the receiving request (RREQ) whether the destination is for intermediate node or not. If the destination meant to him it will send route reply message (RREP). If not, the request will be forward to another node. Before the packet is forward, each node

stores the broadcast identifier and the node number from request node. The broadcast identifier, source ID used to prevent the redundant request receives the same nodes by detecting whether the node already received the previous route request message or not. The source node will determine which message will be selected from many received reply by using hop counts (Gupta, Priyanka, & Upadhyay, 2012).

Table 2.1: RREQ Field

Source address	Source sequence	Broadcast Id	Destination Address	Destination Sequence	Hop Count
----------------	-----------------	--------------	---------------------	----------------------	-----------

Table 2.2: RREP Field

Source address	Destination Address	Destination Sequence	Hop Count	Lifetime
----------------	---------------------	----------------------	-----------	----------

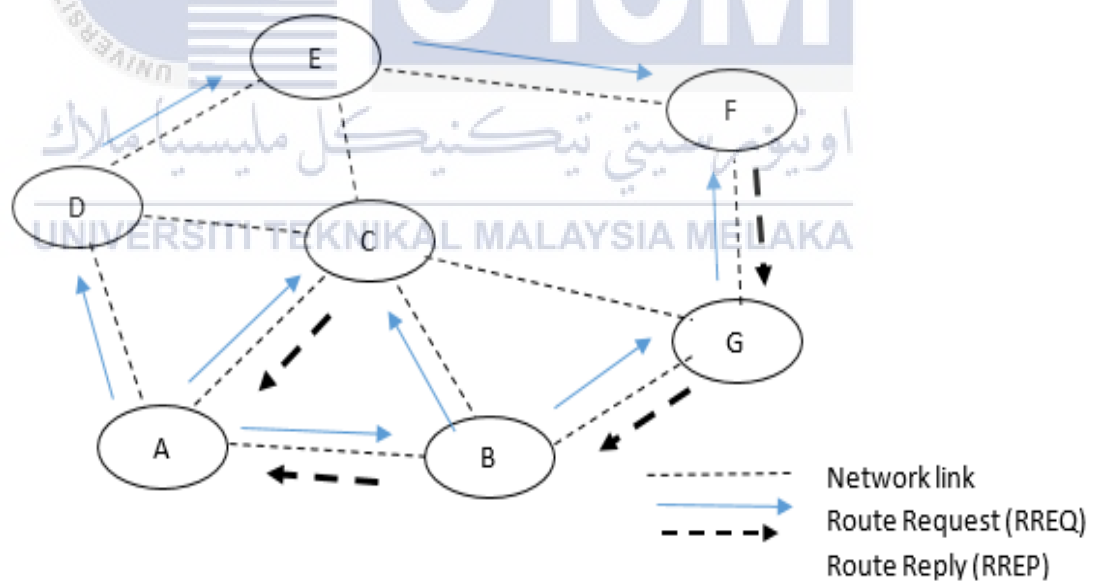


Figure 2.4: Route Discovery in AODV

2.5.2 Route Maintenance

Route maintenance is happened when any link breaks down due to the node mobility, the node will invalidate the routing table. Loss of the link can cause the all destination become unreachable. A route error (RERR) message will be created. Then the node will send RERR upstream to the source node. Source node will reinitiate route discovery when it receives route reply message if it still requires the route (Gupta et al., 2012) .

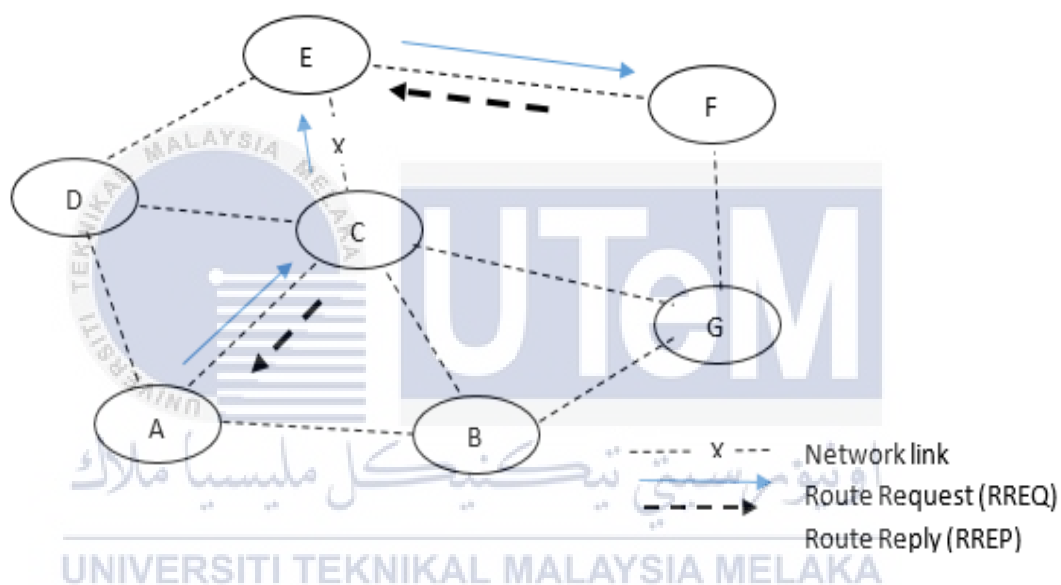


Figure 2.5: Route Maintenance in AODV

2.6 Security Attacks in Ad Hoc Networks

Generally, attack is defined as an attempt to destroy or disrupt the normal functionality of the network and disrupt the basic security goals such as integrity, availability confidentiality, non-repudiation and authentication (Shrivastava & Dubey, 2015). Attacks in ad hoc networks can be categorised into two categories which are passive attack and active attack.

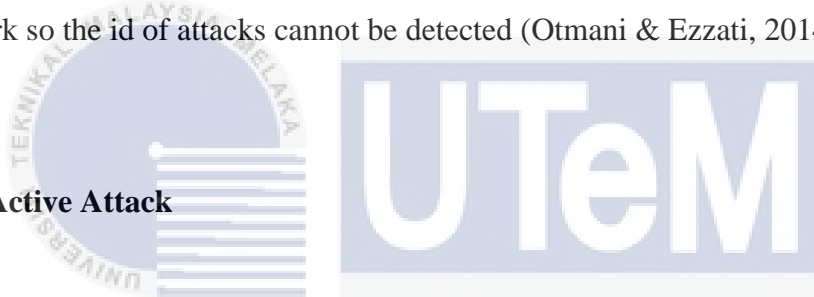
2.6.1 Passive Attack

Passive attacks are the attack that does not destroy a normal operation of network so the id of attacks cannot be detected (Otmani & Ezzati, 2014).

2.6.2 Active Attack

An active attacks are the attack that disrupt the data transmission in network by interrupting the normal functionality of network (Otmani & Ezzati, 2014)

UNIVERSITI TEKNIKAL MALAYSIA MELAKA



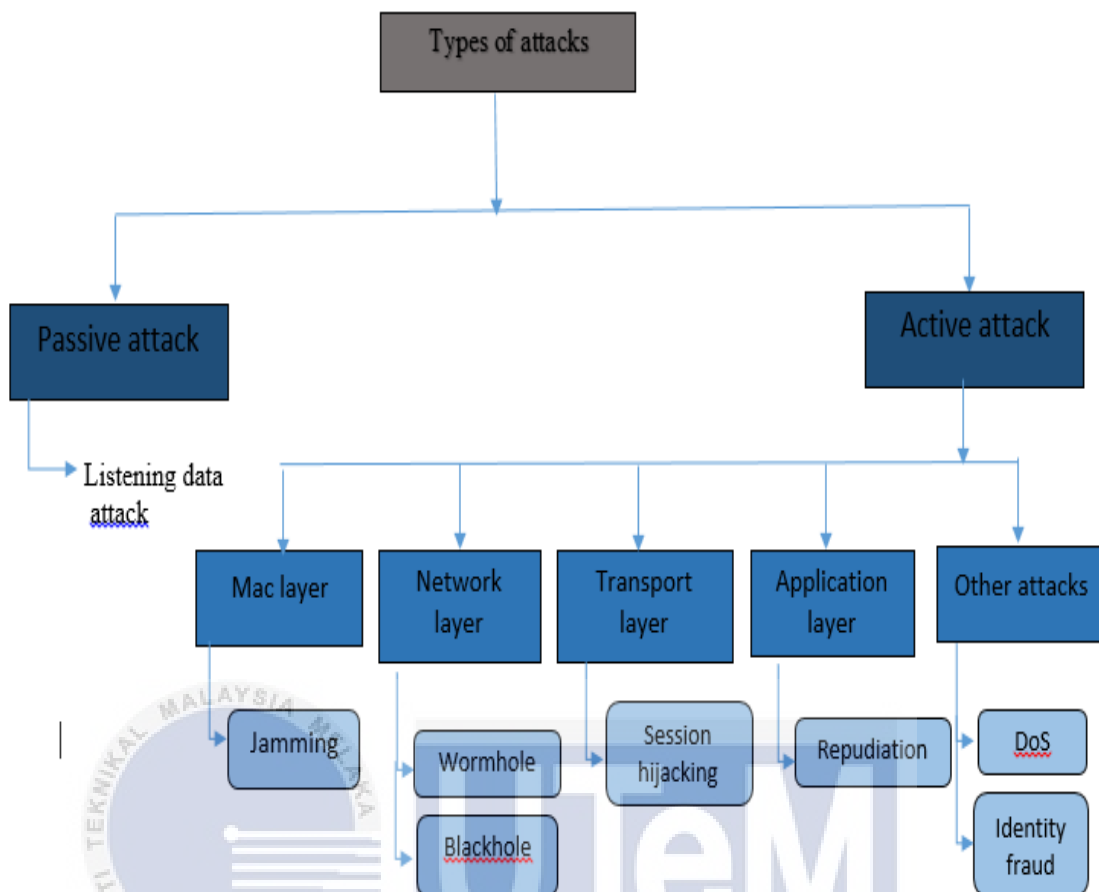


Figure 2.6: Taxonomy of attacks in ad hoc

Table 2.3: Types of attacks in ad hoc

Type of attacks	Attacks	Description
Passive	Traffic Monitoring	<ul style="list-style-type: none"> Developed to identify the communication parties and functionality which could provide information to launch further attacks.
	Traffic analysis	<ul style="list-style-type: none"> Used to gain information on which nodes communicate with each other and how much data is processed.
	Syn flooding	<ul style="list-style-type: none"> Attacker may repeatedly make new connection request until the resources required by each connection are exhausted or reach a maximum limit
Active	Black hole	<ul style="list-style-type: none"> Attacker advertises a zero metric for all destinations causing all nodes around it to route packets towards it. A malicious node sends fake routing information, claiming that it has an optimum route and causes other good nodes to route data packets through the malicious one. A malicious node drops all packet that it receive instead of normally forwarding those packets. An attacker listen the requests in a flooding based protocol.
	Wormhole attack	<ul style="list-style-type: none"> In a wormhole attack, an attacker receives packets at one point in the network, “tunnels” them to another point in the network, and then replays them into the network from that point.

	Sybil attack	<ul style="list-style-type: none"> Attacks will forge the result of a voting used for threshold security methods by acting as virial of identities/nodes rather than one.
	Gray-hole attack	<ul style="list-style-type: none"> This attack have two phases which are: at first phase node will advertise itself as having a valid route to destination. Second phase, nodes will drop the intercepted packets. This attack leads to drop of message.
	DoS	<ul style="list-style-type: none"> Attacker aimed at complete disruption of routing information and therefore the whole operation of ad-hoc network.

2.7 Wormhole Attack

Akansha Shrivastava and Rajni Dubey said that wormhole attack is an attack where two nodes of attacker are connected to each other through the link known as tunnel (Shrivastava & Dubey, 2015). The attacker node at one side captures the packet from the normal node and transferring the packet through the tunnel then transmit it to the other attacker node or malicious node present in the network. It creates the illusion tunnel between two malicious nodes.

