An Improved Adhoc On Demand Distance Vector Protocol for Mobile Adhoc Networks

M.M.Razmy

(169344 J)

Faculty of Information Technology

University of Moratuwa

April 2019

An Improved Adhoc On Demand Distance Vector Protocol for Mobile Adhoc Networks

M.M.Razmy

(169344 J)

Dissertation submitted to the Faculty of Information Technology, University of Moratuwa, Sri Lanka for the partial fulfillment of the requirement of the Degree of Master of Science in Information Technology

April 2019

Declaration

I do hereby declare that this work has been originally carried out by me under the guidance of and supervisor of Dr.M.F.M.Firdhous Director of Post Graduate Studies, Faculty of Information Technology, University of Moratuwa and this work has not been submitted elsewhere for any other degree.

I certify that this dissertation does not incorporate without due acknowledgement any material submitted for a Master Degree or any Degree in any university and to the best of my knowledge and belief, it does not contain any material previously published or written by any other person except where due reference is the text.

M.M.Razmy Signature of Student

Date: 2019/04/10

Supervised by

Dr.M.F.M.Firdhous Signature of Supervisor

Date: 2019/04/10

Dedicated to

My loving father, late Mr. Pakeer Mohamed

Acknowledgement

First of all I want to thank Al mighty Allah to help me to complete this work. Then I express deep gratitude to my loving parents who helped me financially and making me interest to do my work.

I particularly want to thank Dr.M.F.M.Firdhous, Director of Postgraduate Studies, Faculty of Information Technology, University of Moratuwa, who helped me to decide the topic and giving the explanation about the work. Also would like to thank for the all the lecturers taught us in the Master Program who gave their full support to complete this dissertation.

Furthermore, I deeply indebted to my loving wife and a son who have helped me in several ways. Finally, I express deep gratitude to my friends and all, who despite having had to cope with my tendency to become to absorbed in my work gave me all their support.

Abstract

Mobile Adhoc Network is a kind of adhoc network it can change the locations and configure by itself. The Mobile Adhoc Network uses the wireless connections to connect to various networks like standard WiFi connection, cellular or satellite communication. The mobile Adhoc network does not use any static infrastructure due to multipath broadcasting and high flexibility of nodes. Because of the availability of free license in wireless communication, the use of MANET Application has been increased. MANETs Applications are implemented in disaster-management, business meetings, military operations and rescue operations. There are many different protocols are implemented in MANET while sending data packet source node to the destination node. These protocols can be classified as Proactive, Reactive and hybrid Protocols. Reactive is a very popular routing protocol used in wireless communication that provide the accessible solution for large network. Ad-hoc on Demand Distance Vector Protocol is a kind of Reactive routing protocol. There are many issues in MANET. Security issue is one of the main issue in MANET. With the aim of this research, researcher interesting to find to Detect and Prevent the Cooperative Black hole attack for AODV Protocol.

Previous Authors were introduced Dynamic Learning System against Black hole attack Adhoc On Demand Distance Protocol for Mobile Ad-hoc Network. However the introduced method has only support for a single Black hole attack and its routing overhead is very high. This thesis presents the improved further implemented method Detect, Prevent and Reactive of AODV which will reduce the routing overhead and increasing the packet delivery ratio of AODV Protocol. For the implementation of the research, researcher use Network Simulator 3.24, which is new simulator written from scratch. It is supported C++ and Python language. It will depend on the ongoing contributions of the community to develop new models, debug or maintain existing ones, and share results.

Keyword : MANET, AODV, Black hole, Protocol, Cooperative

Table of Contents

List of Figures

List of Tables

Abbreviations

AODV - Ad hoc On Demand Distance Vector

MDPRAODV - Method of Detecting Preventing Reactive AODV

WLAN- Wireless LAN

Wifi - Wireless Fidelity

DSDV -Distance Sequence Distance Vector Routing Protocol

FSR-Fisheye State Routing

OLSR- Optimized Link State Routing Protocol

WRP - Wireless Routing Protocol

DSR- Dynamic Source Routing

TORA-Temporary Ordered Routing Protocol

ZRP-Zone Routing Protocol

RREP - Route Reply

RREQ - Route Request

CAODV - Credit Based Ad hoc On Demand Distance Vector

DOS - Denial of Service Attack

HTTPS- Hyper Text Transfer Protocol Secure

SSL- Secure Socket Layer

WEP - Wired Equivalent Privacy

WAP - Wireless Protected Access

NS3 - Network Simulator 3

Chapter 01

Introduction

1.0 Introduction

Ad-Hoc is a Latin word, that means “for this purpose”. In a computer network environment an ad-hoc describe to a network connection established for a single period does not require any other devices or control station. An Ad-Hoc network has the ability to maintain any broken link or path on the network. Whenever a node in the network leaves the network that causes the link between other nodes is broken. The broken node in the network request other routes and new route are established. The ad - hoc network can be mainly classified into Mobile Ad-hoc network and static ad-hoc network. Mobile Ad-Hoc Network is a wireless or mobile network, information is divided into packets and each packet should have packets sequence number and destination address. These packets are sent one node to another until it reaches the destination. MANET consist of many number of mobile nodes which are free and moving in and out of the network [1].

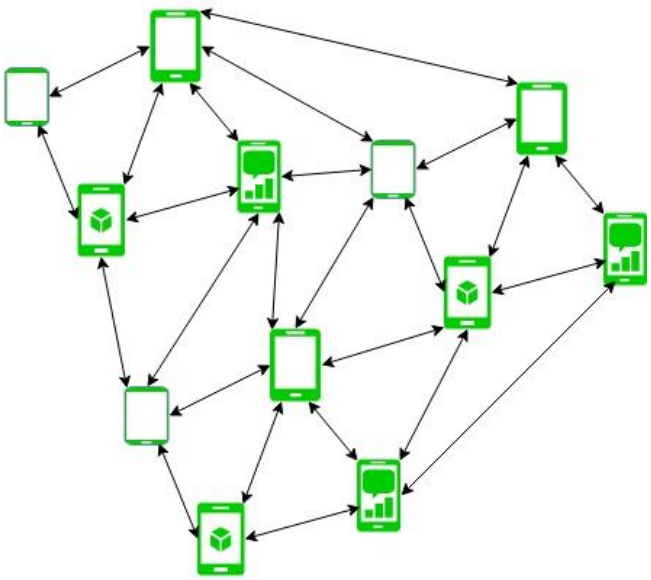


Figure 1. Mobile Adhoc Network

1.1 Background and Motivation

Security is an important Challenge of Mobile Ad-Hoc Network. MANET rarely suffers from a security attack due to the lack of trusted centralized authority and

limited resources. MANET should have a secure way for transmission and communication. In order to provide secure communication and transmission, the researchers should understand different types of attacks and their effects on the MANET. In computer networking, a packet drop attack or black hole attack is a type of denial-of-service attack in which a router that is supposed to relay packets instead discards them. There are mainly two types of Black hole attacks in Mobile Ad-hoc Network. Which are Single Black hole attack and Cooperative Black hole attacks[2]. With the aim of the research, researcher's aim to detect cooperative or Collaborative Black hole attack in Ad-hoc On Demand Distance Protocol in Mobile Ad-hoc network.

1.2 Problem definition

Based on the literature review, many numbers of researchers have detected the single Black hole attack AODV protocol in Mobile Ad-hoc network and also they detect the limited number of malicious nodes in the MANET and unable to Detect the Cooperative Black hole attack and improve the efficiency of the Protocol.

1.3 Objectives

There are four types of objectives have been identified for this research which is related to the Objective of the problem and solution.

1.3.1 Objective related to the Problem

1. To Detect the Cooperative Black hole nodes
2. To Increase the Packet Delivery Ratio of Existing AODV protocol
3. To Decrease the Packet Lost Ratio of Original AODV Protocol
4. To Decrease the Packet Delivery of AODV Protocol

1.3.2 Objectives related to the solution

1. To Modified the Original AODV code as MDPRAODV and detecting the Cooperative Black hole node.
2. To Compare the results of Packet Delivery Ratio output of Pure AODV and MDPRAODV.
3. To compare the results of Packet Loss Ratio output of Pure AODV and MDPRAODV
4. To Analyze the output results and displayed it on Gnuplot graph

5. Preparation of documentation

1.4 Resource required

Resources required for the design and implementation are a personal computer with Ubuntu operating System and Network Simulator 3.24 Version.

1.5 Chapter Organization of the Dissertation

The rest of the dissertation is structured as follows. Chapter 02 presents the previous researcher's work related to the Review of Routing Protocol, Performance of Routing Protocol, Security Attack and Prevent to detect a black hole attack in the AODV Protocol in MANET. Chapter 03 discusses the Research Methodology for Detecting the Cooperative Black hole attack in AODV protocol in MANET. Chapter 04 discusses the Overview MANET and Routing Protocol in MANET. Chapter 05 discusses Security attacks of Mobile Ad-hoc network. Chapter 06 addresses the Software Requirement & implementation of the Project. Chapter 07 addresses the Research Results and Evaluation and Chapter 08 Describe the Conclusion and Future work of the dissertation.

1.6 Summary

This chapter gave the general overview of the dissertation. Background and motivation of the research have been addressed primarily also Problem definition, Objective and Resource are required to perform the research expressed. Next chapter will describe the previous researcher's work related to the Review of Routing Protocol, Performance of Routing Protocol, Security Attack and Prevent to detect a black hole attack in the AODV Protocol in Mobile Ad-hoc Network attacks which were implemented by others.

Chapter 02

Literature Review

2.0 Introduction

Chapter 01 has stated the Background, Motivation, Problem definition and Objective of the Thesis. Chapter 02 presents the previous researcher's work related to the Review of Routing Protocol, Performance of Routing Protocol, Security Attack and how to detect and prevent detect a black hole attack in the AODV Protocol in MANET. At the end of the Chapter 02, the Summary of the critical review is expressed.

2.1 Literature Review

Hinds et al. [3] were presented the review of routing protocols for mobile ad hoc network. The researchers described three categories of routing protocols, namely Reactive routing protocols, Proactive routing protocols and hybrid routing protocols. Reactive Routing protocol consumes less bandwidth and it consists of two main functions known as Route Discovery and Route maintenance. Route Discover will discover new routes and Route Maintenance will identify the link failure and repair of existing routes. Ad hoc on demand distance vector protocol, Dynamic source routing protocol, and Temporary ordered routing algorithm protocol are the example of Reactive Routing Protocols. In Proactive routing, every node or host in the network maintains one or more table which contains details of the entire network. If there any changes occurred in the network, these tables will be updated. Destination sequenced distance vector routing protocol, Wireless routing protocol, Optimized link state routing protocol and Fisheye state routing protocol are examples of Proactive Routing Protocol. A hybrid routing protocol is a third protocol type which uses a combination of Distance Vector Routing Protocol and Link state routing Protocol. It uses distance vector for further parameters to decide the shortest path to the destination and report the message when there is update in the network. Zone routing protocol is the example of the Hybrid routing protocol.

Gupta et al. [4] Identified the Performance analysis of the routing protocol in Mobile Ad-hoc network and evaluate the performance of routing protocols in

Mobile ad-hoc network environment. Further, researchers were interested to evaluate the ad-hoc on Demand distance vector protocol, Dynamic source routing protocol and Temporary ordered routing protocol. For the simulation environment they used network simulator version 2 and they were interested throughput and end-to-end delay as performance metrics. The throughput calculated the total number of data packets transferred effectively from one place to another in a given period. On the other hand End-to-end delay calculated the time for a certain data packet to be transferred across the network from one place to another. The researchers were discovered AODV protocol has higher throughput than DSR, TORA protocols and AODV protocol's end-to-end delay was higher than DSR protocol. However, end-to-end delay of TORA protocol was worse than other two mentioned protocols.

Deng et al.[5] were identified the Routing Security in Ad Hoc Networks. The researchers were stated the security issues of Mobile Ad-hoc network and to prevent the black hole attack for AODV protocol in MANET. Further, researchers were described several attacks in MANET environment, namely Black Hole attack, Denial Of Service attack, Routing Table Overflow, Impersonation, Energy consummation and Information disclosure. The black hole attack is a kind of denial of service attack where malicious nodes announce the invalid path as a good path to the source node when the route discovery process. When the intruder node sending a false message to the network does not consider about the routing table. In another case intruder node in the network consumes all the data packets. Avoid the black hole attack, researchers were proposed to deactivate the capability of reply message of an intermediate node.