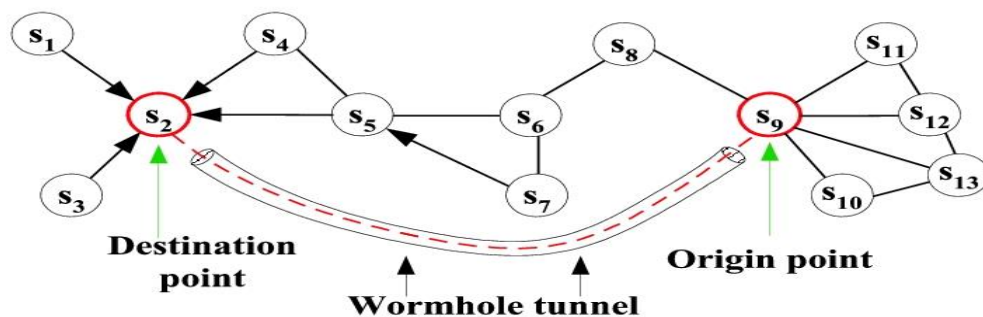


Figure 2.7: Network affected by wormhole attack

In Figure 2.7 shows the tunnel created by attacker. It is where the routing packet being encapsulated. The tunnel can be represented by wired link, wireless out-of-bank link and logical link (Upadhyay & Chaurasia, 2011). In AODV protocol, wormhole attack can implied by transferring each route request (RREQ) packet directly to the destination target node of the request by using the tunnel. When the destination node's neighbours hear this request packet, they will follow normal routing protocol processing to rebroadcast that copy of the request and then discard without processing all other received route request packets originating from the same route discovery.

2.8 Implication of wormhole attack to Ad hoc network.

The impact of wormhole on the network is very serious. It will affect the whole performance of the network by decreasing throughput of network by dropping the data packets. Not only that, this attack also affect the other parameters of network (Shrivastava & Dubey, 2015).

In wormhole, the attacker at one end records the incoming traffic and tunnels packets to the other end. If routing control messages like RREQ are tunnelled, this will result in disrupt routing tables in the network. If fast transmission path exist between two nodes of wormhole, it will tunnel the data at higher speed than the normal mode of wireless multi-hop communication. Thus, they will attract more traffic from their neighbours (Upadhyay & Chaurasia, 2011).

2.9 Related Work

Table 2.4: Justification from previous works

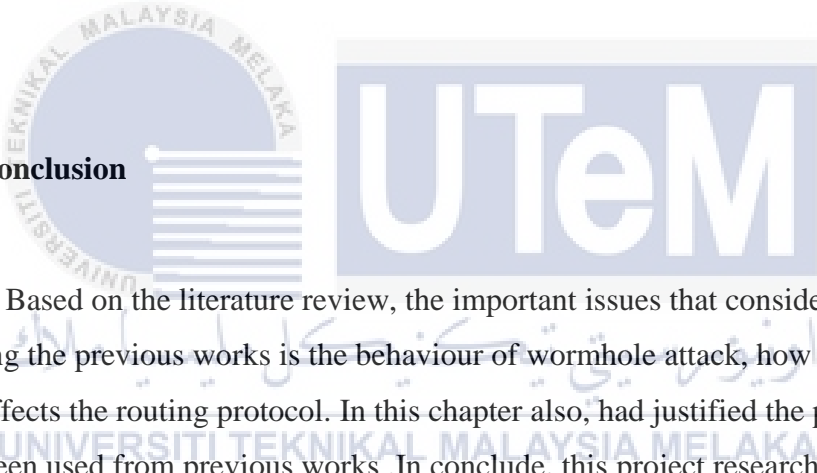
Paper	Parameter				
	Packet drop	Throughput	Delay	Jitter	Packet loss
(Upadhyay & Chaurasia, 2011)	/				
(Kaur, Sandeep, & Dhanda, 2013)		/	/		
(Safwani, Hassan, & Kadhum, 2011)		/	/	/	
(Win, Physics, & Gyi, 2008)	/				
(Gupta et al., 2012)	/	/	/		
(Lu & Lu, 2003)		/	/		/
(Arfaat, 2011)		/	/		/

2.10 Summarization

For this project, the performance of ad hoc network with and without wormhole attack will be tested based on parameter end to end delay, throughput and normalised routing load.

For end to end delay the average time taken for packet to reach the destination from source is calculated. For throughput, the number of data packet that successfully delivered from source to destination will be calculated. For a normalised routing load, the number of routing packets and number of data packet will be calculated.

2.11 Conclusion




Based on the literature review, the important issues that considered when studying the previous works is the behaviour of wormhole attack, how the attack does effects the routing protocol. In this chapter also, had justified the parameter that have been used from previous works .In conclude, this project research will cover on the analysis of wormhole attack in ad hoc networks. Next chapter will cover about methodology used for this project. It includes the Gantt chart and milestone for PSM 1.

CHAPTER III

METHODOLOGY

3.1 Introduction



On this chapter, will be cover about methodology and the analysis phase. Methodology itself is a guideline to solve the problem when doing a project with specific method and phase when the project is start and end. In analysis it break through to a specific category such as experimental, testing, simulation and surveying. A few techniques will be used in order to complete this project. All this technique is followed the project requirement.

3.2 Project Methodology

This section will discuss about methods and principles that will be used to complete this project. It will define every phase that will be taken to complete this project. Waterfall methodology approach which contains five phases will be used to conduct this project. This approach will define the problem statement and objective of the research project, initial study and literature review on the research problem, data collection based on perimeters will be used, analysis reporting module of the problem, and last but not least documentation and publication.

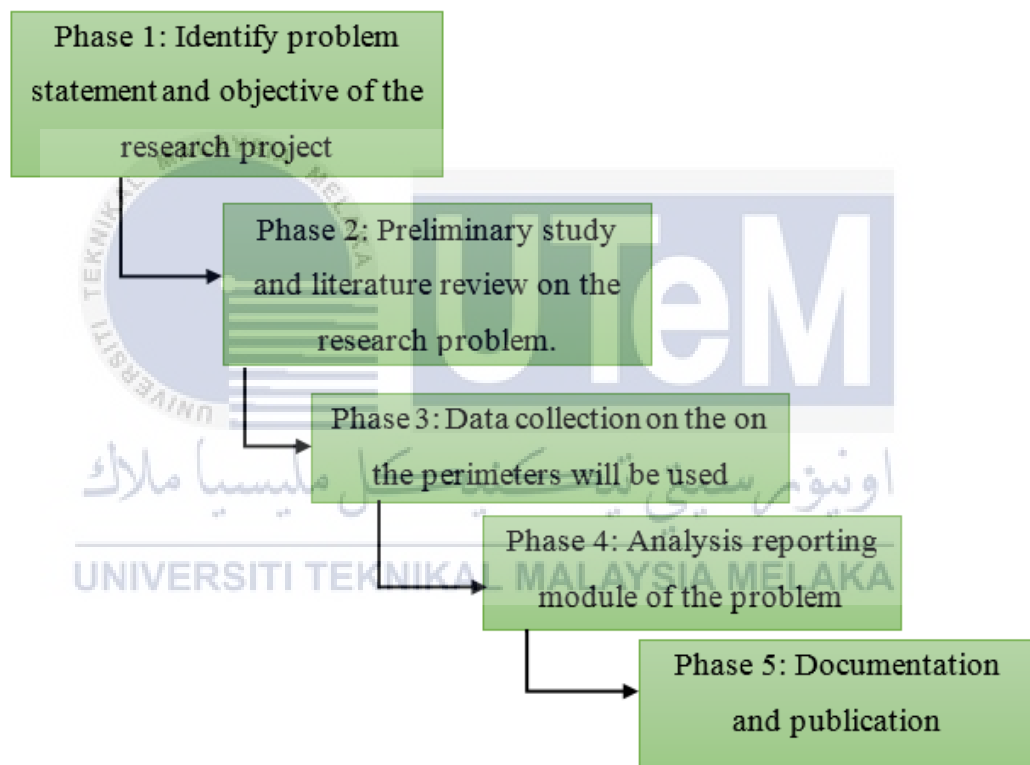


Figure 3.1: Waterfall model

Phase 1: Identify problem statement and objective of the research project

In this phase, problem statement, objectives, scope, and project expectation that conduct this project already explained in Chapter 1. All collection of required data and source for this project are organised and presented as shown in Chapter II. In order to complete this project as planned a Gant Chart and Milestone are created. All activities need to be done and period taken for each activities are stated in this tools.

Phase 2: Initial study.

In this phase, in-depth study are conducted in order to gain detailed information about wormhole attack in ad hoc networks. The output for this activities is explained in Chapter II, literature review. All the information is gained from the previous works.

Phase 3: Data collection

In this phase, data collected from previous work will be used to design ad hoc topology in network simulation. Data collection also involve the data based on perimeters throughput, delay and packet loss that need to perform the result to achieve the objectives that has been stated.

Phase 4: Analysis result.

For this phase, the collected data will be analyse to perform the result. Both of the normal flow packet and with wormhole attack flow packet will be analyse to get the pattern of behaviour for both flow packet based on perimeters stated in phase 3. From the result both of behaviour, the differentiation can be seen between with and without wormhole attack packet flow.

Phase 5: Documentation

This is the last phase where all the activities in this projects will be documented and published in order to help future researcher to gain some additional information about wormhole attack in ad hoc networks.

3.3 Project Tool and Requirement

This section will define the project tools and requirement used for this project which are:

Table 3.1: Hardware and software requirements

Item	Specification	Usage
Operating system	<ul style="list-style-type: none"> • Ubuntu 14.04 	<ul style="list-style-type: none"> - To install NS-2.35
Software	<ul style="list-style-type: none"> • NS2.35 • Microsoft Office Word 2007 • Microsoft Office Project 2010 • Microsoft Office Visio 2007 	<ul style="list-style-type: none"> - To create topology - To documentation the project - To create Gantt Chart -To create flow chart
Hardware	<ul style="list-style-type: none"> • Personal Computer 	<ul style="list-style-type: none"> - For personal use

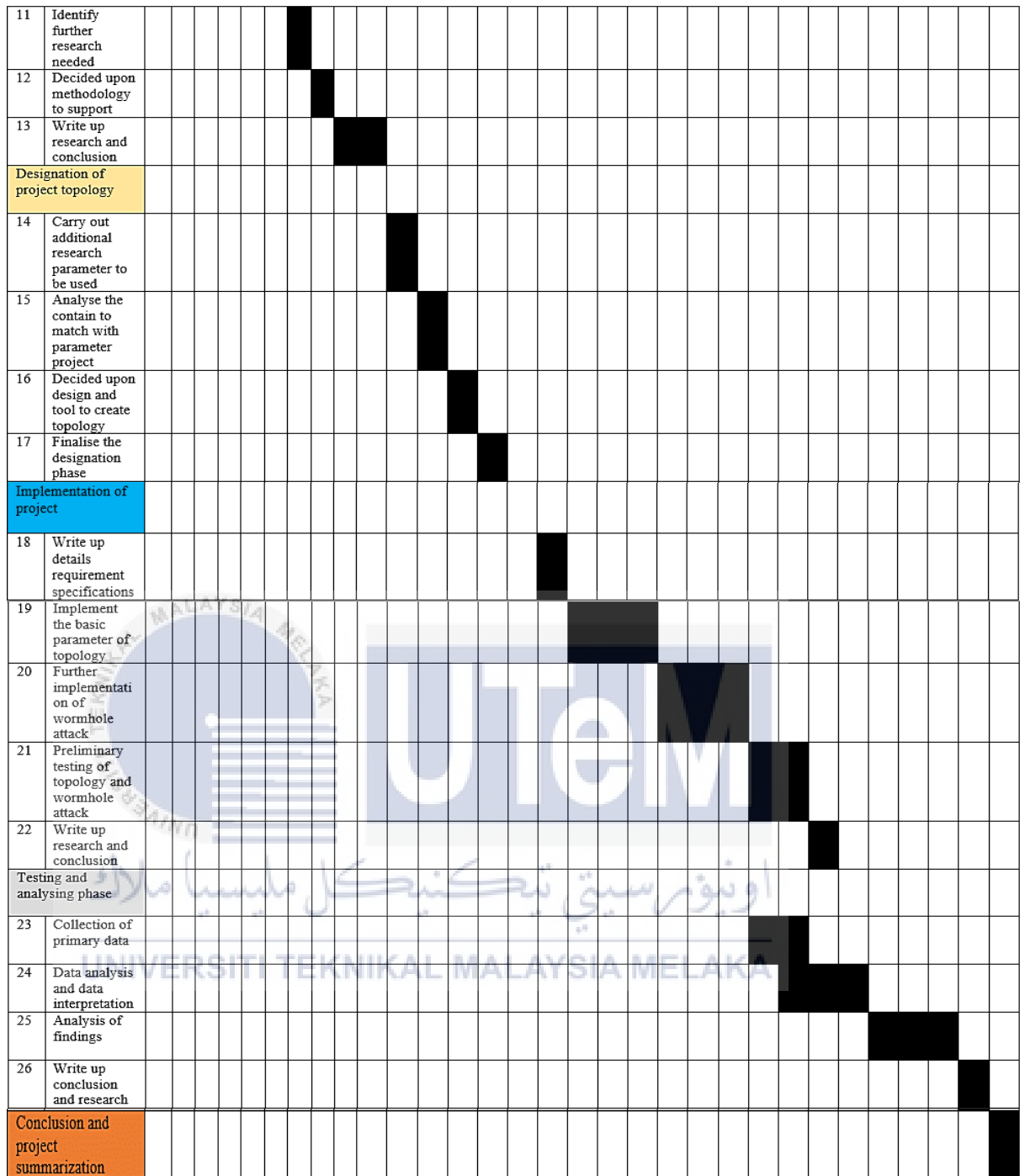


Figure 3.2: Project Gant Chart

3.5 Conclusion

Overall this chapter describe the Waterfall methodology used to complete this project. This methodology is chose because the step is suitable for this project that based on simulation. Gantt chart and milestone also are created based on plan of the whole PSM 1 activities and their period. Next, chapter IV will describe about the design of ad hoc topology in NS2 and the metric measurement used for this project.



CHAPTER IV

DESIGN

4.1 Introduction

This chapter will define the initial design of this project that will be applied. The design will cover on the network design, results of the analysis of the initial design and project implementation flow.

4.2 Network System Architecture

Network system architecture is an initial design that used before other stage in designing takes place. It is important to avoid any mistake during implementing the network.

4.2.1 Logical design

Basically, logical design is conducted by modelling of the actual system. Logical design is a graphical diagram that shows the flows of traffic between nodes. It able to present the task of system without specify the system.

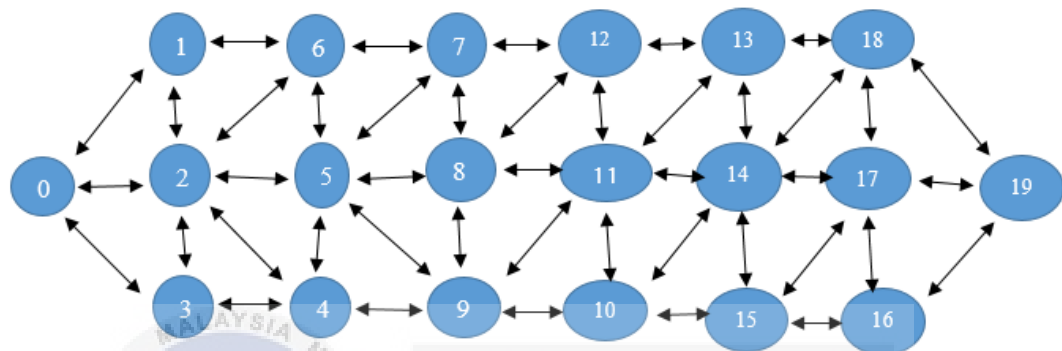


Figure 4.1: Ad hoc topology

4.2 Simulation Parameter

4.2.1 Simulation parameter from previous work

This section will discuss about the simulation parameter used from previous work. It can be guider for simulation parameter for this project.

Table 4.1: previous work of simulation parameter

Paper	Protocol	Simulation time	Simulation area	Number of nodes	Network traffic
(Gupta et al., 2012)	AODV	2000 sec	100*100m	16	CBR
(Kaur, Sandeep, & Dhanda, 2013)	AODV,DSR, ZRP,ANODR	150 sec	1500*1500m	10,20,30	CBR
(Lu & Lu, 2003)	AODV, DSR	1000 sec	1000*1000m	30	CBR
(Otmani & Ezzati, 2014)	AODV, DSR	1000 sec	1500*1500m	20	CBR
(Upadhyay & Chaurasia, 2011)	AODV,DSR	2000 sec	1500*1500m	50	CBR
(Win, Physics, & Gyi, 2008)	DaW, LF	900 sec	200*200m	16, 24, 32, 40, 48, 56	CBR

4.2.2 Project simulation parameter

Some constant values will be used for this simulation parameter of project so that the result will be equivalent.

Table 4.2: Simulation parameter

Parameters	Values
Ad hoc routing protocols	AODV
Radio propagation model	Propagation/ <u>TwoRayGround</u>
Network Interface Type	<u>Phy/WirelessPhy</u>
MAC Type	Mac/802.11
Interface Queue Type	Queue/ <u>DropTail/PriQueue</u>
Link Layer Type	LL
Simulation Time	150 sec
Routing Protocols Number of Nodes	20 nodes
Environment size	1000*1000
Traffic type	CBR
No of wormhole	2 (15, 9)
No of testing	7

4.3 Possible Scenarios

The possible scenario for this project is comparison between normal packets flows in ad hoc and with existing wormhole attack in ad hoc network. By comparing this, the result based on perimeter throughput, end to end delay and normalised routing load (NRL) will be show in graph.

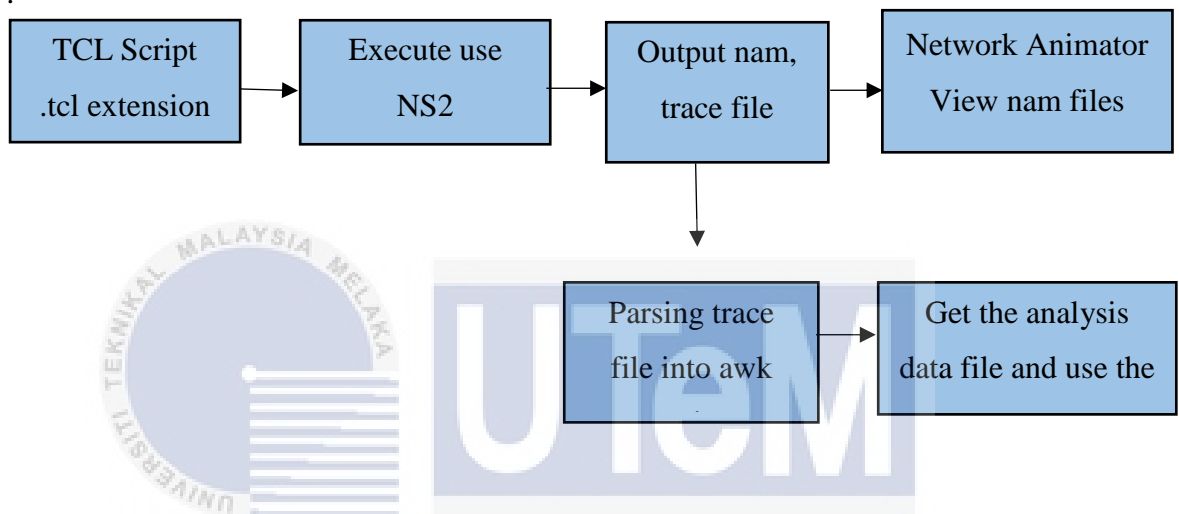


Figure 4.2: process in NS2 simulator

4.5 Metric Measurement

In this stage, the performance metrics will be used to calculate the ad hoc network performance. A programming script using AWK is created. In order to achieve the required results the programming script will be analysed to the NS-2 traced files. The performance metrics will be used are:

Throughput

Throughput is defined as measurement number of units' information that a system can process in a set amount of time.

Normalised Routing Load

Normalised Routing Load is defined as ratio of the total packet size of control packets (include RREQ, RREP, RERR and Hello) to the total packet size of data packets transferred to the destinations.

Packet end to end delay

Delay of packet is defined as the time taken to reach the destination after the packet is generated by source.



4.6 Conclusion

In this chapter, the early design phase have been discussed. Based on the design that have been decided the network environment for this project is created. For the next chapter, it will cover on the implementation of collected dataset based on perimeter used to the NS-2 simulator.

CHAPTER V

IMPLEMENTATION

5.1 Introduction

Relevant design that required for this project already explained in chapter 4. After all this chapter will describe about implementation phase. This phase includes collecting data for analysing the performance of Ad hoc networks with and without wormhole attack.

5.2 Simulation Setup

5.2.1 Simulation Overview

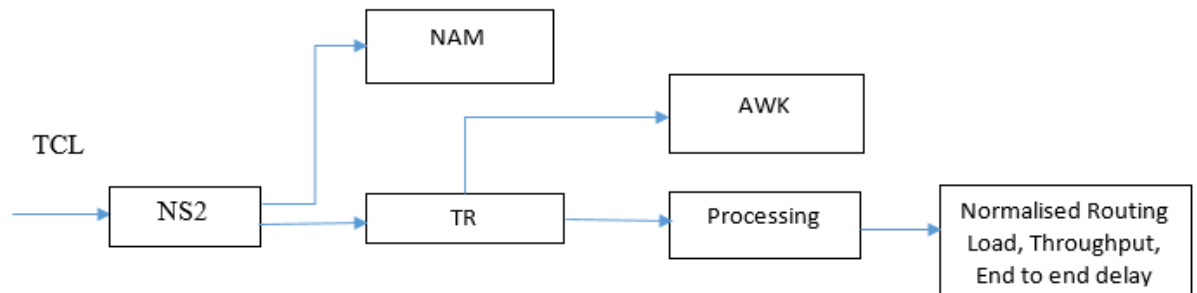


Figure 5.1: NS2 Overview

5.2.2 Simulation Parameter

The basic parameters for analysis activities are presented in Table 5.1 relevant to the simulation environment. For implementation phase, NS2 simulator and the Microsoft Excel 2013 is used as the tool for the analysis. The mobile Ad hoc network of 20 nodes is constructed in the NS2 with the area of 1000m x 1000m by using Tcl script. AODV routing protocol is used for this project as the protocol for the analysis.

The performance of Ad hoc network is analysed on parameter like throughput, end to end delay and normalised routing load.

Table 5.1: AODV Parameter for 20 nodes.

Simulation Parameter	Value
Simulator	NS-2 (V 2.35)
Topology Size	1000*1000
No. of nodes	20
Simulation Time	150 seconds
Traffic Type	CBR/UDP
Nodes of CBR connection	0,15,17,19,9,18
Packet Size	512
Packet Rate	100Kb
Parameter measurement	Throughput, End to end delay, Normalised Routing Load
No of wormhole	2 (15, 9)
No of testing	7

5.2.3 Wormhole Attack Implementation

In order to stimulate the wormhole attack, the wormhole attack simulation developed for NS2. The module is implemented as part of the NS2 Link Layer (LL) object which lies directly above the MAC layer. The modified LL files which are ll.cc and ll.h have several commands that allow it to be configured from the simulation TCL setup script.