Manickam et al.[6] have presented the Performance Comparisons of Routing Protocols in Mobile Ad-hoc networks. The researchers were stated the Performance comparison of Reactive and Proactive routing Protocols in Ad-hoc network. Above research, researchers were described classification of routing protocols and analyze their performance using Network simulator version 2. The researchers were compared the simulation results of Packet Delivery Ratio, throughput, End-to-end delay of AODV, DSDV and DSR protocols. The DSDV Routing protocol is suitable when the nodes of the network are limited with low

movement also DSDV deliver lower end-to-end delay. The DSR routing protocol is desirable for limited traffic with limited mobility. On the other hand AODV protocol has a high packet delivery ratio. However, comparing with DSR protocol AODV protocol has higher throughput and end-to-end-delay.

Upadhyay et al.[7] were presented the Different Types of Attacks on Integrated Ad-hoc network. Further, they were stated many attacks, namely wormhole attack, Black hole attack, Syn flooding, snooping, replay attacks, jamming and Denial of service attack. In a wormhole attack, an intruder or wicked node gets the packets at one location in the network and bridges them to another location in the network and rebroadcast them there into the network. In a black hole attack that mischievous node consumes the routing path and stated it has the valid path to any desired in the network does not forward the packets to its neighbors. Syn flood attack or half-open attack is a kind of denial-of-service attack which main target is to avoid reaching the server by using all the server resources. In snooping attack, an attacker observing data traffic sending between the source node and destination node on the network. In replay attack an actual data transmission between source and destination is repeated or delayed. Jamming attack is a kind of denial of service attack in which wicked node blocks the allowed packet, which are transmissions from source to destination node in the network. In a Denial of service attack attacker attempt to avoid authorized users to consuming the service and also attacker typically sends unnecessary data packets to a server that source request that has an invalid return address.

Pandey and Swaroop were identified Performance Analysis of Proactive, Reactive and Hybrid Protocols in Ad-hoc environment [8]. The researchers were compared the Performance Routing Protocols using the throughput, average delay, routing overhead and Packet drop as evaluation metrics. Destination Sequence Distance Vector is a Proactive routing protocol which is using distributed Bellman-Ford Algorithm. The DSDV, every node in the network conserves a routing table that has records for each of the destinations in the network and the number of hops needed to reach each of them. The DSDV protocol periodically sends the routing updates for maintaining the reliability of the network. Therefore the traffic flow of the network is high. Aware the above issue the DSDV sends two types of update

packets known as full dump and incremental packets. Dynamic Source Routing another reactive routing protocol and it has two parts, namely the route discovery and route maintenance. The DSR does not send routing packets periodically. Therefore, traffic flow is very low compared to the DSDV protocol. The route discovery process finds the new route in the network and route maintenance allows maintaining the route in the network. The DSR Protocol is designed especially for mobile ad hoc networks of up to about two hundred nodes. AODV is a reactive protocol, it has characteristics of both DSR and DSDV Protocol. This protocol creates routes to the destination and support unicast as well as multicast routing. The AODV protocol maintains three types of messages, namely route request, Route Response and Route Error. The Zone Routing Protocol is a Hybrid Routing Protocol, which is a combination of both reactive and proactive routing protocols. Further, researchers stated the simulation results of routing protocol. For the simulation environment authors were used network simulator version 2 used and they interested throughput, average delay, Number of Packets dropped and Routing Overhead as performance metrics. Throughput calculated the total number of data packets transferred effectively from one place to another in a given period. Average Delay is defined as the average amount of time taken by a packet to go from source to destination. Routing Overhead is the number of routing packets required for network communication. No of Packet dropped is the number of packets lost by the router. Evaluating the Performance metrics Researcher has invented that ZRP and AODV have more routing overhead in comparison to DSR and DSDV routing protocols. However, The AODV and ZRP protocol have a higher average delay than DSR and DSDV protocol.

Kalia and Sharma were proposed the Detection of Multiple Black hole nodes attack in MANET by modifying AODV Protocol [9]. The researchers were describing the Overview of AODV Protocol. The protocol controls the three types of messages, namely Route Request, Route Reply and Route Error. Route Request message used for requesting a new route, whereas Route Reply message used for reply the route request. On the other hand Route Error message used to inform to the node if there any error or link failure in the network. Further, researchers were stated the AODV protocol black hole attack. In a black hole attack that

mischievous node consumes the routing path and stated it has the valid path to any desired in the network does not forward the packets to its neighbors. The researchers were described two types of black hole attack in AODV namely internal black hole attack and external black hole attack. Internal black hole attack occurred inside the network, whereas external black hole attack occurred outside the network. They also stated the detecting the Multiple black hole attack in AODV protocol in MANET. They compared the Route request Sequence number with Route Reply Sequence numbers to detect the multiple black hole attack.

Mistry et al. [10] were proposed the technique to prevent against Black hole Attack in AODV Protocol. The researchers were improving the security issues of AODV Protocol. They were interested to avoid the Black hole and shows major progress in Packet Delivery Ratio in AODV Protocol. Further, researchers were examined and introduce new MOS_WAIT_TIME variable and a new Cmg_RREP_Tab table. Using above two variables the researchers were successfully detected the black hole nodes on the AODV Protocol. The final results show that the Proposed AODV protocol reaches high Packet Delivery Ratio than the existing pure AODV protocol in MANET.

Raj et al. [11] were introduced Dynamic Learning System against Black hole attack in AODV based MANET. The Researcher stated two secure methods of MANET namely, secure routing and Intrusion detection System. Secure routing they mentioned Secure, Efficient Ad-hoc Distance Vector Protocol, the Authenticated routing protocol for Ad-hoc network and Security Aware ad-hoc routing. Intrusion Detection they mentioned Real time intrusion Detection for Ad-hoc network and Confirmation Request. Further, Researcher were introduced the additional parameter known as Threshold value. In every time interval the threshold value is updated. In the Proposed solution, the researcher were introduced the additional parameter known as Threshold value. In every time interval the threshold value is updated. If the Route Reply sequence number value is higher than the threshold value the particular node to be intruder node and it is reported to the blacklist. If the node detected irregularly, it generated the new control packet known as Alarm packet and forward it to other nodes in the

network. The Alarm packet has details of blacklist node, then the adjacent nodes know that RREP packet from the node will discard.

Panda et al. [12] were proposed the technique to Prevent of Black hole Attack in AODV protocols for Mobile Ad Hoc Network by Key Authentication. The researchers were described the black hole attack in AODV and prevent the Black hole attack for AODV Protocol. In a black hole attack the intruder node consumes the routing path and stated it has the valid path to any desired in the network does not forward the packets to its neighbors. Further, the researchers were proposing a new algorithm which was based on Key Mechanism process. The attacker wishes to trace the IP address. Avoid the black hole attack researcher took the 12 digit number left shift of binary numbers. After that researchers used mathematical AND operator for rearranging number to very difficult. If the step was succeeded next step key was fixed into 9 digits and allocated it on data packets. The valid data packets conclude the originator for IP address, Sequence number and Hop count. Finally the Hello message broadcast to the adjacent node or host. If it is successful, then the researchers move to the next step to detect the black hole node in AODV Protocol.

Ukil et al. [13] were introduced Mechanism for Detection of Cooperative Black Hole Attack in Mobile Ad Hoc Networks. The researchers were described security mechanisms against cooperative black hole attack in a MANET. Further, they stated that there were two parts of the Cooperative black hole attack. In the first stage malicious nodes broadcast that it has valid path to destination, in the second stage the malicious nodes stopped the packets without forwarding them. Avoid the Cooperative black hole attack in MANET Researchers were proposed two mechanisms which are Data routing information (DRI) table and Cross checking. In Data routing information describes two bits of additional information are sent by the nodes that respond to the message of a source node during the route discovery process. Each node maintains an additional data routing information (DRI) table. At the DRI table, the bit 1 stands for ‘true’ and the bit 0 stands for ‘false’. The cross Checking method describes their lies on nodes through which source has routed data previously and knows them to be trustworthy to transfer data packets.

Dr. Sankaranarayanan and Selven were introduced the technique to Prevention of Co-operative Black hole Attack in MANET [14]. The researchers were described Prevent Against the cooperative black hole attack in MANET. They proposed a solution that is an improvement of the basic AODV routing protocol, which will be able to avoid multiple black holes acting in the group. Further, researchers were presented a technique to identify multiple black holes cooperating with each other and a solution to discover a safe route avoiding cooperative black hole attack. The source node transmits the RREQ to all its neighbors. Then the source waits for 'TIMER' seconds to collect the replies, RREP. In each of the received RREP, the fidelity level of the responding node, and each of its next hop's levels are checked. If two or more routes seem to have the same fidelity level, then select the one with the least hop count otherwise select the one with the highest level. Further, they implemented the proposed solution in a Global Mobile simulator and shows that the improved AODV called PHCBA has a higher packet Delivery than the Pure AODV Protocol.

Himral et al.[15] were proposed to Preventing AODV Routing Protocol from Black hole attack. The researchers were proposed secure routes or methods to avoid the Black hole attack. They were interested in using the sequence number parameter in the data packets to implement the proposed method. Further, they compared if there any huge difference between sequence numbers in the node which was previously sent Route Reply message. The initial route reply will be from the malicious node with high destination sequence number, which is stored as the first entry in the Route Reply Table. After that it compare the first destination sequence number with the source node sequence number, if there exists much more difference between them, confident that node is the malicious node, immediately remove that entry from the Route Reply-Table. If the malicious or wicked node is identified in the routing table and removed. Further, they were implemented the solution in Network simulator version 2.34 and improved that the Proposed Program has a higher Packet delivery ratio than existing AODV Proposed Program's packet Loss ratio is less than the original AODV Program.

Saetang and Charoenpanyasak were introduced CAODV Free Black hole Attack in Ad Hoc Networks[16].The researchers were proposed a Credit Based AODV Routing protocol technique to prevent against the black attack in MANET. The CAODV method, the source node broadcasts a Route Request to other nodes until a destination node or node having a route to destination replies Route Reply back to the source. The receiving node will assign a credit to the next hop node or who sent Route Reply. When a node in the path sends one packet, one credit is deducted from the next hop node. As soon as a destination node receives data packet, it will send Credit Acknowledge back to a source node. A node within a way back will increase credit of the next hop by 2 to indicate a higher trust level of the next hop. On the other hand, credit will be decreased if a node cannot receive a Credit Acknowledge. The node will be untrusted and marked as a blacklist, when a credit reaches zero. Further, the researchers were implemented the solution in Network simulator version 2 and improved that the black hole attack cannot damage network when the CAODV is active.

Golok and et al.[17] Proposed Prevention of Black hole Attack in AODV protocols for Mobile Ad Hoc Network by Key Authentication. The researchers have proposed a new Algorithm for AODV routing protocol. Hello message spread to its neighbor nodes and finding the occurrences in the network. If the nodes are active each node should maintain a routing table. The researchers were proposed algorithm mainly created on Key generation and key comparison methods. Further, they were used Throughput, End-to-end delay of data packets, First Packet Received and Average jitter as the Parameter metrics. For the implementation they were used Qual Net 5.2 simulator.

Surana et al. [18] were introduced Securing Black hole attack in AODV routing protocol in MANET with Watch dog mechanism. The researchers were introduced Pending packet table and node rating table. The pending packet table includes the path of the packets which was sent. And also it includes the IP address of the destination node, IP address of the adjacent node, packet id and expiry time. In node routing table include the details of next adjacent node in the network and it also the special field which concludes dropped packets and successfully forwarded packets as a ratio. If the ratio is greater than the given threshold value, then the

variable value increased by one otherwise it is considered as unaffected node in the network.

2.2 Summarization of the Literature reviews

Based on the above literature review, the various authors have given various proposals for detection and prevention of AODV Cooperative black hole attack in Mobile Ad-hoc network. However every proposal has some limitations and their respected solutions. These Limitations are listed in the summarized in the table 1.0.