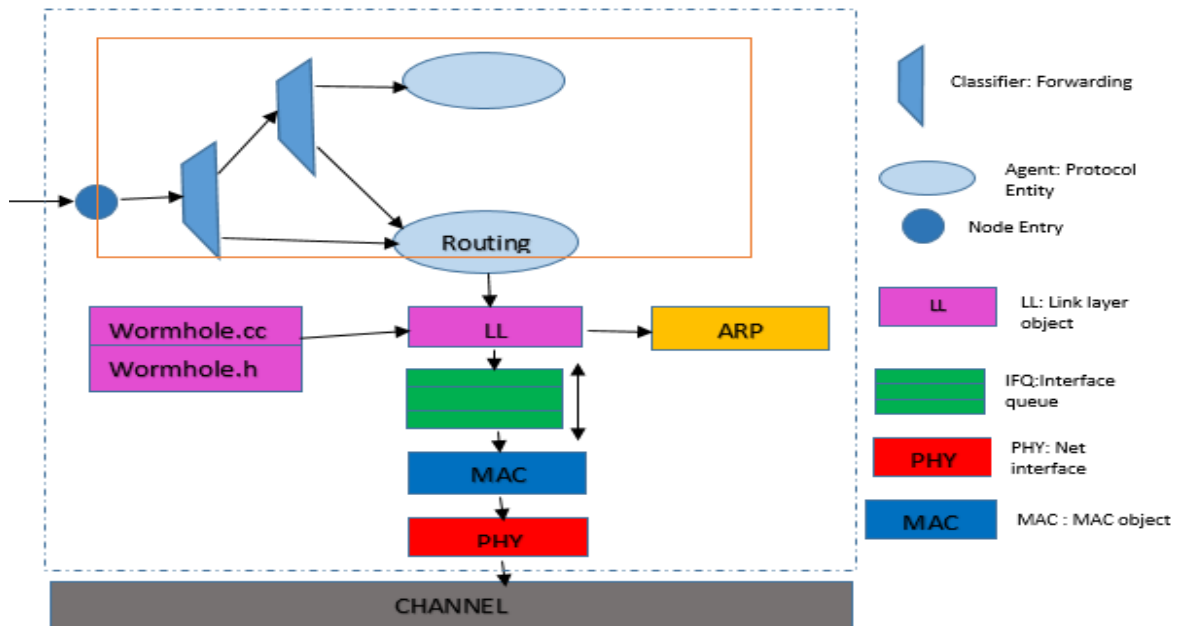


Figure 5.2: Mobile node schematic diagram

```

=====
# CREATING WORMHOLE NODES FOR SIMULATION
=====
$ns_ at 100.0 "$node_($val(wp1)) set ragent_ wormhole"
$ns_ at 100.0 "$node_($val(wp2)) set ragent_ wormhole"
[$node_($val(wp1)) set ll_(0)] wormhole-peer [$node_($val(wp2)) set ll_(0)]
[$node_($val(wp2)) set ll_(0)] wormhole-peer [$node_($val(wp1)) set ll_(0)]
#$ns_ duplex-link $node_($val(wp1)) $node_($val(wp2)) 512Mb 155ms at 50.0
DropTail
$ns_ at 100.0 "$node_($val(wp1)) label \"wormhole\""
$ns_ at 100.0 "$node_($val(wp2)) label \"wormhole\""

=====
# End wormhole
=====

```

Figure 5.3: Wormhole attack implementation on Tcl script

5.2.4 Recompilation of NS-2

After changes in the AODV directory recompilation is needed. First change the path to ~/ns-allinone-2.35/ns-2.35 and do ./configure (the Makefile will be replaced with modified one).

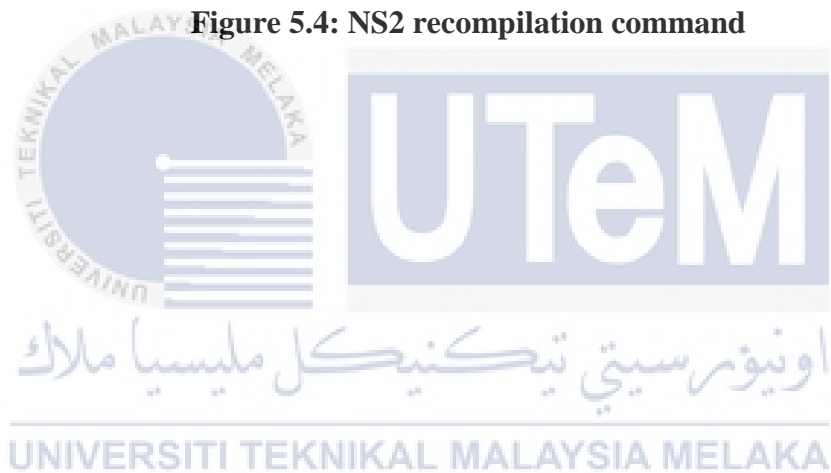
```
$make clean (it will recompile the whole ns2 and remove all the object files in NS2).
```

```
$make (removes all object files which are old and generates new object files).
```

```
#make install (login as super user)
```

When the compile is finished, run the tcl script using

Figure 5.4: NS2 recompilation command



5.2.5 Generate NAM Network Animator

NAM animator is an animation tool based on Tcl script for viewing network simulation traces and real packet traces. It provides packet level animation, various data inspection tools and topology layout.

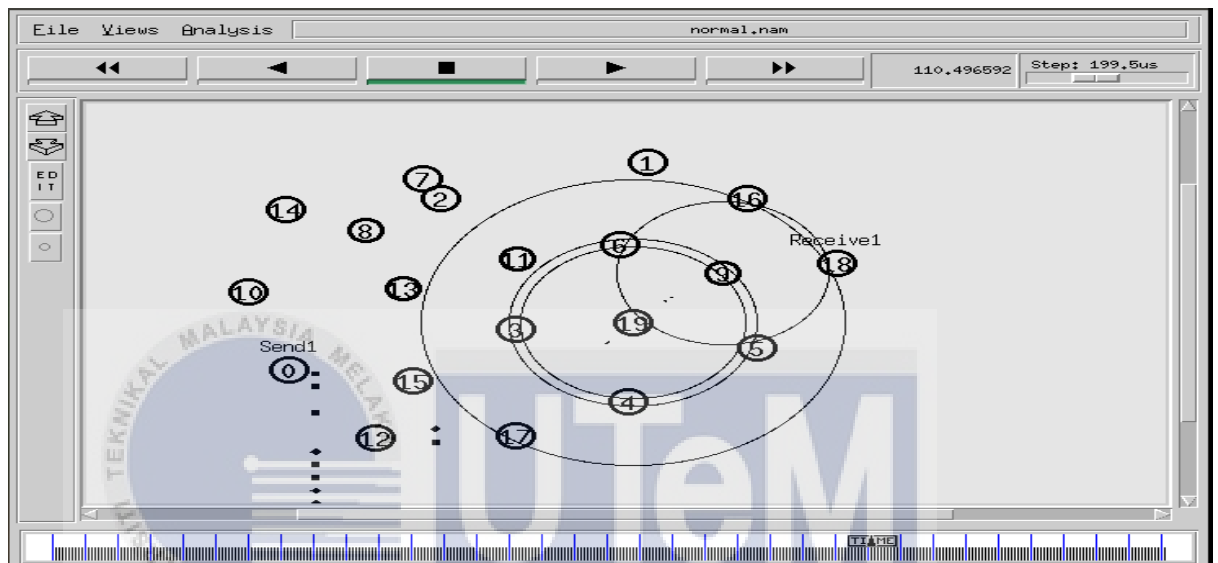


Figure 5.5: NAM animator without wormhole attack

Figure 5.5 shows the NAM animator without wormhole attack. Node (0) acts as a sender while node (18) acts as a receiver. When the sender want to sends packet to receiver, the node of sender will look the available path to the receiver in its local routing table. If there no path exists then it will broadcasts a RREQ message to its neighbourhood nodes. The discovered routes is node (0) > node (15) > node (17) > node (19) > node (9).>node (18).



Figure 5.6: Path discovered

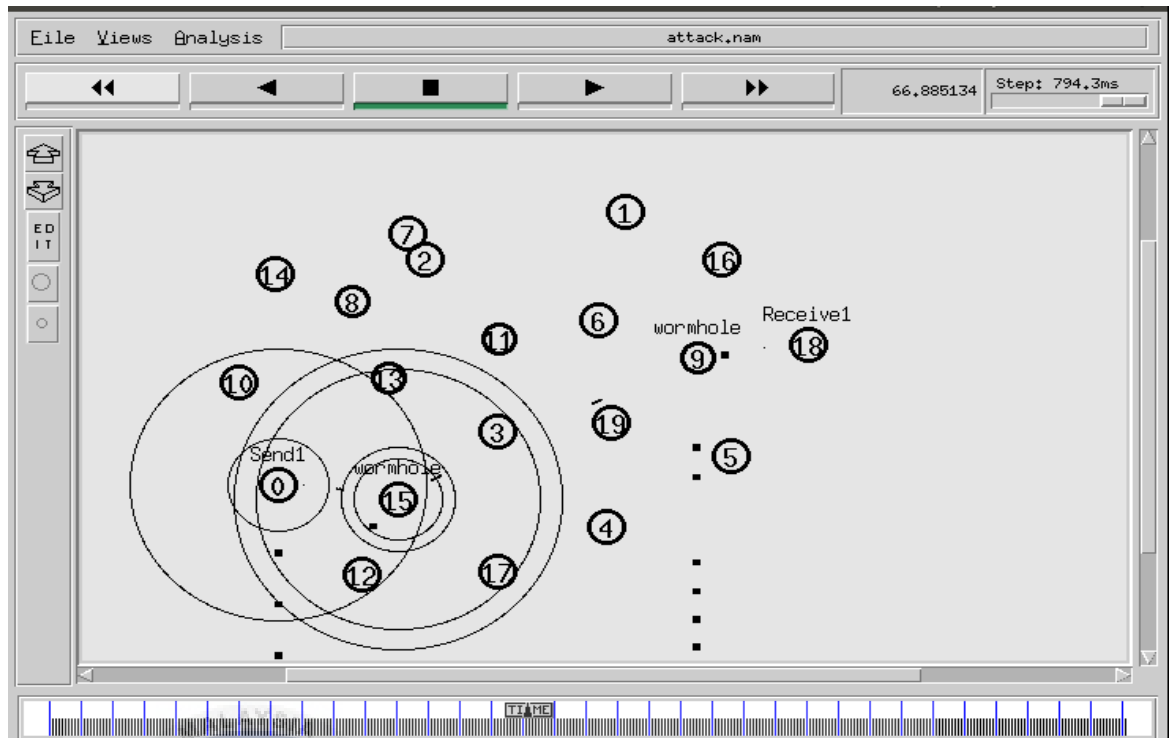


Figure 5.7 NAM animator with wormhole attack

Figure 5.6 shows the figure NAM animator with wormhole attack that interrupts the usual routing packet flow. Basically, this attack can be done by two nodes that connected via link called “wormhole link”. In this project, these two nodes are located near the source node and near to the destination which are node (15) and node (9) respectively and disrupting proper routing.

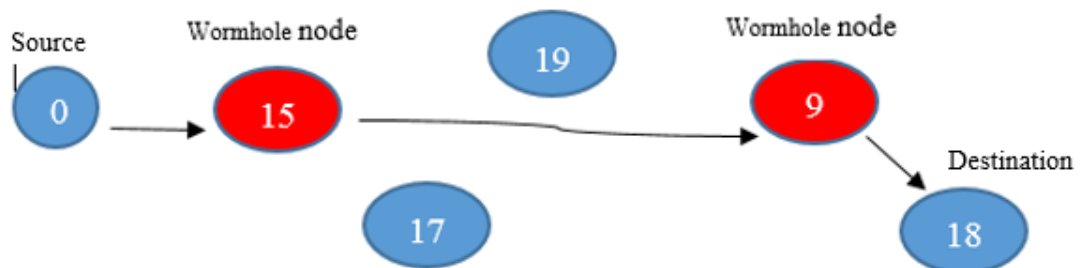


Figure 5.8: Path discovery with wormhole attack

5.3 Conclusion

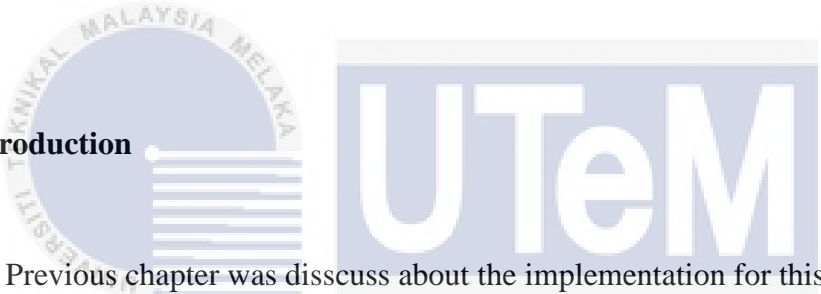
This chapter explained in detail about the implementation of wormhole attack as one of the malicious attack. This chapter helps to understand how the attack is working. In chapter 6, a further research about analysing and testing between normal packet flow and packet flow with wormhole attack.