Table 1.0 Summarization of the identified issues in Literature Review

Reference No	Research	Limitation
3	A review of routing protocols for mobile ad hoc networks	Described pros and cons of each category of routing protocol, unable to provide the Security mechanism.
4	Performance analysis of AODV, DSR & TORA Routing Protocols	Described general performance of the Protocol, Did not provide the security mechanism of the Protocol
5	Routing Security in Wireless Ad Hoc Networks	Unable to detect cooperative malicious nodes in the MANET.
6	Performance comparison of Routing Protocols in Mobile Ad-hoc Networks.	Considered only the general performance of the Protocol, Not consider to prevent against attack in MANET.
7	Different Types of Attacks on Integrated MANET-Internet Communication	Described to Prevent the many attacks in MANET rather than avoiding Black hole attack.
8	Comprehensive Performance Analysis of Proactive, Reactive and Hybrid MANETs Routing Protocols	No proposal for the improvements Routing Protocol

9	Detection of Multiple Black hole nodes attack in MANET by modifying AODV protocol	Detecting the Multiple Black hole node unable to detect Cooperative Black hole node
10	Improving AODV Protocol against Black hole Attacks	The proposed solution to detect single black hole attack, did not propose a security mechanism to detect cooperative black hole attack.
11	Dynamic Learning System Against Black hole attack in AODV	Described to detect multiple black hole nodes rather than cooperative black hole attack.
12	Prevention of Black hole Attack in AODV protocols for Mobile Ad Hoc Network by Key Authentication.	Proposed Solution only detects the single black hole unable to detect cooperative black hole.
13	Mechanism for Detection of Cooperative Black Hole Attack in Mobile Ad Hoc Networks	Described to detect the cooperative Black hole node, the researcher assumes nodes are already authenticated
14	Prevention of Co-operative Black Hole Attack in MANET	Nodes are already authenticated and end-to-end delay is high
15	Preventing AODV Routing Protocol from Black Hole Attack	It is only detects the single black hole and removed unable to detect cooperative black hole.
16	CAODV Free Black hole Attack in Ad Hoc Networks	The Proposed solution can detect and protect a single malicious node, unable to protect cooperative black hole.
17	Prevention of Black hole Attack in AODV protocols for Mobile Ad Hoc Network by Key	Researchers were described to detect the single black hole unable to detect the cooperative black hole malicious

	Authentication	nodes.
18	Securing Black hole attack in Routing Protocol AODV in MANET with Watchdog	Proposed solution maintains two extra tables, namely Pending Packet table and rating table which required additional space of the node.

2.3 Summary

This Chapter described previous researcher's work related to the Review of Routing Protocol, Performance of Routing Protocol, Security issues and Prevent to detect a black hole attack in the AODV Protocol in MANET. Next Chapter will address the Research Methodology.

Chapter 03

Research Methodology

3.0 Introduction

Chapter 02 has stated the Previous, work related to the Review of Routing Protocol, Performance of Routing Protocol, Security issues and Prevent to detect a black hole attack in the AODV Protocol in MANET. Chapter 03 presents the Research Methodology of Detecting the Cooperative Black hole attack in AODV protocol in MANET.

3.1Methodology

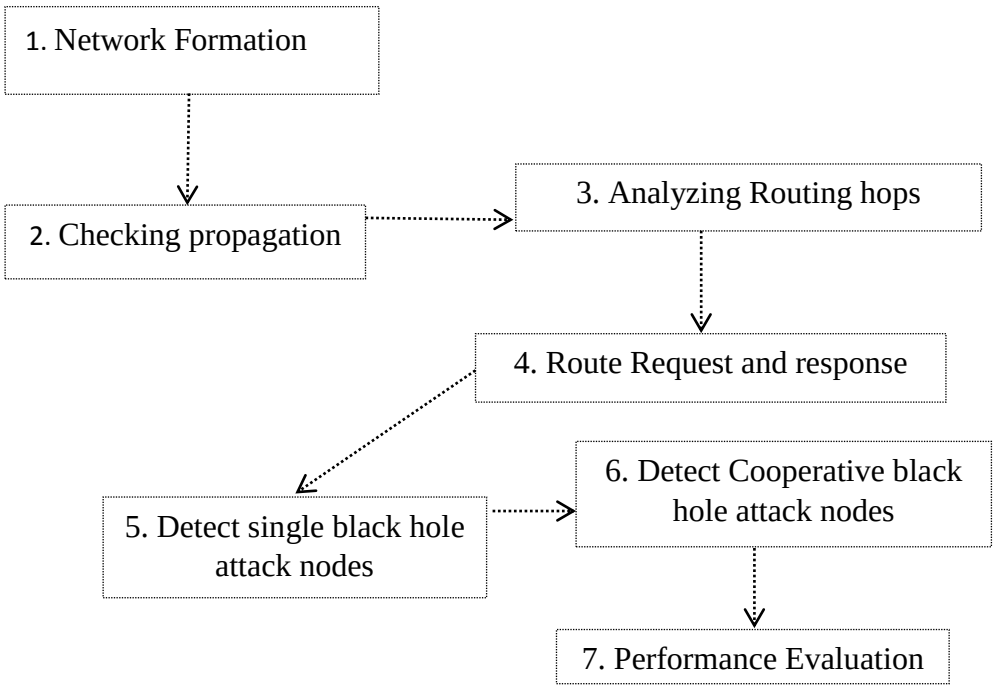


Figure 2. Research Methodology

Initial Stage of Research Methodology is Network formation**,** which is feature of network that finds to model how a network changes by identifying which factors move its erection and how these mechanisms work.

The second step of Research Methodology is checking the Propagation in the MANET. Propagation will identify the spread nodes in MANET.

The third step in the Methodology is Analyzing Routing hop**.** It is one portion of the path between source and destination. Data packets pass through bridges,

routers and gateways as they travel between the source and destination. Each time packets are passed to the next network device, a hop count occurs.

The fourth step of the Methodology is Route Request and Route Response. Route Request and Route Response are two messages in AODV Protocol. Route Request is requesting route to the source node to the destination node and Route Response is responding the valid route to the Source node. The AODV Protocol used both Route Request and Route Response for establishing a valid path.

Fifth step of the methodology is detecting the black hole attack. A black-hole attack in the Mobile Ad-hoc Network is an attack occurs due to malicious nodes, which attracts the data packets by falsely advertising a fresh route to the destination. Therefore, in this step will detect the single black hole attack nodes in MANET.

The sixth step of the methodology is detecting Cooperative black hole attack. In cooperative Black hole attack, malicious nodes act as a group or unit. Therefore, in this step will detect the cooperative black hole attack nodes in MANET.

Final Step of the methodology is Performance Evaluation. Evaluate and compare the Performance of Original AODV with improved AODV, in this case researcher interesting about routing delay, Packet delivery ratio, Packet Loss and throughput. The Detailed graph will be shown in the Evaluation Chapter.

3.2 Improved Ad-hoc demand Distance Vector Protocol

There are two main objectives of the research. The main objective detects the cooperative Black hole attack. And the second one is to improve performance of AODV by reducing the routing delay, increase the packet delivery ratio, increased the throughput and reduce packet loss ratio of ad-hoc network. For the aim of the second purpose researcher introduce the Improved Ad-hoc Demand Distance Vector Protocol. The flow chart of the improved AODV protocol will be shown in Annex C.

3.3 Summary

This Chapter described the Research Methodology is used in the dissertation and the introduction of improved AODV protocol. There were seven steps used in the methodology. Each step was mentioned clearly. The next chapter will address the Overview of Mobile Ad hoc network and MANET Routing Protocol.

Chapter 04

Overview of Mobile Ad hoc Network and MANET Routing Protocol

4.0 Introduction

Chapter 03 has stated the Research Methodology of Detecting the Cooperative Black hole attack in AODV protocol in MANET. Chapter 04 presents the Overview of Mobile Ad hoc network and MANET Routing Protocol.

4.1 MANET

A Mobile Ad-hoc network is a kind of network it has the ability to change and configure itself. Nodes can be easily join and leave the network. There is no central control mechanism to control the nodes. Which is the main feature of Mobile Ad-hoc network. The Mobile Ad-hoc network has many Positive features and Negative features. Mobility and the topology changes are Positive features of MANET and Power consumption, Energy and computing power are negative features of MANET[19].

4.2 MANET Characteristics

The Mobile Ad-hoc network has many characteristics. The main characteristics are the lack of centralized control. Other characters are namely Multi hop routing, Autonomous Terminal, Distributed operation, Dynamic topology [20]. Each of the Characteristics is explained below.

1. Lack of centralized control: Mobile Ad-hoc network does not have centralized control. Therefore, each node or host in the network operates independently.
2. Multi hop routing: MANET uses two or more nodes to transfer the data packets from source device to the destination device.
3. Autonomous Terminal: Mobile Ad-hoc environment each and every node are independent nodes. Therefore nodes are operated as both host and router in the network.
4. Distributed Operation: Mobile Ad-hoc network is a kind of distributed network, which does not have fixed network. Therefore nodes or host in the network finds the route easily.

5. Dynamic topology: nodes in the MANET are easily joined and leave the network. Therefore, topology in the MANET is changed randomly and frequently.

4.3 MANET Applications

Mobile Ad-hoc network has the capability self-organizing feature. Therefore, MANET Applications are very popular. It has implemented in many areas namely Military Application, Mobile Computing, Personal Are Networking, Mobile Business and Emergency Service. Each of the applications is described in detail[21].

Military Application: in the Battlefield Mobile Ad-hoc networking very helpful to military peoples to exchange their information between the soldiers, vehicles, and military information headquarters over the normal networking.

Mobile Computing: Ad-hoc networks allow mobile communication for commercial users to communicate in the remote location when there is no other network facility is available.

Personal Area Networking: Ad-hoc network is suitable for small area networking applications. Mobile devices with WLAN can be easily configured. If the internet connectivity available at home MANET device can be easily accessed via remote locations

Mobile Business: Ad-hoc networks allow the users to pay the payment electronically anytime, anywhere. It is a tremendous advantage which will save money and time of Business peoples.

Emergency Services: Once the available existing network facility has terminated or damaged due to disaster or fire ad-hoc network can be easily established and provide the solutions to emergency services.

4.4 MANET Limitation

Due to the flexibility of MANET the Applications of MANET become very popular. However, it has some limitations as well. Which are listed as,

1. Security: security is the main key drawback of Mobile Ad-hoc network. MANET nodes are easily can be accessed. Therefore, nodes in MANET are more vulnerable to attack than wired networks.
2. Nodes speed: in MANET, the speed of the nodes supports a maximum of 11 Mbps. On the other hand Wifi devices should be supported to 54 Mbps or higher speed.
3. Error rate is very high: compared to the wired network MANET has a higher error rate. This is another key issue in MANET.
4. Routing: In a MANET, the topology of the network will change frequently, therefore the issue of routing packets between any pair of the network is a difficult task.

4.5. MANET Routing Protocol

There are many Protocols are implemented in Mobile Ad-hoc network. These Protocols are mainly classified into three categories according their performance and functionality. This is shown below,

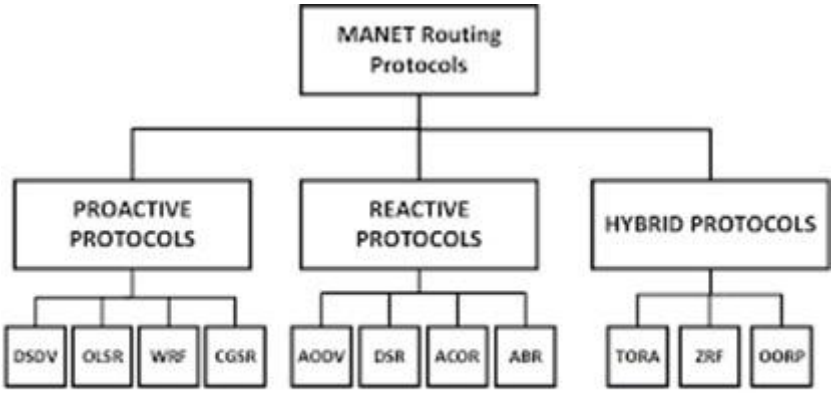


Figure 3. MANETs Routing Protocols

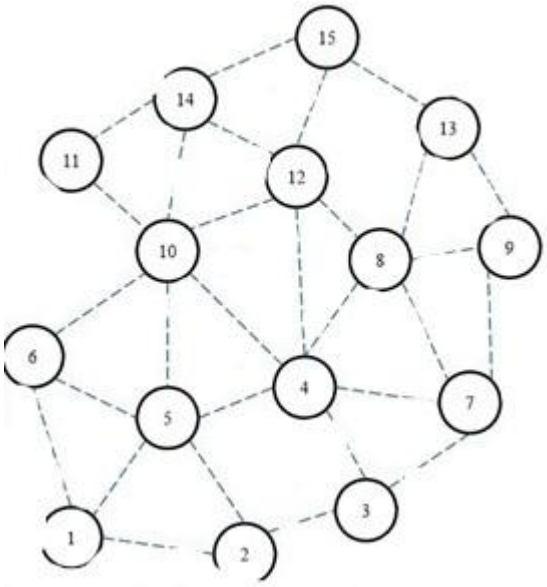
4.5.1. Proactive Routing Protocol

In Proactive Routing each node or host in the network preserves one or more routing tables which keep the information about the entire topology of the network. These routing tables are updated frequently. Destination Sequence Distance Vector Protocol, Fisheye State Routing (FSR) protocol and Optimized

Link State Routing Protocol are examples of Proactive Protocols in Mobile Ad-hoc network. Each of the Protocol are described below.

4.5.1.1 Distance Sequence Distance Vector Routing Protocol

It is table driven or proactive protocol functioning in Mobile Ad-hoc network. DSDV is an improved protocol of Distributed Bellman-Ford algorithm. DSDV protocol requires periodically broadcasting hello messages to each and every node in the network to maintain the link and every node in the DSDV protocol has a separate table it includes shortest path. DSDV Protocol uses bidirectional link. Which is a main feature of DSDV protocol. However the protocol provides only particular route from source to destination. DSDV protocol maintains a sequence number for each entry in the routing table. When there is a change in the network the node will send an updated message to the adjacent node by incrementing the sequence number. Every DSDV node conserves separate routing tables for packet sending and publishing the incremental routing packets. If there are any changes occurred in the network, an identifying node sends an update packet to its adjacent node in the network. Once the adjacent node received an update packet from an adjacent node, a node abstracts the information from the packet and updates its routing table. Following figure shows the routing table operations of the DSDV protocol .



Routing table for Node 1

Dest	NextNode	Dist	seqNo
2	2	1	22
3	2	2	26
4	5	2	32
5	5	1	134
6	6	1	144
7	2	3	162
8	5	3	170
9	2	4	186
10	6	2	142
11	6	3	176
12	5	3	190
13	5	4	198
14	6	3	214
15	5	4	256

Figure 4. DSDV Protocol

4.5.1.2 Fisheye State Routing (FSR) protocol

Fisheye state routing protocol is another table driven protocol. FSR Protocol uses the feature of the fisheye technique to transmit the routing update information with dissimilar varied frequencies in that way reducing the routing overhead and reducing flooding update information. The above protocol three has major functions namely,

1. Neighbor Discovery: maintain the relationship with other nodes broadcasting hello messages.
2. Information Distribution: each node broadcasting Link state announcement message to all other nodes in the network.
3. Route Calculation: node can reconstruct the network topology by calculating the link state announcement. [23]

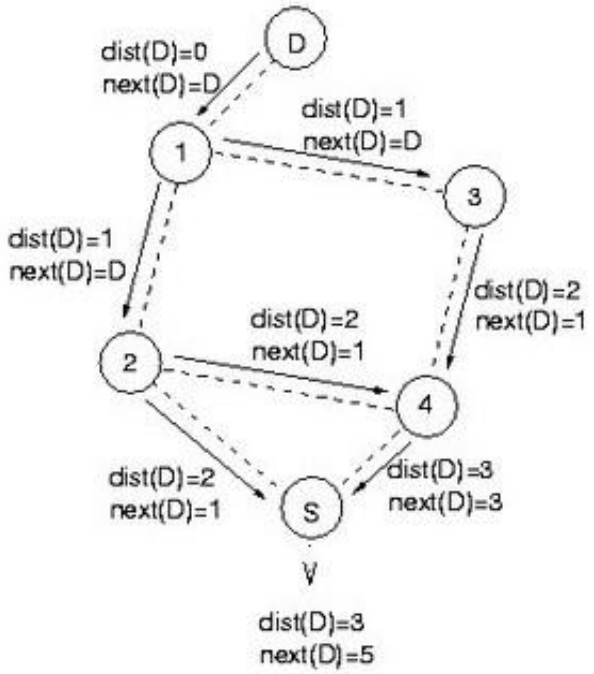


Figure 5. Fisheye Routing Protocol

4.5.1.3 Optimized Link State Routing Protocol

Optimized Link State Routing Protocol is a Proactive Routing Protocol designed for Mobile ad hoc environment. It is finding the alternate path of existing link state protocols in that it decreases the size of control packet and reduces the transmission of control packets. The OLSR protocol exchanges the routing information with adjacent nodes regularly. OLSR Protocol uses Multipoint Relay Packets to reduce the data traffic in the network. A Multipoint Relay is a kind of a

node's one-hop neighbor, which will decide to forward the data packets. The following figure shows the Operation of optimized Link State Routing Protocol [24].

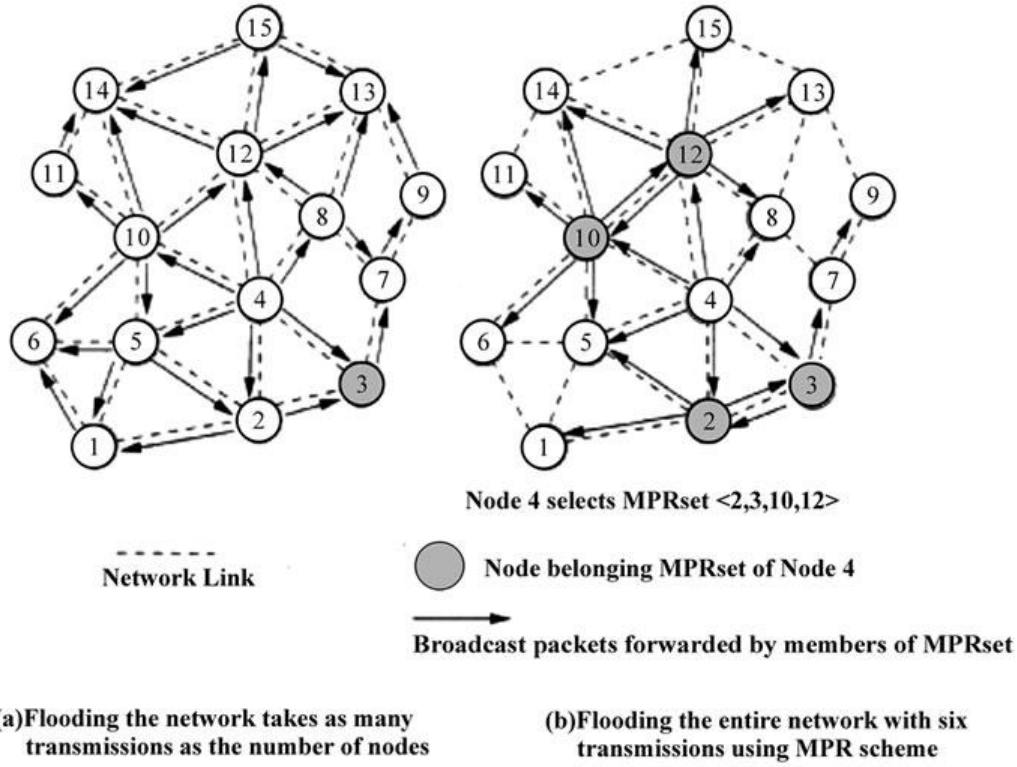


Figure 6. Optimized Link State Routing Protocol

4.5.2. Hybrid Routing Protocol

Hybrid Routing is the combination of both Distance vector protocol and Link state routing protocols and act as a new protocol. Temporary ordered routing Protocol and Zone routing protocol are example of Hybrid Routing Protocol.

4.5.2.1Temporary Ordered Routing Protocol

The main aim of the TORA is to control the message broadcast in Mobile Ad-hoc network. It refers that TORA is designed to reduce the network overhead by adjusting the local topology changes in MANET. TORA protocol supports the multiple paths to send the data packets from source node to destination node [25].

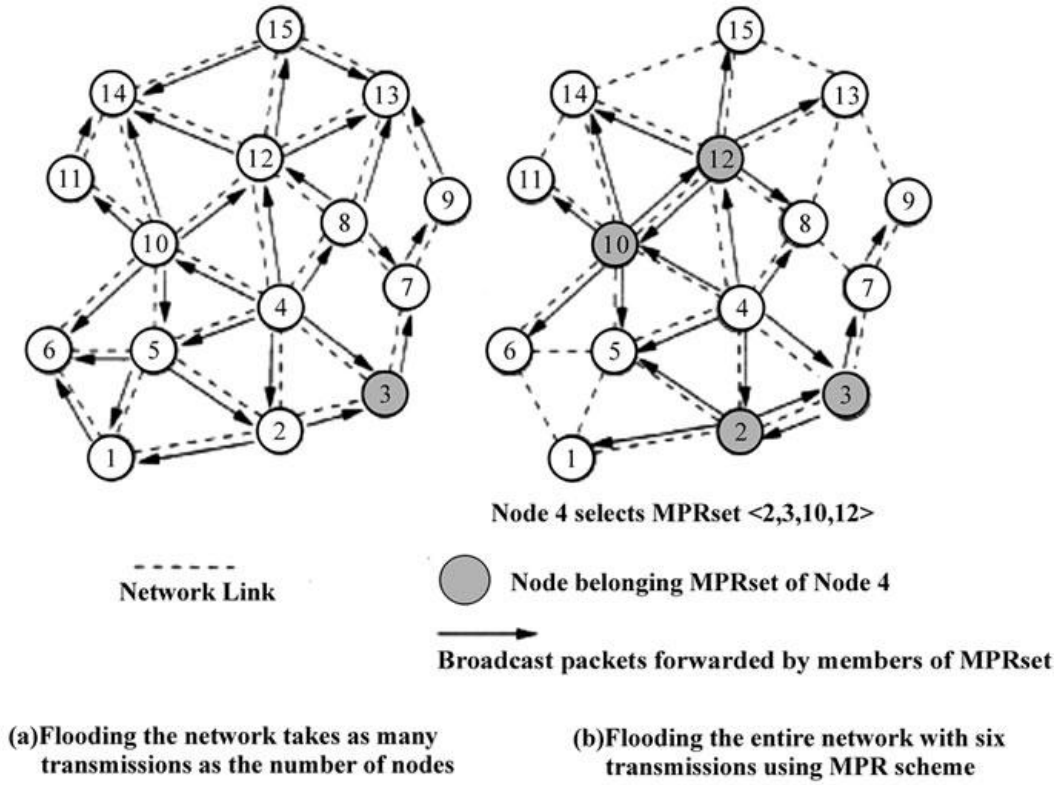


Figure 7. Temporary Ordered Routing Protocol

4.5.2.2 Zone Routing Protocol

Zone Routing Protocol was invented by Zygmunt Haas of Cornell University. ZRP is Hybrid Protocol, which uses the combination of Distance vector protocol and Link state routing protocol when transferring data packets over the network. And it finds the loop free routes to destinations. The ZRP main concept is based on Zones. Zone consists of several nodes whose minimum length between the nodes is defined as Zone radius. The main objective of ZRP was designed to speed up sending and decrease the processing delay [26].