CHAPTER VI

TESTING AND ANALYSIS

6.1 Introduction



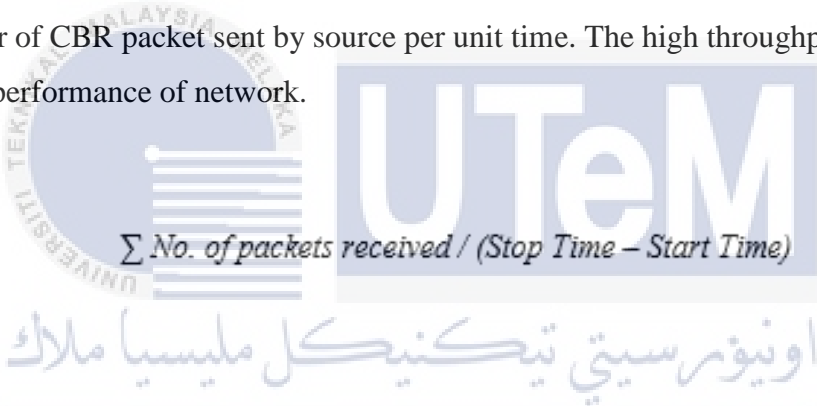
Previous chapter was discuss about the implementation for this project. Its included the creating of Ad Hoc network using Tel script and implementation of wormhole attack. In this chapter, will be focus on the testing and analysis phase. The data had been collected in two different situation which are normal packet flow and packet flow with existing of wormhole attack. This phase is very important because it will determine the achievement for this research project. All the data will be presented in the graph form.

6.2 Result and Analysis

The data was collected based on different scenarios which are normal packet flows and packet flow with attack. The data was collected by using AWK tool. AWK tool will trace all the data in tracefile from both scenarios. After compile the AWK, all the data will be collected and generated by using Microsoft Office Excel 2013.

6.2.1 Metric measurement 1: Throughput (bits per second)

Throughput is described as the rate of total CBR packet received to the total number of CBR packet sent by source per unit time. The high throughput leads to the better performance of network.



$$\sum \text{No. of packets received} / (\text{Stop Time} - \text{Start Time})$$

```

for(seqno = 0; seqno < 2; seqno++) {
if (event == "r" && layer == "AGT") {
    count0++
    if (!startTime || (time < startTime)) {
        startTime = time;
    }
    if (time > stopTime) {
        stopTime = time;
    }
    recvdSize += pkt_size;
    throughput= recvdSize/(stopTime-startTime*0.008);
}}
}
END {
startTime = time;
printf("%f\t%2f\n",startTime,throughput);
}

```

Figure 6.1 : AWK Script for Throughput

AWK Scriting tool is used in order to analyze the parameter of network with presence and absent of wormhole attack. The Figure 6.1 shows the AWK script that will trace data from trace file in order to calculate the throughput of network. From throughput AWK scripting it shows that information is collected from specific data in trace file which is receiving event in AGT layer. The total throughput will be calculated at the time of CBR traffic stop which is at 150 seconds.

Table 6.1 below shows the data collected for throughput after the AWK script is running

Table 6.1: Data collected for throughput

Time (seconds)	Throughput	
	Without Attack	With Attack
30	0	0
40	8475.866324	6411.313616
50	13752.80704	10425.60773
60	17238.81738	12871.01778
70	19849.29246	14838.59742
80	21701.61575	16172.0535
90	23134.20568	17427.69691
100	24323.32919	18427.70355
110	25266.04816	19038.00357
120	26084.63524	19783.5081
130	26708.40292	20151.95718
140	27305.36571	20681.52354
150	27832.71594	20948.34066

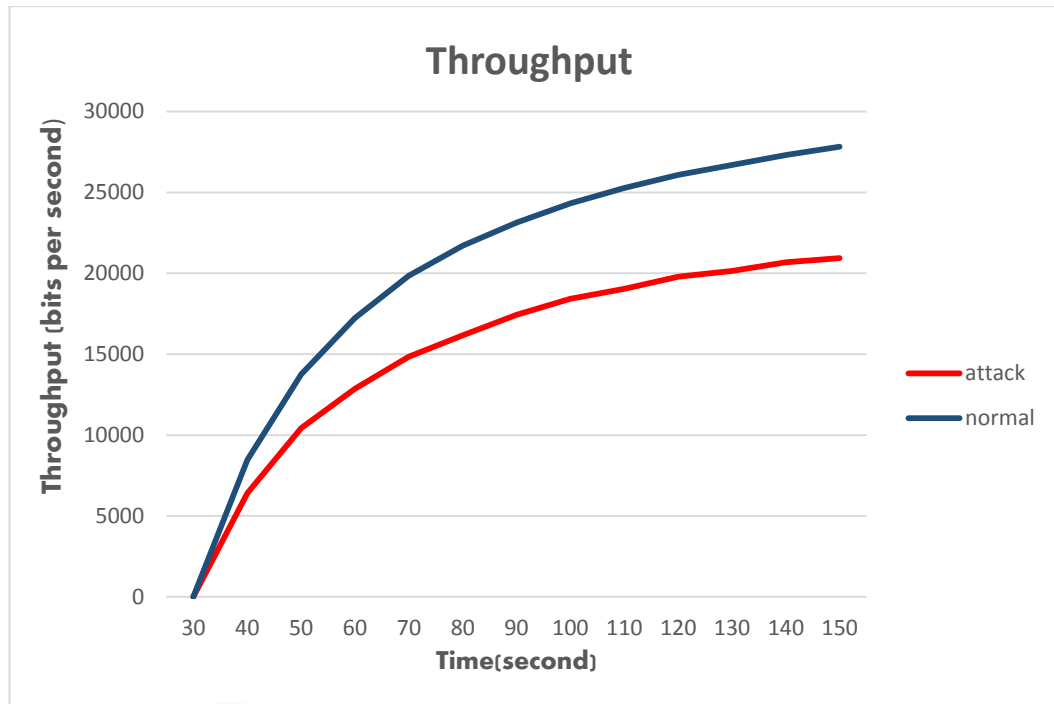


Figure 6.2: Graph of throughput

Based on Figure 6.2, it shows the throughput for network for both with and without wormhole attack keep increasing start from 30 seconds until 150 seconds. However, the value of throughput obtained from network with wormhole attack is lower than without attack.

This is happened when the traffic is routed through wormhole node, the attacker will gain full control over the traffic. The wormhole node transfer the data through the tunnel. Thus in this process it put large number of the data packet in its queue to process the large number data and while processing all the data it reduces the number of packet which will decrease the throughput of network.

6.2.2 Metric measurement 2: End To End Delay (second)

Delay is defined as the average time taken by a data packet to arrive at destination from the source. Only the data packets that are successfully delivered to destination are counted. Low value of delay leads to the better performance of network.

$$\text{(Arrive time - send time)} / \sum \text{number of connection}$$

```

if(layer == "AGT" && event == "s" && seqno<$6){
  seqno = $6;
}
if(layer == "AGT" && event == "s"){
  start_time[$6] = $2;
}
else if((type == "cbr") && (event == "r")){
  end_time[$6]= $2;
}
else if(event == "D" && type == "cbr"){
  end_time[$6]= -1;
}
}
END{
  for(i=0; i<=seqno; i++){
    if(end_time[i] > 0){
      delay[i] = end_time[i]-start_time[i];
      count++;
    }
    else
    {
      delay[i] = -1;
    }
  }
  for(i=0; i<=seqno; i++){
    if(delay[i] > 0){
      n_to_n_delay = n_to_n_delay + delay[i];
    }
  }
  n_to_n_delay = n_to_n_delay/count;
  startTime = time;
  printf("%f\t%f\n", startTime, n_to_n_delay);
}

```

Figure 6.3 : AWK Script for delay

In order to analyze the effect of wormhole attack on delay, the AWK script is used as shown on Figure 6.3. From the AWK script it will trace information from specific data which are from sender event in AGT layer while receive event at cbr traffic and drop event.

Table 6.2: Data collected for delay

Type	End To End Delay (seconds)
Without Attack	6.347991
With Attack	7.593173

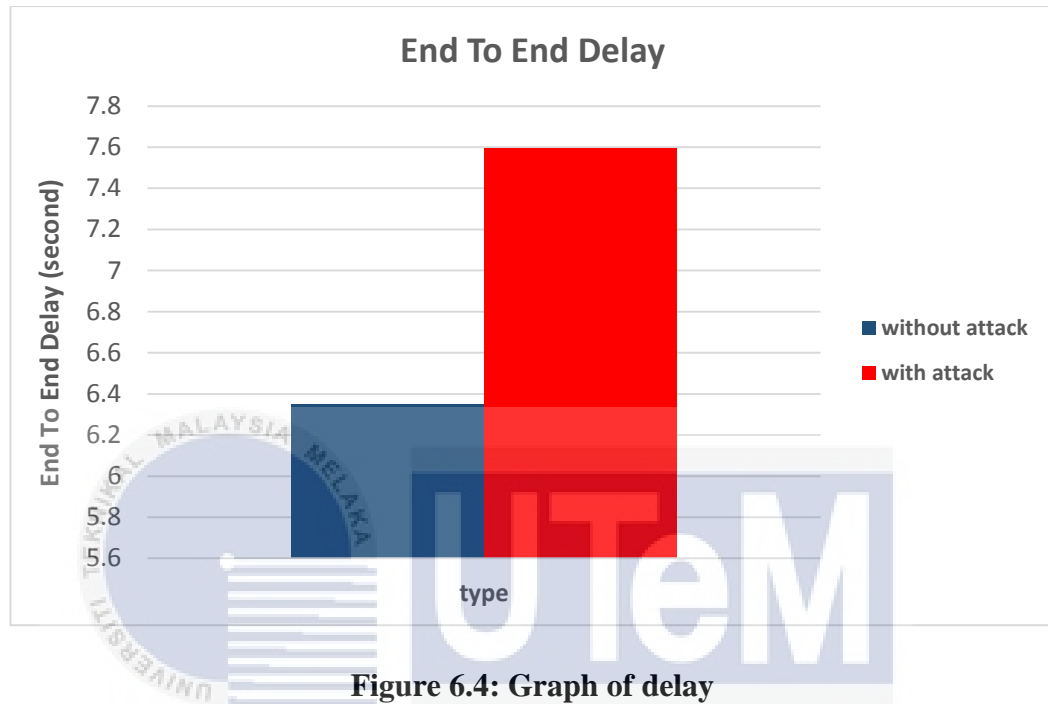


Figure 6.4: Graph of delay

The effect on delay of network with and without wormhole attack are evaluated from the trace files and graph plotted using Microsoft Excel 2013 clearly shows that the delay increased to approximately 7.593173 seconds for one packet transmission from sender to the receiver. While for the delay without wormhole attack is approximately 6.347991 seconds. The reason for this situation is because the wormhole nodes drop the packet in the network and these kind of malicious activities degrade the network routing services and the packet processing become slow.

6.2.3 Metric measurement 3: Normalised Routing Load

Normalized Routing Load is defined as the ratio of number routing packets to the number of data packets delivered to the destination. Low normalized routing overhead leads to the better performance of protocol

$$\frac{\sum \text{No. of Routing packets}}{\sum \text{No. of data packets}}$$

```
##### Check if it is a data packet
if (( $1 == "r" ) && ( $7 == "cbr" || $7 == "tcp" ) && ( $4=="AGT" )) recvd++;

##### Check if it is a routing packet
if (($1 == "s" || $1 == "f") && $4 == "RTR" && ($7 == "AODV" || $7 == "message"
|| $7 == "DSR" || $7 == "OLSR")) rt_pkts++;
}

END{
printf("#####\n");
printf("\n");
printf("    Normalized Routing Load = %.3f\n", rt_pkts/recvd);
printf("\n");
printf("#####\n");
}
```

Figure 6.5: AWK Script for Normalised Routing Load

From the AWK script the specific data is collected based on event, payload type and trace level. For data packet, the specific data is collected which are receiver event, cbr traffic and AGT level. While routing packet, the specific data collected which are sender event, RTR level, AODV payload type.