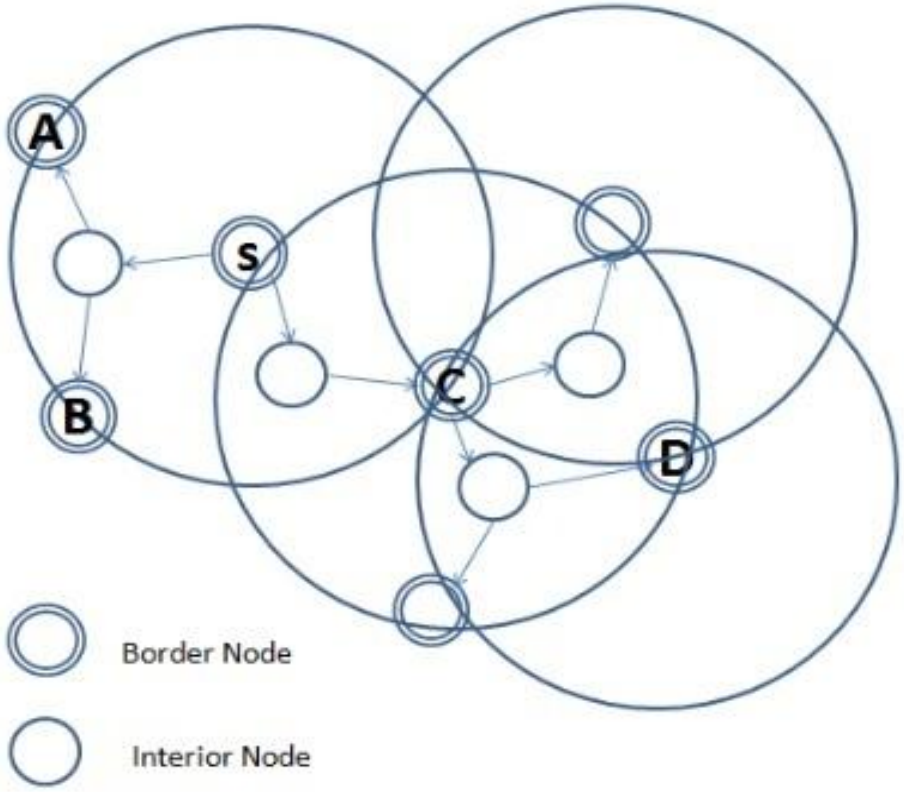


Figure 8. Zone Routing Protocol

4.5.3. Reactive Routing Protocol

Reactive routing protocol does not make the nodes initiate a route discovery process until a route to a destination is required. It is a bandwidth efficient on-demand routing protocol for Mobile Ad-Hoc Networks. The protocol has many features which are listed as,

- Do not consume bandwidth for sending information.
- Protocol consumes bandwidth only, when the node start transmitting the data to the destination node.

Ad-hoc on Demand Vector Protocol and Dynamic Source Routing Protocols are examples of Reactive Routing Protocols. There are various parameters available in Reactive Protocols, which are throughput, delay time, Packet Delivery Ratio and Packet Loss Ratio. With the aim of research, researcher interesting to detect the Cooperative Black hole attack nodes in Ad-hoc on Demand Distance Vector Protocol in Mobile Ad-hoc network environment.

4.5.3.1 Ad-hoc on demand Distance Vector Protocol

An Ad-Hoc On demand Distance Vector is a kind of Reactive Routing Protocol, which was jointly developed by Nokia Research Center, the University of California, Santa Barbara and the University of Cincinnati in 1991. The main features of AODV protocol is it supported both unicast and multicast routing.

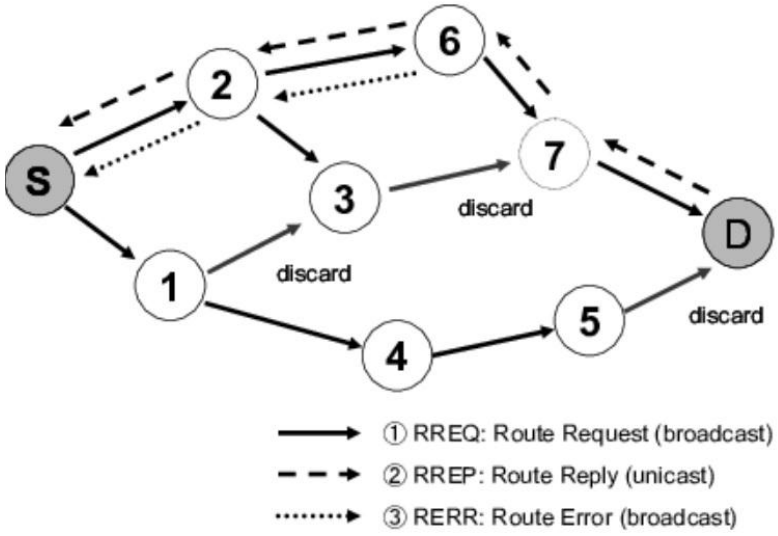


Figure 9. AODV Routing Protocol

There are two main processes implemented in the AODV protocol, which are known as route discovery and route maintaining. There are mainly four numbers of messages are implemented in AODV Protocol, namely as Route Reply Message, Route Request Message, Route Error Message and Hello Messages. Each of the messages is described below with diagram[4] .

4.5.4 Route Request Message

If the nodes in the network need to establish a link with destination node it sends Route request broadcast the message in the network. The Route Request message format is attached is given below[27].

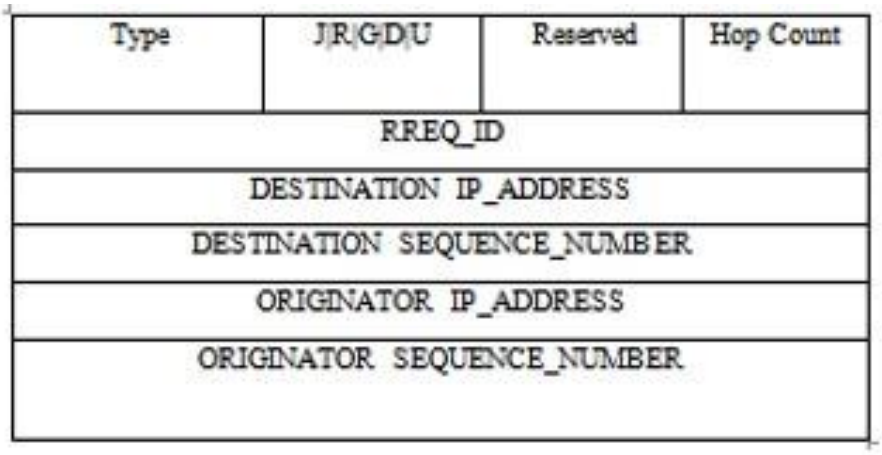


Figure 10. Route Request Message Format

Type: Type described the message type, By default it should be set at 0

Reserved: this is for future use. By defualt it should be set as 0;

Hop Count: the number of hosts or nodes from the Discover IP Address to the node.

RREQ ID:it is a unique number identify the particular Route Request.

Destination IP Address: IP Address of the Destination Node.

Destination Sequence Number: The up-to-date sequence number received in the past by the sender

Originator IP Address: IP Address of the node or host invented the Route Request

Originator Sequence Number: The Present sequence number to be used in the route entry pointing.

4.5.5 Route Reply message

Once the node receives a route request message, it should be discarded and a route reply message will reply. The Route Reply message format is given below[27].

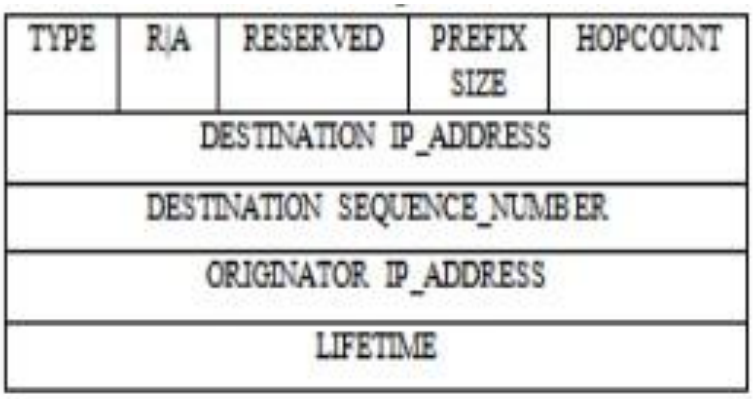


Figure 11: Route Reply Message Format

Destination IP Address: IP Address of the Destination Node.

Destination Sequence Number: The up-to-date sequence number received in the past by the sender.

Originator IP Address: IP Address of the node or host invented the Route Request

Originator Sequence Number: The Present sequence number to be used in the route entry pointing.

Lifetime updated every time the route is used.

4.5.6. Route Error

If there is any link break occurs in the network the route error message is forwarded to the nodes. The Error message format is given below[27].

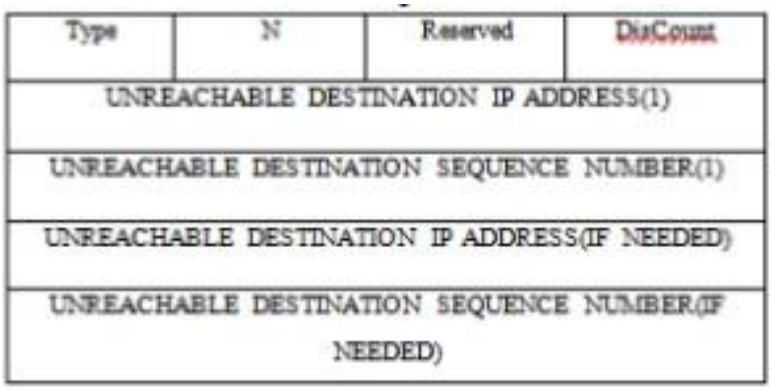


Figure 12. Route Error Message Format

4.5.7. Route Discovery Process

The route finding process is extended by broadcasting packets to entire nodes in the network. Route Request and Route Reply messages are detected to find the route between source to destination[28]. The following figure shows the AODV Protocol Route discovery.

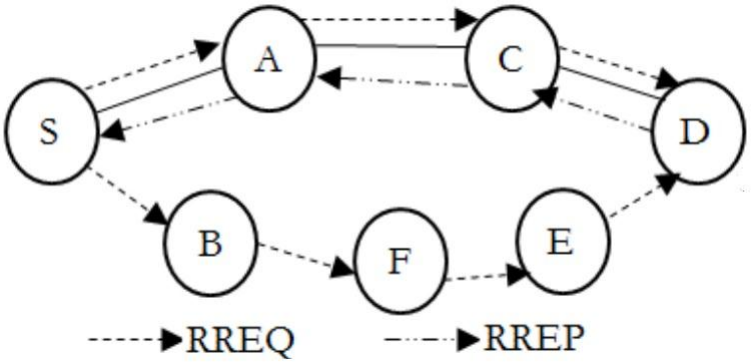


Figure 13. Route Discovery

4.5.8. Route Maintenance Process

In AODV protocol, the Hello message is used for determining link connectivity between nodes. If the particular route is active, the node should use the Hello messages to maintain the particular route. The route Error message is informed when there is link failure occurred in the network [28]. The Route Maintenance Process is shown in following figure.

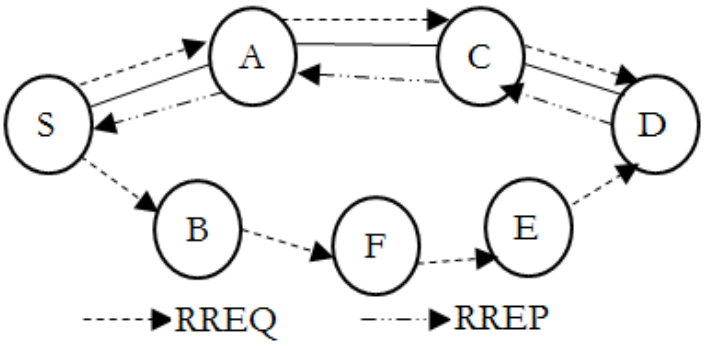


Figure 14. Route Maintenance Process

4.5.9. Merits and Limitations of AODV Protocol

There are certain merits and limitations of the AODV Protocol, which are listed below.

4.5.9.1 Merits of AODV Protocol

1. AODV Protocol support unicast and Multicast routing
2. The Protocol is loop free and self-starting to a large number of nodes.
3. The Protocol does not need any central administrative system to handle the routing process.
4. The protocol respond time is very quick. If the network topology change it will quickly respond to the nodes.

4.5.9.2 Limitation of AODV Protocol

1. The AODV Protocol utilizes large share of bandwidth.
2. It takes much considerable time to build the routing table
3. It has high processing demand
4. If there is link failure, the AODV protocol generate a large number of packets, this may lead to congestion of the network.

4.6. Summary

This chapter described the Overview of MANET Structure and Routing Protocols namely Proactive, Reactive and Hybrid Routing Protocol. Each of the Routing Protocols is described clearly with diagram. Next chapter will address the Security Attacks on Mobile Ad-hoc network.

Chapter 05

Security Attacks on Mobile Ad-hoc network

5.0 Introduction

Chapter 04 has stated the Overview Structure of MANET and Routing Protocol on Ad-hoc network environment. Chapter 05 presents Security Attacks of Mobile Ad-hoc network.

5.1Security Attack

Mobile Ad-hoc Network is not secure and very flexible for the nodes. Nodes can be easily join and leave the network. There is no central control mechanism to control the nodes. Therefore, many security attacks occurred in MANET. Security Attacks mainly classified as Active Attacks and Passive Attack[7].

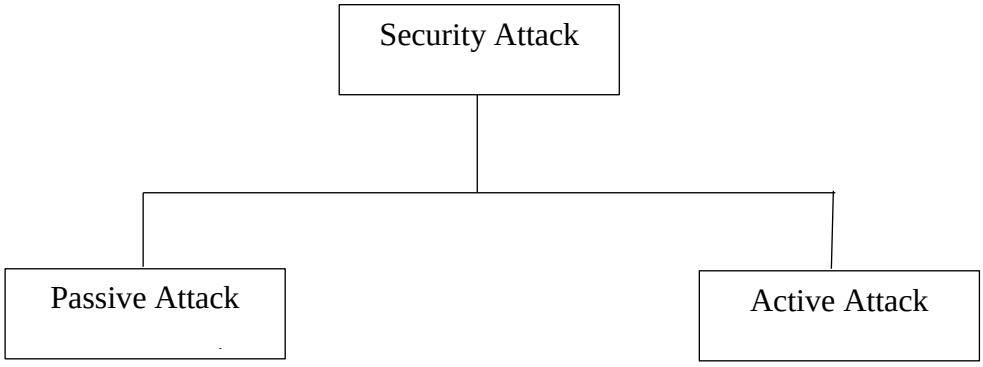


Figure 15. Classification of Attack

5.2. Active Attack

An active attack, attacker can change the packet or drop the packets which are being transferred in the network. There are two kinds of active attacks, namely internal or external active attack. In the External attack conceded out by host or nodes that do not belong to the area on the network. On the other hand internal attack is from contained host or nodes which are located inside in the network[29]. There are many active attacks in MANET namely rushing attack, Flooding attack, Gray-hole attack, Denial of service attack, Man-in-middle attack, wormhole attack and Black hole attack. On the other hand Traffic Monitoring, Eavesdropping and Traffic Analysis are Passive attacks.

5.2.1. Rushing attack

The rushing attack is a kind of active attack occurred in Mobile Ad hoc network. Rushing attack uses replicate destruction technique by quickly forwarding route discovery packets. In other words, it is a type of Denial of service attack. This attack can be performed in two ways. Sending a false route request id is a first one, the other one is sending false route request to false destination. These causes intermediate nodes to send many messages. In order to prevent the rushing attack, there were many techniques implemented, namely secure neighbor detection, and secure route delegation, specifying timeout and randomized message forwarding [30].

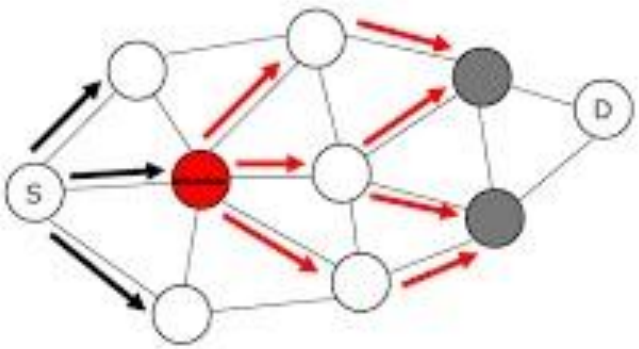


Figure 16. Rushing Attack

5.2.2. Flooding attack

Flooding attack is a type of Active attack occurred in Mobile Ad hoc network in the attacker node sends unwanted false duplicate packets to the network and absorb the network bandwidth, battery power and nodes other resources and making the destination nodes to receive the actual data packets. Flooding attack can be classified as hello flooding, RREQ Flooding and Data flooding. In Hello Flood attack, hello packets are broadcast with very high power. Therefore, a large number of nodes in the network will select it as the main node in the network. Using identifying verification protocol is a major technique to protect against Hello flooding attack[31].

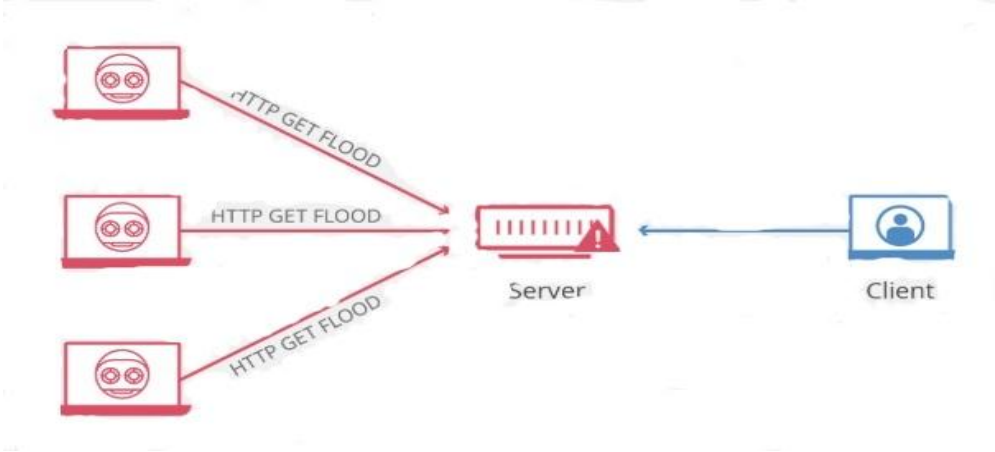


Figure 17. Flooding Attack

5.2.3. Gray-hole attack

This attack also called as misbehavior attack. The main aim of the Gray-hole attack is to drop messages. It has two phases. In the initial part it broadcasts it contains the actual path to destinations and other phase it drops the packets which are received from the adjacent node. The main aim of this update is to forward all the packets to intruder or malicious node rather than original node. In the second part, gray hole attack where intruder node dropped the interjected packets with a certain probability. To avoid the Gray-hole attack in MANET the Gray-hole prevent technique is used[32].

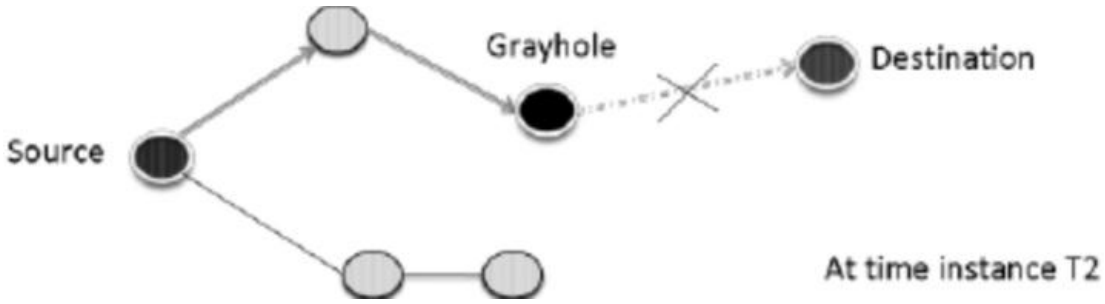


Figure 18. Gray hole Attack

5.2.4. Denial of Service attack

A Denial-of-service is any type attack where the intruder or attacker tries to avoid the authorized users from using the service or network. In DOS, the intruder broadcasts the data packets and asking the server to agree to accept the unwanted data packets from illegal return addresses. Then the network is unable to find the return address of the intruder when sending validation approval. This will continue

until server or networks find the return address and it makes network is busy. A denial of service attack is performed in many ways. Preventing a particular individual from accessing services, disrupting connections between two machines, disrupting a service to a specific system and disrupting the state of information are the form of denial service attacks. There are many techniques were proposed to avoid Denial of service attack, namely thread model and Dealing with Misbehaving Nodes [33].

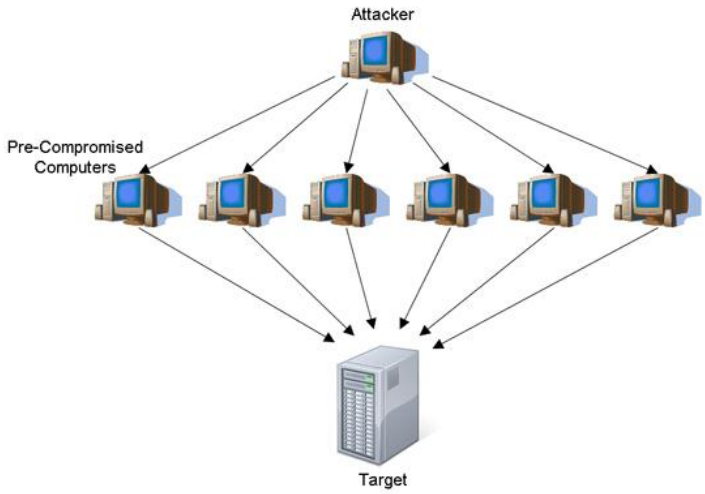


Figure 19. Denial of Service Attack

5.2.5. Man-in-the-middle attack

A man-in-middle attack is a type of active attack when an intruder or attacker capture the communication between two parties secretly adjust the message transferring between source node to the destination node. Man-in-middle attack has two different phases, namely interception and Decryption. In interception, the sending data packets are stopped over the intruder's network until it touches specific destination on the network. Interception can be achieved by IP spoofing, ARP spoofing or DNS Spoofing. In Decryption phase data traffic can be modified without the involvement of the sender. Decryption can be achieved by HTTPS Spoofing, SSL Hijacking, SSL Beast or SSL Stripping. Prevent against the Man-in-middle attack there were many techniques proposed. Strong WEP/WAP Encryption, Virtual Private Network and Public key pair based authentication were some of them[34].

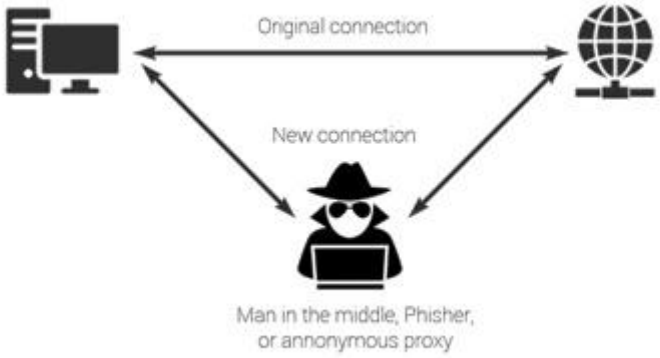


Figure 20. Man-in-middle attack

5.2.6. Wormhole attack

Worm hole attack is a server type attack for Mobile ad hoc network. In worm hole attack two more malicious nodes are connected channel called wormhole link. In this attack, attackers receive packets in a certain location in the network and tunnel them to another location in the network, and then repeat them into the network from that location. In other words, routing messages are forward in the wrong direction. Therefore, this attack performs tremendous effect on Mobile ad hoc network. To avoid the wormhole attack in MANET Random Approach Transfer Model, Packet leash and Hello message timing interval procedure are implemented in MANET[35].

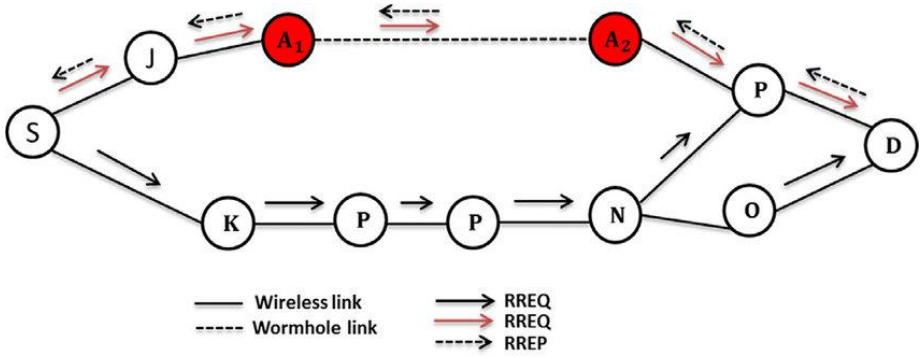


Figure 21. Wormhole attack

5.2.7. Black hole Attack

Black hole attack is a special Active type attack occurred in Mobile Ad hoc network. In a black hole attack a malicious node consumes the routing path and stated it has the valid path to any desired in the network does not forward the packets to its neighbors. These malicious black hole node executes various dangerous actions on the network, which are represented as a source node by

wrongly sending the Route Request message, represented as a destination node by wrongly sending the Route Reply Packets and decreasing the hop count. There are two types of black hole attack happened in the Mobile Ad-hoc network. Which are known as Single Black hole attack and Cooperative Black hole attack.