Table 6.3: Data collected for Normalised Routing Load

Type	Normalised Routing Load (bits per second)
Without Attack	0.006
With Attack	0.008

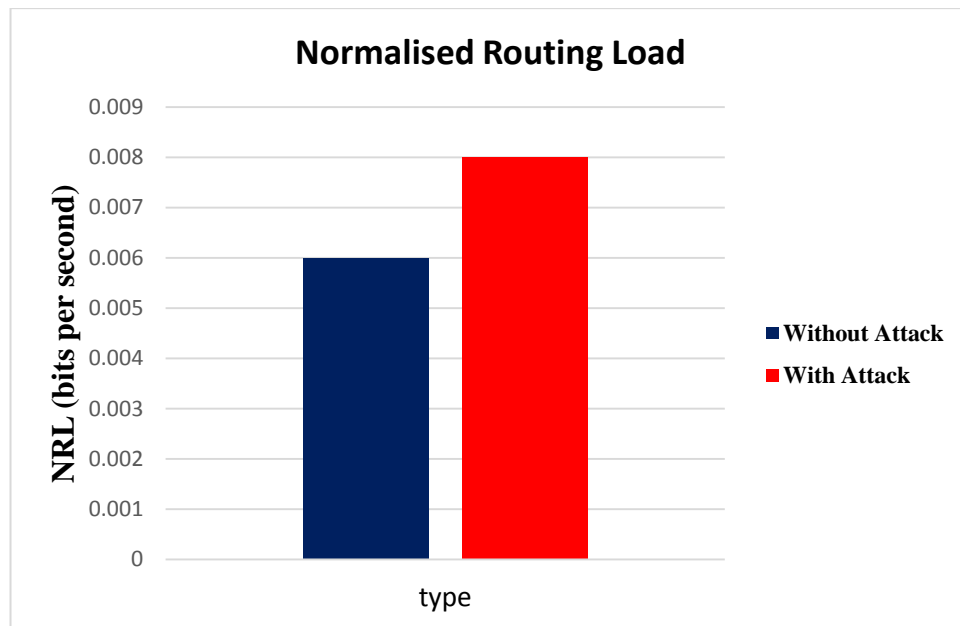


Figure 6.6: Graph of Normalised Routing Load

The effect on Normalised Routing Load of network with present of wormhole attack is increase to approximately 0.008 as shown in Figure 6.6 while without wormhole attack the NRL is approximately 0.006. Clearly, the NRL for network with wormhole attack is increase. With existing of wormhole attack the number of received packets decreases which in turn decreases the control packet.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

6.3: Conclusion

In this testing and analysis phase, performance of ad hoc routing which is AODV under present and absent wormhole attack is compared. The performance parameters taken into throughput, delay and normalised routing load (NRL). The results shown that wormhole attack is active attack that can totally disrupt routing protocol.

CHAPTER VII

CONCLUSION

7.1 Introduction

This chapter will conclude this project with project summarization, project contribution, project limitation and improvement that can be done in future works.

اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

7.2 Project Summarization

As been stated in chapter 1, this project is done based on these objectives which are to study attack on Ad hoc network, to observe the wormhole attack behaviour on Ad hoc network and to discuss the performance of Ad hoc network by comparing the results without and with wormhole attack.

First objective have been achieved by studying and understand about Ad hoc network and its security.

Second objective have been achieved by implementing wormhole attack in Ad hoc network using NS2.35 simulator.

After implementing wormhole attack on Ad hoc network, the third objective have been achieved by generating graph without and with wormhole attack on Ad hoc network based on metric measured which are throughput, end to end delay and normalised routing load.

7.3 Project Contribution

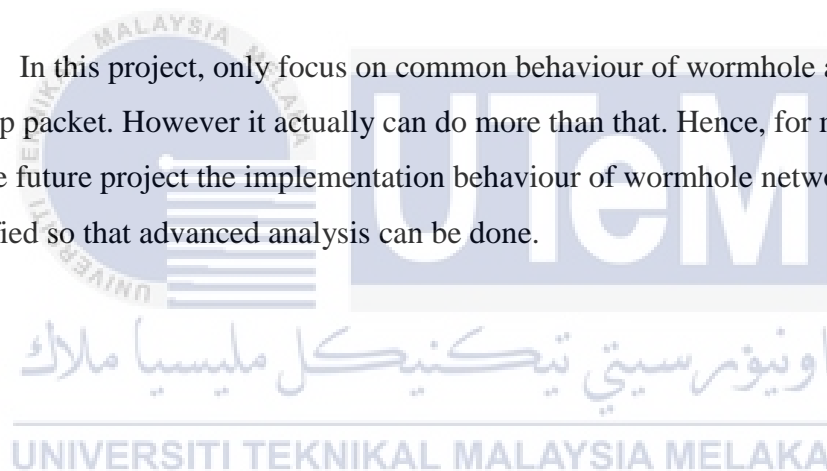
Wormhole attack can be considered as the most complicated attack in Ad hoc network. This project contributes the depth study about effect of wormhole attack in Ad hoc network. Analysing about performance on Ad hoc without and with existing wormhole attack also have been carried out so that better prevention mechanism can be implement against this attack. This project give information to other researcher to study about wormhole attack and ad hoc network.

7.4 Project Limitation

This project only create one node that act as wormhole attack. This project did not specify the behaviour of wormhole attack that create virtual “tunnel” or illusion of shortest path for packet travel from source to destination. Hence, it only act common behaviour as malicious attack that drop the packet.

7.5 Future Work

In this project, only focus on common behaviour of wormhole attack that able to drop packet. However it actually can do more than that. Hence, for more advance for the future project the implementation behaviour of wormhole network need to be specified so that advanced analysis can be done.



7.6 Conclusion

From this project it can be conclude that this project successfully met the three of objectives. This project able to give some information to other researcher to study about the wormhole attack and Ad hoc network.

REFERENCES

- (Aur et al Otmani & Ezzati, 2014; Safwani, Hassan, & Kadhum, 2011 Shrivastava & Dubey, 2015; Upadhyay & Chaurasia, 2011; Win et al., 2008)
- Arfaat, P. G. (2011). The Impact of Wormhole Attack on the Performance of Wireless Ad-Hoc Networks, *8491*, 421–425.
- Bakiler, H., & Şafak, A. (2015). Applied Mathematics , Electronics and Computers Analysis of Current Routing Attacks in Mobile Ad Hoc Networks.
- Goldsmith, A. (2004). WIRELESS COMMUNICATIONS.
- Gupta, A., Priyanka, V. J., & Upadhyay, S. (2012). Analysis of Wormhole Attack in AODV based MANET Using OPNET Simulator, *C(2)*, 63–67.
- Kaur, G., Sandeep, E., & Dhanda, K. (2013). Analysing the effect of Wormhole Attack on Routing Protocol in Wireless Sensor Network, *2(8)*.
- Lu, Y., & Lu, Y. (2003). Packet Loss in Mobile Ad Hoc Networks, 3–9.
- Moudni, H., Er-rouidi, M., Mouncif, H., & Hadadi, B. El. (2016). Attacks against AODV Routing Protocol in Mobile Ad-Hoc Networks. <http://doi.org/10.1109/CGiV.2016.81>
- Otmani, M., & Ezzati, A. (2014). Effects Of Wormhole Attack On AODV And DSR Routing Protocol Through The Using NS2 Simulator, *16(2)*, 101–107.
- Safwani, N. M. A. Al, Hassan, S., & Kadhum, M. M. (2011). MOBILE AD HOC NETWORKS UNDER WORMHOLE ATTACK : A SIMULATION STUDY, (091), 8–9.
- Shrivastava, A., & Dubey, R. (2015). Wormhole Attack in Mobile Ad-hoc Network : A Survey, *9(7)*, 293–298.
- Upadhyay, S., & Chaurasia, B. K. (2011). Impact of Wormhole Attacks on MANETs, *2(1)*, 77–82.
- Win, K. S., Physics, E., & Gyi, P. (2008). Analysis of Detecting Wormhole Attack in Wireless Networks, *2(12)*, 2704–2710.

APPENDICES

Awk Script

End To End Delay.awk

```
BEGIN{  
  
    startTime = 0;  
  
    count = 0;  
  
    }  
  
    {  
  
    event = $1;  
    time = $2;  
    node_id = $3;  
    layer = $4;  
    flags = $5;  
  
    seqno = $6;  
  
    type = $7;  
  
    size = $8;  
  
    a = $9;  
  
    b = $10;  
  
    c = $11;  
  
    d = $12;  
  
    energy = $14;
```



UNIVERSITI TEKNIKAL MALAYSIA MELAKA


```

if(layer == "AGT" && event == "s" && seqno<$6){

seqno = $6;

}

if(layer == "AGT" && event == "s" && type == "cbr"){

start_Time[$6] = $2;

}

else if((type == "cbr") && (event == "r") && (layer == "AGT")){

end_time[$6]= $2;

}

else if(event == "D" && type == "cbr" && layer == "IFQ"){

end_time[$6]= -1;

}

}

END{
for(i=0; i<=seqno; i++){
if(end_time[i] > 0){

delay[i] = end_time[i]-start_Time[i];

count++;

}

else

{

delay[i] = -1;

}

}
}

```

```

for(i=0; i<=seqno; i++){
if(delay[i] > 0){
n_to_n_delay = n_to_n_delay + delay[i];
}
}

n_to_n_delay = n_to_n_delay/count;

startTime = time;

printf("%f\t%f\n",startTime,n_to_n_delay);

}

```

Throughput.awk

```

BEGIN {

```

```

    recvdSize = 0;

```

```

    startTime = 0;

```

```

    stopTime = 0;

```

```

    sent=0;

```

```

    receive=0;

```

```

    count0 = 0;

```

```

}

```

```

{

```

```

    event = $1;

```

```

    time = $2;

```

```

    node_id = $3;

```



```

layer = $4;

flags = $5;

seqno = $6;

type = $7;

pkt_size = $8;

a = $9;

b = $10;

c = $11;

d = $12;

energy = $14;

for(seqno = 0; seqno < 2; seqno++) {
if (event == "r" && layer == "AGT") {
    count0++
    if (!startTime || (time < startTime)) {
        startTime = time;
    }

    if (time > stopTime) {
        stopTime = time;
    }

    recvdSize += pkt_size;

    throughput= recvdSize/(stopTime-startTime*0.008);
}}

```

```

startTime = time;

printf("%f\t%2f\n",startTime,throughput);

}

END {}

```

Normalised Routing Load.awk

```

BEGIN{

recvd = 0;##### to calculate total number of data packets received

rt_pkts = 0;##### to calculate total number of routing packets
received

}

{

for(seqno = 0; seqno < 2; seqno++) {

##### Check if it is a data packet

if (( $1 == "r" ) && ( $7 == "cbr" || $7 == "tcp" ) && ($4=="AGT")) recvd++;

##### Check if it is a routing packet

if (($1 == "s" || $1 == "f") && ( $4=="RTR" ) && ($7 == "AODV")) rt_pkts++;

if (!startTime || (time < startTime)) {

    startTime = $2;}

NRL=rt_pkts/recvd;

startTime = $2;

printf("%f\t%2f\n",startTime,NRL);

}}END{}