5.2.7.1. Single Black hole Attack

The black hole attack contains two stages. In the initial stage the hateful node's adventures the AODV protocol to present it contains the shortest path destination even though the particular path is not correct. In the second stage hateful node discards the interrupted packets without broadcasting. Following figure shows the single Black hole attack in MANET[36].

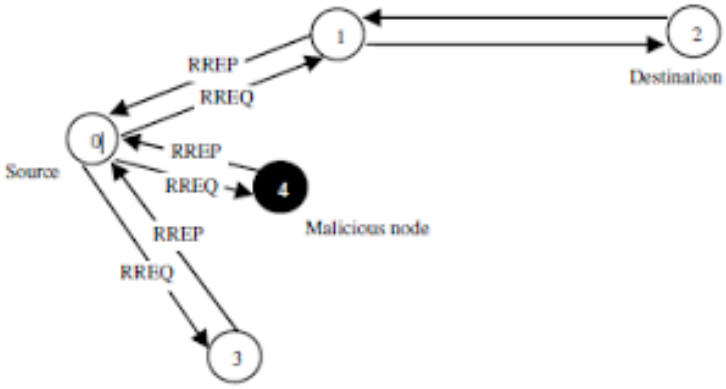


Figure 22. Single Black Hole attack

5.2.7.2. Cooperative Black hole attack

In Cooperative Black hole attack there are more than one malicious nodes are working as group to interrupt the routing protocol specification [36]. The following graphs show the cooperative black hole attack in Mobile Ad-hoc network. With the aim of research, researcher interesting to detect the cooperative Black hole attacking nodes in AODV protocol in Mobile Ad-hoc network.

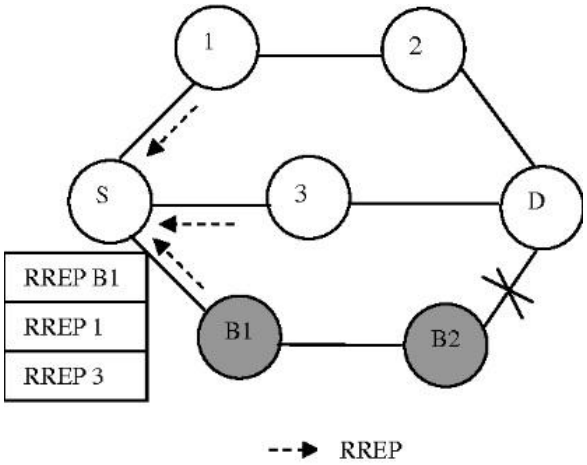


Figure 23. Cooperative Black hole attack

5.3. Passive Attack

A Passive attack is a kind of attack where an attacker monitoring transferring data packets over the network. This can happen in many forms, such as tracing the packets or reading the unauthorized email. Contrast to an Active attack, passive attack intruder or attacker does not modify or change the transferring data pass through the network[37].

5.3.1. Traffic Monitoring

Traffic monitoring is the procedure which a person can track the information about the network and which resources are being used each device in the network[38].

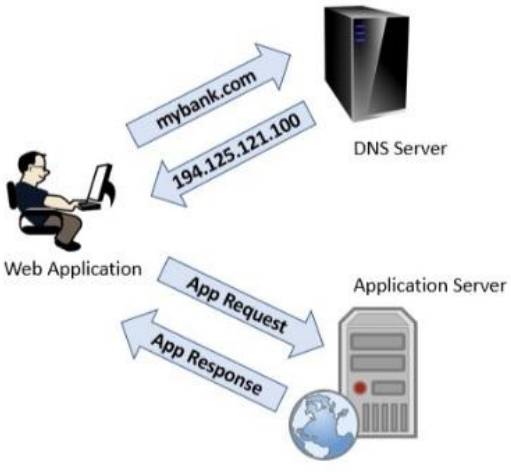


Figure 24. Traffic monitoring attack

5.3.2. Eavesdropping

An Eavesdropping is another kind of Passive attack which is also known as snooping attack. In Eavesdropping, the attacker wishes to steal valuable information which is transferred through the network. This information includes the Public key, Private Key and the Password [39].

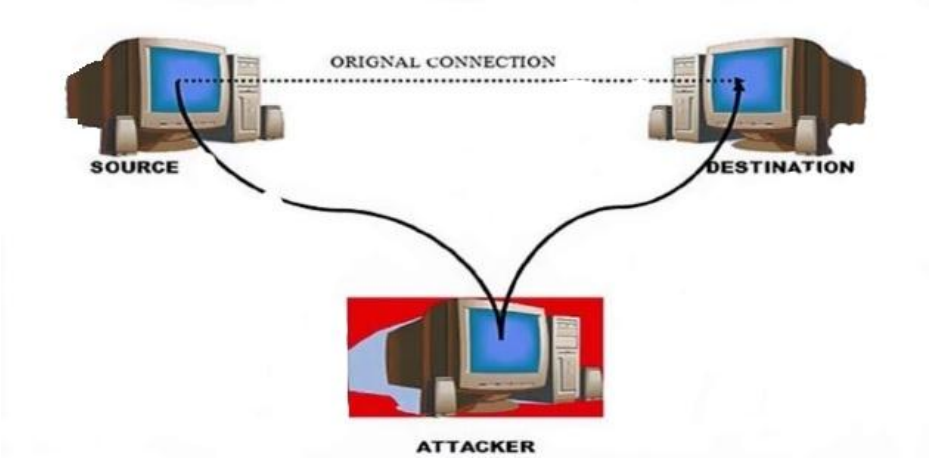


Figure 25. Eavesdropping Attack

5.3.3. Traffic Analysis

Traffic Analysis is a method of capturing and inspecting transferring the data packets over the network. The attack can perform Even if the data packets are encrypted. In other words the attacker listens the data packets which are transferring between source node to destination node[40].

Figure 26. Traffic Analysis

5.4. Summary

This Chapter described the Security attacks which were known as Passive attack and Active Attack. There are many types of Active Attacks namely Rushing attack, Flooding attack, Gray-hole attack, Denial of service attack, Man-in-middle

attack, wormhole attack and Black hole attack. Traffic Monitoring, Eavesdropping and Traffic Analysis are Passive attacks. Next Chapter will address the Software Requirement & implementation of the Project.

Chapter 06

Software Requirement & Implementation of the Project

6.0. Introduction

Chapter 05 has stated the Security Attacks of Mobile Ad-hoc network environment and also stated the Each and every Attack detecting technique. Chapter 06 presents the Software Requirement & implementation of the Project.

6.1. Software Requirements

6.1.1. Ubuntu

Ubuntu is an open-source operating system based on the Debian GNU/Linux distribution. Ubuntu cooperate all the features of a Unix OS with an added customizable GUI, which makes it popular in universities and research organizations. Ubuntu is primarily designed to be used on personal computers. Researcher is interesting to use Ubuntu 14.04.1 version of Ubuntu for implementing the research.

6.1.2. Network Simulator

Network Simulator Version 3, widely known as NS3, is new simulator written from scratch. It is supported C++ and Python language. It will depend on the ongoing contributions of the community to develop new models, debug, or maintain existing ones, and share results. The researcher is interesting to use ns3. 24 for implementing the research. The result of an output trace file that can be used to do data processing (Calculate Delay, throughput, etc.) and to visualize the simulation with a program called Network Animator, which is a very good visualization tool that visualizes packets as they transmit through the network. An overview of how a simulation is done ns3 shown below.

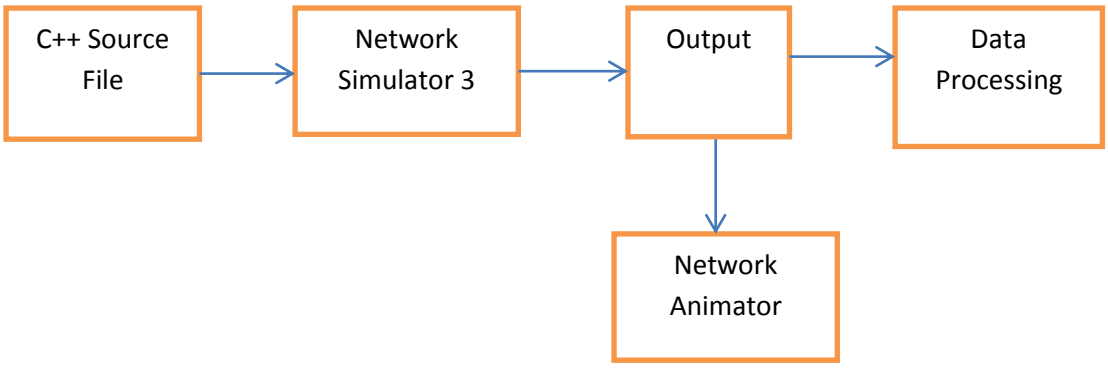


Figure 27. Network Simulation Diagram

6.2. Implementation

The implementation of the project is proposed to Linux environment. Therefore, Beginning of the research Researcher installed the virtual machine software to windows environment Computer. Then the researcher installed Ubuntu 14.04 on the virtual machine. After installing the Virtual machine, next step is to install the Ubuntu. Then the researcher needs to install network simulator 3.24 to the Ubuntu virtual machine. The final step is to install the black hole attack file to network simulator 3.24. After successfully installed Black hole attack program he needs to run AODV program named as OAODV.cc which contains the black hole node. Finally, researcher finds the Cooperate black hole detection code and named as MDPRAODV.cc file and run MDPRAODV.cc

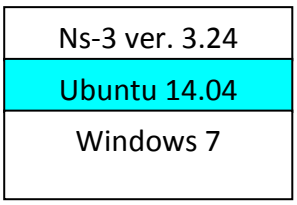


Figure 28.Ns3 over Ubuntu 14.04

6.3. Simulation Environment

The identification of Black hole attack is done in Network Simulator Version 3.24.

Table 2.0 Simulation Setup Parameters

Parameters	Value
Simulator	Ns-3
Version	Ns-3.24
Simulation Time	420 s
Number of nodes	100
Routing protocol	AODV
No. Of malicious node	10
Movement model	Random waypoint
Simulation Area	800 * 600

6.4 installing the Black hole Program

The first step of the Simulation is to implement the Black hole attack program to the network simulator. If the patch file was successfully installed, following output will be displayed.

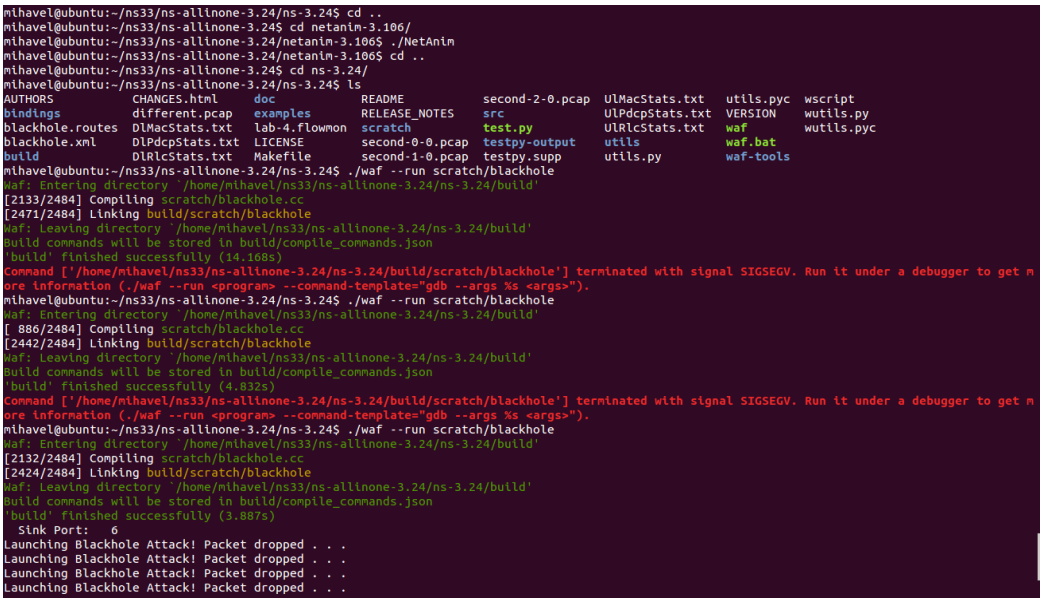


Figure 29. Black hole Program installation

This Black hole program allows a network simulator 3 to pretend Black hole attack in Mobile Ad-hoc environment. The above program version is black hole 3.24. If the black hole attack is installed successfully the “Black hole Attack! Packet dropped” message displayed as an output. The above message describes the data packets inside the network is transmitted instead of destroying the packets.