```

Tcl Script

```

# A 20-nodes for ad-hoc simulation with AODV

# Define options

set val(chan)      Channel/WirelessChannel  ;# channel type
set val(prop)      Propagation/TwoRayGround ;# radio-propagation model
set val(netif)     Phy/WirelessPhy         ;# network interface type

set val(mac)       Mac/802_11              ;# MAC type
set val(ifq)       Queue/DropTail/PriQueue ;# interface queue type
set val(ll)        LL                       ;# link layer type
set val(ant)       Antenna/OmniAntenna     ;# antenna model
set val(ifqlen)    150                      ;# max packet in ifq
set val(nn)        20                       ;# number of mobilenodes
set val(rp)        AODV                     ;# routing protocol
set val(x)         1000                     ;# X dimension of topography
set val(y)         1000                     ;# Y dimension of topography
#set val(cstop)    90                       ;# time of simulation end
set val(stop)      150                      ;# time of simulation end
set val(connect)   1;

#set val(wp1) 15;                          #wormhole node 1
#set val(wp2) 9;                           #wormhole node 2

# initialize Global Variables
set ns_ [new Simulator]
#$ns_ use-newtrace
set tracefd [open attack.tr w]
$ns_ trace-all $tracefd

set namtrace [open attack.nam w]
$ns_ namtrace-all-wireless $namtrace $val(x) $val(y)

```

```

# set up topography object
set topo [new Topography]
$topo load_flatgrid $val(x) $val(y)

#create Good
create-god $val(nn)

#create channel #1 and #2
set chan_1_ [new $val(chan)]
set chan_2_ [new $val(chan)]

# configure the nodes
$ns_ node-config -adhocRouting $val(rp) \
                 -llType $val(ll) \
                 -macType $val(mac) \
                 -ifqType $val(ifq) \
                 -ifqLen $val(ifqlen) \
                 -antType $val(ant) \
                 -propType $val(prop) \
                 -phyType $val(netif) \
                 -topoInstance $topo \
                 -agentTrace ON \
                 -routerTrace ON \
                 -macTrace ON \
                 -movementTrace ON \
                 -channel $chan_1_

#creating mobile AODV nodes for simulation
puts "creating nodes....."
for {set i 0} {$i < $val(nn)} {incr i} {
    set node_($i) [$ns_ node]
    $node_($i) random-motion 0; #disable random motion

```

```
}
```

```
#=====
```

```
# Provide initial location of mobile nodes
```

```
#=====
```

```
$node_(0) set X_ 110.666
```

```
$node_(0) set Y_ 385.43
```

```
$node_(0) set Z_ 0.0
```

```
$node_(1) set X_ 526.075
```

```
$node_(1) set Y_ 742.198
```

```
$node_(1) set Z_ 0.0
```

```
$node_(2) set X_ 286.405
```

```
$node_(2) set Y_ 680.553
```

```
$node_(2) set Z_ 0.0
```

```
$node_(3) set X_ 372.892
```

```
$node_(3) set Y_ 455.324
```

```
$node_(3) set Z_ 0.0
```

```
$node_(4) set X_ 503.224
```

```
$node_(4) set Y_ 331.549
```

```
$node_(4) set Z_ 0.0
```

```
$node_(5) set X_ 652.712
```

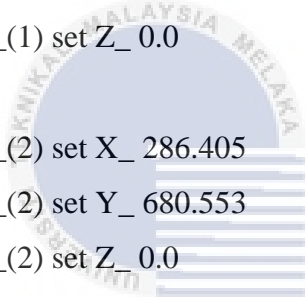
```
$node_(5) set Y_ 422.823
```

```
$node_(5) set Z_ 0.0
```

```
$node_(6) set X_ 493.261
```

```
$node_(6) set Y_ 601.608
```

```
$node_(6) set Z_ 0.0
```



اونيورسيتي تيكنيكل ماليسيا ملاك
UNIVERSITI TEKNIKAL MALAYSIA MELAKA

\$node_(7) set X_ 265.405

\$node_(7) set Y_ 714.219

\$node_(7) set Z_ 0.0

\$node_(8) set X_ 198.061

\$node_(8) set Y_ 624.927

\$node_(8) set Z_ 0.0

\$node_(9) set X_ 611.437

\$node_(9) set Y_ 552.154

\$node_(9) set Z_ 0.0

\$node_(10) set X_ 62.9053

\$node_(10) set Y_ 519.543

\$node_(10) set Z_ 0.0

\$node_(11) set X_ 373.502

\$node_(11) set Y_ 575.66

\$node_(11) set Z_ 0.0

\$node_(12) set X_ 210.938

\$node_(12) set Y_ 269.301

\$node_(12) set Z_ 0.0

\$node_(13) set X_ 241.325

\$node_(13) set Y_ 525.078

\$node_(13) set Z_ 0.0

\$node_(14) set X_ 106.508

\$node_(14) set Y_ 660.878

\$node_(14) set Z_ 0.0



اونيورسيتي تيكنيكل مليسيا ملاك
UNIVERSITI TEKNIKAL MALAYSIA MELAKA

\$node_(15) set X_ 254.379

\$node_(15) set Y_ 367.516

\$node_(15) set Z_ 0.0

\$node_(16) set X_ 641.404

\$node_(16) set Y_ 680.265

\$node_(16) set Z_ 0.0

\$node_(17) set X_ 373.267

\$node_(17) set Y_ 272.287

\$node_(17) set Z_ 0.0

\$node_(18) set X_ 745.384

\$node_(18) set Y_ 568.295

\$node_(18) set Z_ 0.0

\$node_(19) set X_ 508.734

\$node_(19) set Y_ 467.827

\$node_(19) set Z_ 0.0

#=====

set a UDP connection between nodes

#=====

set udp [new Agent/UDP]

\$ns_ attach-agent \$node_(0) \$udp

set null [new Agent/Null]

\$ns_ attach-agent \$node_(18) \$null

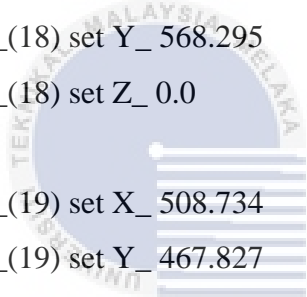
set cbr [new Application/Traffic/CBR]

\$cbr set packetSize_ 512

\$cbr set rate_ 100Kb

\$cbr set interval_ 0.005

#\$cbr set maxpkts_ 10000



اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

```

$nbr attach-agent $udp
$ns_ connect $udp $null
$ns_ at 30.0 "$nbr start"
$ns_ at 150.0 "$nbr stop"
$ns_ at 0.1 "$node_(0) label \"Send1\""
$ns_ at 0.1 "$node_(18) label \"Receive1\""
#=====
# CREATING WORMHOLE NODES FOR SIMULATION
#=====
$ns_ at 70.0 "[$node_(15) set ragent_] wormhole"
$ns_ at 70.0 "[$node_(9) set ragent_] wormhole"
[$node_(15) set ll_(0)] wormhole-peer [$node_(9) set ll_(0)]
[$node_(9) set ll_(0)] wormhole-peer [$node_(15) set ll_(0)]
#$ns_ duplex-link $node_(15) $node_(9) 512Mb 155ms DropTail
$ns_ at 0.1 "$node_(15) label \"wormhole\""
$ns_ at 0.1 "$node_(9) label \"wormhole\""
#=====
# End wormhole
#=====
for {set i 0} {$i < $val(nn)} {incr i} {
    $ns_ initial_node_pos $node_( $i) 40
}

# ending nam and the simulation
$ns_ at $val(stop) "$ns_ nam-end-wireless $val(stop)"
$ns_ at $val(stop) "stop"
$ns_ at 150.0 "puts \"end simulation\" ; $ns_ halt"
proc stop {} {
    global ns_ tracefd namtrace
    $ns_ flush-trace
    close $tracefd
    close $namtrace
exec nam attack.nam &

```

```
exit 0
}
```

```
$ns_run
```

ll.cc

```
#ifndef lint
```

```
static const char rcsid[] =
```

```
"@(#) $Header: /cvsroot/nsnam/ns-2/mac/ll.cc,v 1.47 2010/03/08 05:54:51
tom_henderson Exp $ (UCB)";
```

```
#endif
```

```
#include <errmodel.h>
```

```
#include <mac.h>
```

```
#include <ll.h>
```

```
#include <address.h>
```

```
#include <dsr/hdr_sr.h>
```

```
#define TRACE_DROP 0
```

```
int hdr_ll::offset_;
```

```
static class LLHeaderClass : public PacketHeaderClass {
```

```
public:
```

```
    LLHeaderClass() : PacketHeaderClass("PacketHeader/LL",
```

```
        sizeof(hdr_ll)) {
```

```
        bind_offset(&hdr_ll::offset_);
```

```
    }
```

```
} class_hdr_ll;
```

```

static class LLClass : public TclClass {

public:

    LLClass() : TclClass("LL") {}

    TclObject* create(int, const char*const*) {

        return (new LL);

    }

} class_ll;

LL::LL() : LinkDelay(), seqno_(0), ackno_(0), macDA_(0), ifq_(0),

    mac_(0), lanrouter_(0), arptable_(0), varp_(0),

    downtarget_(0), uptarget_(0), drop_on_send(0), drop_on_recv(0),

routing_packet_count(0), routing_byte_count(0), data_packet_count(0),

data_byte_count(0)

{

    bind("macDA_", &macDA_);

    wormhole_head.ll = NULL;

    wormhole_head.id = -1;

    wormhole_head.next = NULL;

}

int LL::command(int argc, const char*const* argv)

{

Tcl& tcl = Tcl::instance();

    if (argc == 3) {

        if (strcmp(argv[1], "ifq") == 0) {

            ifq_ = (Queue*) TclObject::lookup(argv[2]);

            return (TCL_OK);

        }

    }

}

```

```

}

if(strcmp(argv[1], "arptable") == 0) {

arptable_ = (ARPTable*)TclObject::lookup(argv[2]);

assert(arptable_);

return TCL_OK;

}

if(strcmp(argv[1], "varp") == 0) {

varp_ = (VARPTable*)TclObject::lookup(argv[2]);

assert(varp_);

return TCL_OK;

}

if (strcmp(argv[1], "mac") == 0) {

mac_ = (Mac*) TclObject::lookup(argv[2]);

assert(mac_);

return (TCL_OK);

}

if (strcmp(argv[1], "down-target") == 0) {

downtarget_ = (NsObject*) TclObject::lookup(argv[2]);

return (TCL_OK);

}

if (strcmp(argv[1], "up-target") == 0) {

uptarget_ = (NsObject*) TclObject::lookup(argv[2]);

return (TCL_OK);

}

```

```

if (strcmp(argv[1], "lanrouter") == 0) {

    lanrouter_ = (LanRouter*) TclObject::lookup(argv[2]);

    return (TCL_OK);

}

else if( strcmp( argv[1], "wormhole-peer" ) == 0 ) {

    wormhole_peer*    wp    =    (wormhole_peer*)malloc(    sizeof(
wormhole_peer ) );

    if( !wp ) {

fprintf( stderr, "(%03d) - LL::command - error allocating memory for new wormhole
peer!" );

        exit(-1);

    }

    // init fields
    wp->ll = (LL *) TclObject::lookup( argv[2] );
    wp->id = wp->ll->mac_->addr();

// insert at head of list

    wp->next = wormhole_head.next;

    wormhole_head.next = wp;

    printf( "(%03d) - LL::command - added %d to wormhole peer
list\n", mac_->addr(), wp->id );

    return TCL_OK;

}

}

```

```

else if (argc == 2) {

    if (strcmp(argv[1], "ifq") == 0) {

        tcl.resultf("%s", ifq_>name());

        return (TCL_OK);

    }

    if (strcmp(argv[1], "mac") == 0) {

        tcl.resultf("%s", mac_>name());

        return (TCL_OK);

    }

    if (strcmp(argv[1], "down-target") == 0) {

        tcl.resultf("%s", downtarget_>name());

        return (TCL_OK);

    }

    if (strcmp(argv[1], "up-target") == 0) {

        tcl.resultf("%s", uptarget_>name());

        return (TCL_OK);

    }

    if (strcmp(argv[1], "drop-on-send") == 0) {

        drop_on_send = 1;

        return TCL_OK;

    }

    if (strcmp(argv[1], "drop-on-recv") == 0) {

        drop_on_recv = 1;

```

```

        return TCL_OK;
    }

    if (strcmp(argv[1], "routing_packet_count") == 0) {
        tcl.resultf("%d", routing_packet_count );
        return (TCL_OK);
    }

    if (strcmp(argv[1], "routing_byte_count") == 0) {
        tcl.resultf("%d", routing_byte_count );
        return (TCL_OK);
    }

    if (strcmp(argv[1], "data_packet_count") == 0) {
        tcl.resultf("%d", data_packet_count );
        return (TCL_OK);
    }

    if (strcmp(argv[1], "data_byte_count") == 0) {
        tcl.resultf("%d", data_byte_count);
        return (TCL_OK);
    }
}

return LinkDelay::command(argc, argv);
}

void LL::recv(Packet* p, Handler* /*h*/)
{

```



```

hdr_cmn *ch = HDR_CMN(p);

//char *mh = (char*) HDR_MAC(p);

//struct hdr_sr *hsr = HDR_SR(p);

/*

* Sanity Check

*/

assert(initialized());

//if(p->incoming) {

//p->incoming = 0;

//}

// XXXXX NOTE: use of incoming flag has been deprecated; In order to track
direction of pkt flow, direction_ in hdr_cmn is used instead. see packet.h for details.

// If direction = UP, then pass it up the stack
// Otherwise, set direction to DOWN and pass it down the stack

if(ch->direction() == hdr_cmn::UP) {

    //if(mac_->hdr_type(mh) == ETHERTYPE_ARP)

    if(ch->ptype_ == PT_ARP) {

//printf( "%010.6f - (%03d) - LL::recv - got ARP packet\n",
Scheduler::instance().clock(), mac_->addr() );

        arptable_->arpinput(p, this);

    }

else{

        switch( ch->ptype() ) {

```

```

//case PT_AODVUU:

case PT_MAC:

case PT_DSR:

case PT_ARP:

case PT_CBR:

// let routing and control packets through

break;

case PT_AODV:

if( drop_on_recv ) {

static int corrupt_count = 1;

if( TRACE_DROP ) printf( "%010.6f -
(%03d)- LL::recv - dropping data packet (%d)\n", Scheduler::instance().clock(),
mac_>addr(), corrupt_count++ );

free( p );

return;

}

break;

default:

fprintf( stderr, "%010.6f - (%03d) - LL::recv -
ERROR - unknown packet type (%d = %s)!\n", NOW, mac_>addr(), ch->ptype(),
packet_info.name( ch->ptype() ) );

exit( -1 );

}

uptarget_ ? sendUp(p) : drop(p);

}

```

```

        return;
    }

    ch->direction() = HDR_CMN::DOWN;

    sendDown(p);
}

void LL::sendDown(Packet* p)
{
    HDR_CMN *ch = HDR_CMN(p);

    HDR_IP *ih = HDR_IP(p);

    int is_broadcast = 0;
    int unicast_addr = -1;

    int is_routing = 0;
    switch( ch->ptype() ) {
        //case PT_AODVUU:

        case PT_DSR:

        case PT_ARP:

        case PT_MAC:

        //case PT_SEC_RT:

        case PT_CBR:

        // let routing and control packets through

```

```

is_routing = 1;

break;

case PT_AODV:

if( drop_on_send ) {

        static int corrupt_count = 1;

if( TRACE_DROP ) printf( "%010.6f - (%03d) - LL::sendDown - dropping data
packet (%d)\n", Scheduler::instance().clock(), mac_->addr(), corrupt_count++ );

        free( p );

        return;

}

break;

default:

        fprintf( stderr, "%010.6f - (%03d) - LL::sendDown - ERROR -
unknown packet type (%d = %s)!\n", NOW, mac_->addr(), ch->ptype(),
packet_info.name( ch->ptype() ) );

        exit( -1 );

}

nsaddr_t dst = (nsaddr_t)Address::instance().get_nodeaddr(ih->daddr());

//nsaddr_t dst = ih->dst();

hdr_ll *llh = HDR_LL(p);

char *mh = (char*)p->access(hdr_mac::offset_);

llh->seqno_ = ++seqno_;

```

```

llh->lltype() = LL_DATA;

mac_->hdr_src(mh, mac_->addr());

mac_->hdr_type(mh, ETHERTYPE_IP);

int tx = 0;

switch(ch->addr_type()) {

case NS_AF_ILINK:

    mac_->hdr_dst((char*) HDR_MAC(p), ch->next_hop());

    // check next hop for wormhole peer / broadcast

    if( ch->next_hop() == MAC_BROADCAST ) {

        //printf( "%010.6f - (%03d) - LL::sendDown - sending a
broadcast - ILINK\n", NOW, mac_->addr() );
        is_broadcast = 1;
    }
    else {

        //printf("%010.6f - (%03d) - LL::sendDown - sending a unicast
- ILINK \n", NOW, mac_->addr() );

        unicast_addr = ch->next_hop();

    }

    //else {

        //    printf( "%010.6f - (%03d) - LL::sendDown - sending a
unicast - ILINK\n", NOW, mac_->addr() );

    //}

    break;

case NS_AF_INET:

```

```

dst = ch->next_hop();

/* FALL THROUGH */

case NS_AF_NONE:

if (IP_BROADCAST == (u_int32_t) dst)

{

    mac_->hdr_dst((char*)          HDR_MAC(p),
MAC_BROADCAST);

    //printf( "%010.6f - (%03d) - LL::sendDown - sending a
broadcast - NONE\n", NOW, mac_->addr() );

    is_broadcast = 1;

    break;
}

else {

//printf( "%010.6f - (%03d) - LL::sendDown - sending a unicast
(wormhole) - NONE\n", NOW, mac_->addr() );
unicast_addr = dst;

}

//else {

//    printf( "%010.6f - (%03d) - LL::sendDown - sending a unicast
- NONE\n", NOW, mac_->addr() );

//}

/* Assuming arptable is present, send query */

if (arptable_) {

    tx = arptable_->arpresolve(dst, p, this);

    break;
}

```

```

    }

    //if (varp_) {

    //tx = varp_->arpresolve(dst, p);

    //break;

    //}

    /* FALL THROUGH */

    default:

    int IPnh = (lanrouter_) ? lanrouter_->next_hop(p) : -1;

    if (IPnh < 0)

        mac_->hdr_dst((char*) HDR_MAC(p), macDA_);

    else if (varp_)

        tx = varp_->arpresolve(IPnh, p);

    else

        mac_->hdr_dst((char*) HDR_MAC(p), IPnh);

        break;
}

```

```

if (tx == 0) {

```

```

    Scheduler& s = Scheduler::instance();

```

```

    // wormhole decision point (decide if this packet is going through the
    wormhole or not)

```

```

    if( wormhole_head.next ) {

```

```

        if( is_broadcast ) {

```

```
//printf( "%010.6f - (%03d) - LL::sendDown - broadcast split\n", NOW, mac_->addr()
);
```

```
// send a copy to each wormhole peer
```

```
wormhole_peer *wp = &wormhole_head;
```

```
while( wp->next ) {
```

```
    wp = wp->next;
```

```
    Packet *p_copy = p->copy();
```

```
    hdr_cmn::access(p_copy)->direction() = hdr_cmn::UP;
```

```
    s.schedule( wp->ll, p_copy, delay_ );
```

```
}
```

```
// AND send it out our "real" interface (and gather stats)
```

```
if( is_routing ) {
```

```
    routing_packet_count++;
```

```
    routing_byte_count += ch->size_;
```

```
}
```

```
else {
```

```
    data_packet_count++;
```

```
    data_byte_count += ch->size_;
```

```
}
```

```
s.schedule(downtarget_, p, delay_);
```

```
return;
```

```
}
```

```
else {
```

```
// scan through the list to see if it is for a wormhole peer
```



```

wormhole_peer *wp_curr = wormhole_head.next;

wormhole_peer *wp_prev = &wormhole_head;

while( wp_curr ) {

    if( wp_curr->id == unicast_addr ) {

// if we found a match then send the packet too this wormhole peer only

        hdr_cmn::access(p)->direction() = hdr_cmn::UP;

        s.schedule( wp_curr->ll, p, delay_ );

// move this wormhole peer to the front of the list

// (optimizes many unicasts to the same peers)

        wp_prev->next = wp_curr->next;
        wp_curr->next = wormhole_head.next;
        wormhole_head.next = wp_curr;

        return;
    }

// otherwise keep looking through list

    wp_prev = wp_curr;

    wp_curr = wp_curr->next;

}

// fall through if we don't find a matching wormhole peer

}

}

```

```

//printf( "%010.6f - (%03d) - LL::sendDown - normal only\n", NOW, mac_-
>addr() );

// let mac decide when to take a new packet from the queue.

if( is_routing ) {

    routing_packet_count++;

    routing_byte_count += ch->size_;

}

else {

    data_packet_count++;

    data_byte_count += ch->size_;

}

s.schedule(downtarget_, p, delay_);
}

void LL::sendUp(Packet* p)
{
    Scheduler& s = Scheduler::instance();

    if (hdr_cmn::access(p)->error() > 0)

        drop(p);

    else

        s.schedule(uptarget_, p, delay_);

}

inline void LL::hdr_dst(Packet *, int)

{}

```

ll.h

```

#ifndef ns_ll_h
#define ns_ll_h

#include <delay.h>
#include <queue.h>
#include <arp.h>
#include <classifier.h>
#include <lanRouter.h>
#include <varp.h>

enum LLFrameType {
    LL_DATA = 0x0001,
    LL_ACK = 0x0010
};

struct hdr_ll {
    LLFrameType lltype_; // link-layer frame type
    int seqno_; // sequence number
    int ackno_; // acknowledgement number
    int bopno_; // begin of packet seqno
    int eopno_; // end of packet seqno
    int psize_; // size of packet
    double sendtime_; // time the packet is sent
    static int offset_;
    inline int& offset() { return offset_; }
    static hdr_ll* access(const Packet* p) {

```

```

        return (hdr_ll*) p->access(offset_);
    }

    inline LLFrameType& lltype() { return lltype_; }

    inline int& seqno() { return seqno_; }

    inline int& ackno() { return ackno_; }

    inline int& bopno() { return bopno_; }

    inline int& eopno() { return eopno_; }

    inline int& psize() { return psize_; }

    inline double& sendtime() { return sendtime_; }
};

// define elements for the wormhole peer list
class LL;

typedef struct wormhole_peer_struct {
    LL* ll;
    int id;

    struct wormhole_peer_struct* next;
} wormhole_peer;

class LL : public LinkDelay {
public:
    friend void ARPTable::arpinput(Packet *p, LL* ll);

    friend void ARPTable::arprequest(nsaddr_t src, nsaddr_t dst, LL* ll);

    LL();

```

```

virtual void recv(Packet* p, Handler* h);

void handle(Event* e) { recv((Packet*)e, 0); }

inline int initialized() {
    return (mac_ && uptarget_ && downtarget_);
}

virtual void sendUp(Packet* p);

virtual void sendDown(Packet* p);

inline int seqno() { return seqno_; }

inline int ackno() { return ackno_; }

inline int macDA() { return macDA_; }

virtual void hdr_dst(Packet *p, int macDA);

inline Queue *ifq() { return ifq_; }

inline NsObject* downtarget() { return downtarget_; }

inline NsObject* uptarget() { return uptarget_; }

inline ARPTable *arp_table() { return arptable_; }

protected:

int command(int argc, const char*const* argv);

int seqno_;           // link-layer sequence number

int ackno_;          // ACK received so far

int macDA_;          // destination MAC address

Queue* ifq_;         // interface queue

Mac* mac_;           // MAC object

LanRouter* lanrouter_; // for lookups of the next hop

```

```

ARPTable* arptable_;      // ARP table object

VARPTable* varp_;        // Virtual ARP object

NsObject* downtarget_;   // for outgoing packet

NsObject* uptarget_;     // for incoming packet

float drop_on_send;

float drop_on_rcv;

wormhole_peer wormhole_head;

// statistics variables for tracking routing overhead
int routing_packet_count;
int routing_byte_count;
int data_packet_count;
int data_byte_count;
};

#endif

```