6.5. Compiling the Ad-hoc on Demand Distance Program

The next step of the Simulation is to run the Pure or Original AODV file known as OAODV.cc. The running command of OAODV.cc file is given below.

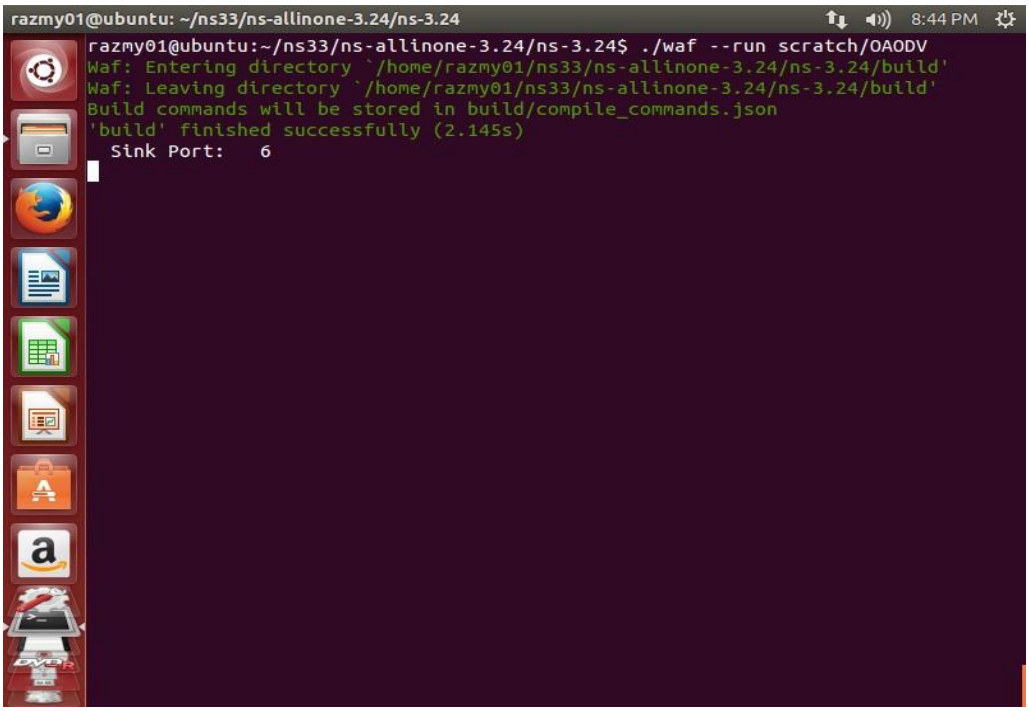


Figure 30. Compiling the Pure OAODV Program

The Pure OADV file is located inside the scratch folder. Scratch folder is one of the folders freely distributed folder in network simulator 3. Which is located inside the nose-allinone 3.24. The entire source files are located in a Scratch folder.

6.6 Out of Original AODV

If the Original AODV is running successfully following Message will be displayed on the screen.

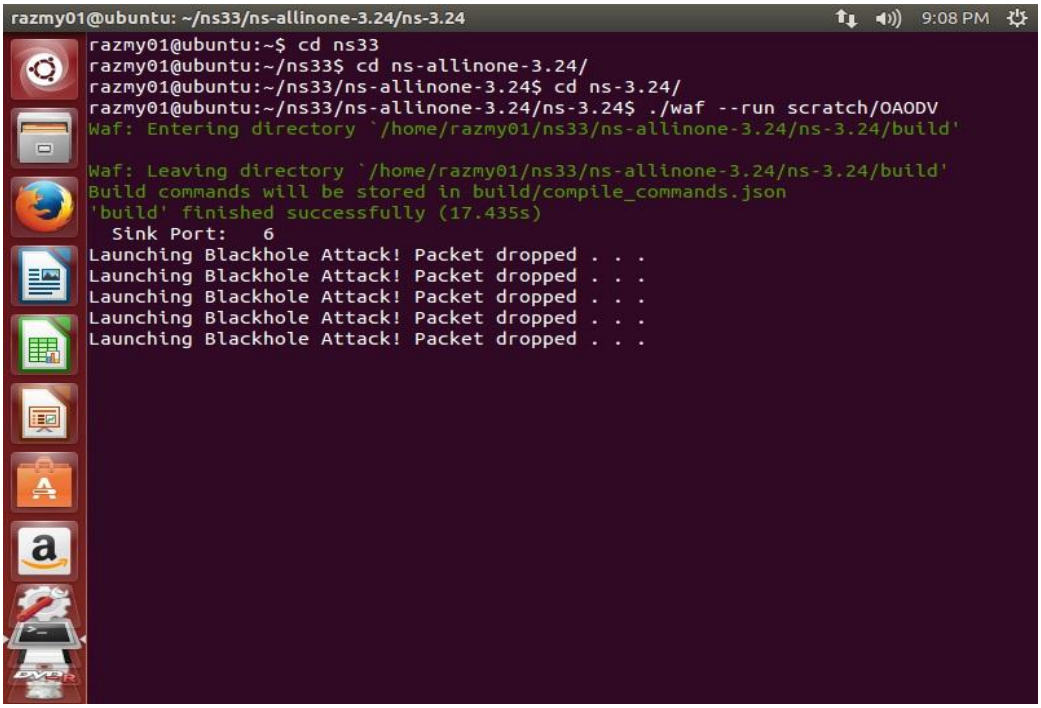


Figure 31. Output of AODV Program

If the Black hole attack is working with the pure AODV code, the Packet dropped message displayed on the screen. Also displayed the sink port and Max Packets per trace file exceed. The Sink port is a port which Receive and consumes traffic generated to an IP address and port**.** Max packets per trace file exceed message indicate about the length of the animation, NetAnim stops recording when the output file gets too big.

6.7. Overall output of AODV

In the next step of AODV it takes 8 minutes to display the Overall Output, which is shown below.

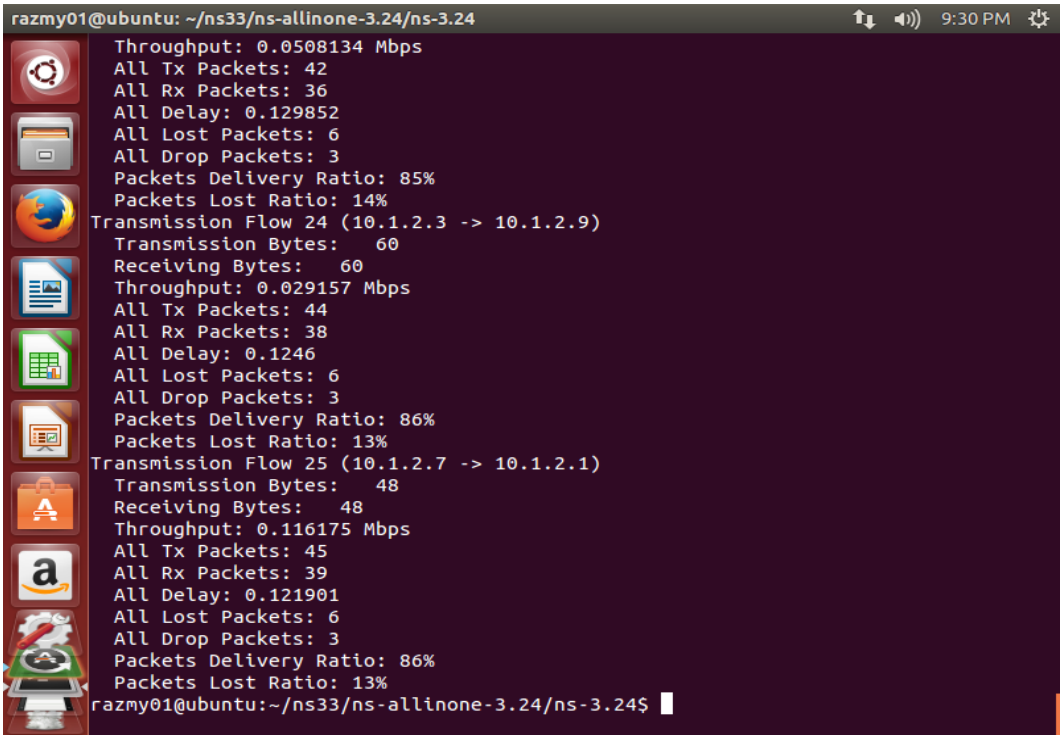


Figure 32. Overall output of OAODV Program

In the above output display Packet delivery ratio 86% as percentage and 13% displayed as Packet Lost Ratio for OAODV in Mobile Ad-hoc Network.

6.8 Compiling the Modified Ad-hoc on Demand Distance Program

The next step is running the Modified AODV file known as MDPRAODV.cc in Mobile Ad-hoc network. Which is the program file detecting the cooperative Black hole nodes in AODV protocol in Mobile Ad-hoc network. The running command of MDPRAODV.cc file is given below.

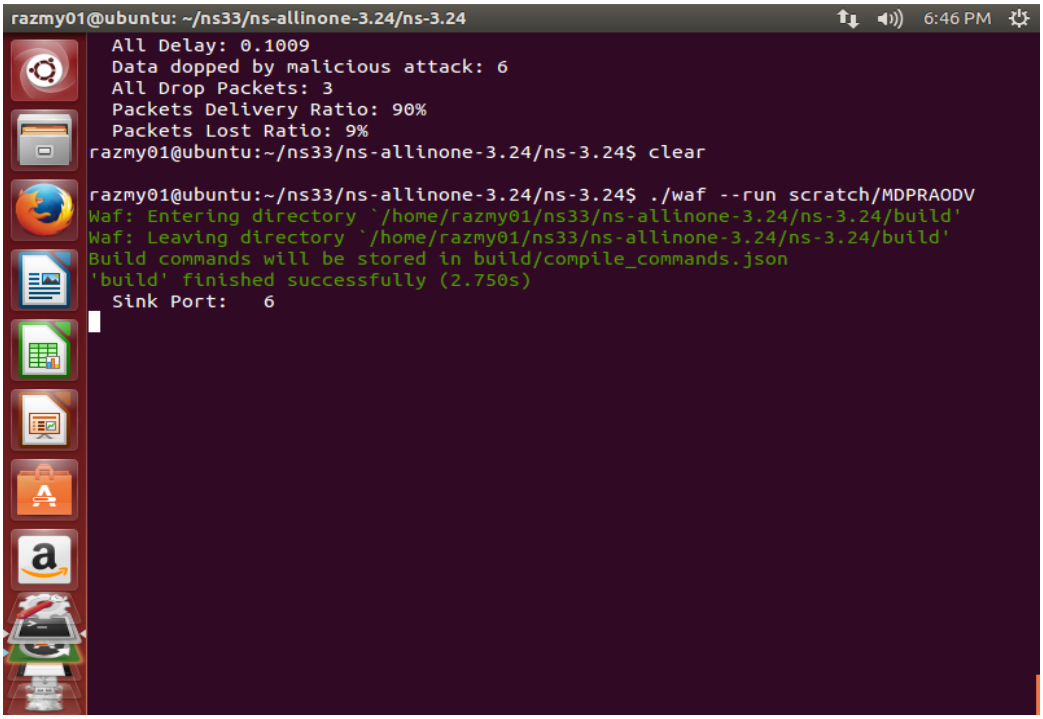


Figure 33. Compiling the Modified Ad-hoc on Demand Distance Program

6.9. Out of Modified AODV

If the Original MDPRAODV is running successfully following Message will be displayed on the screen.

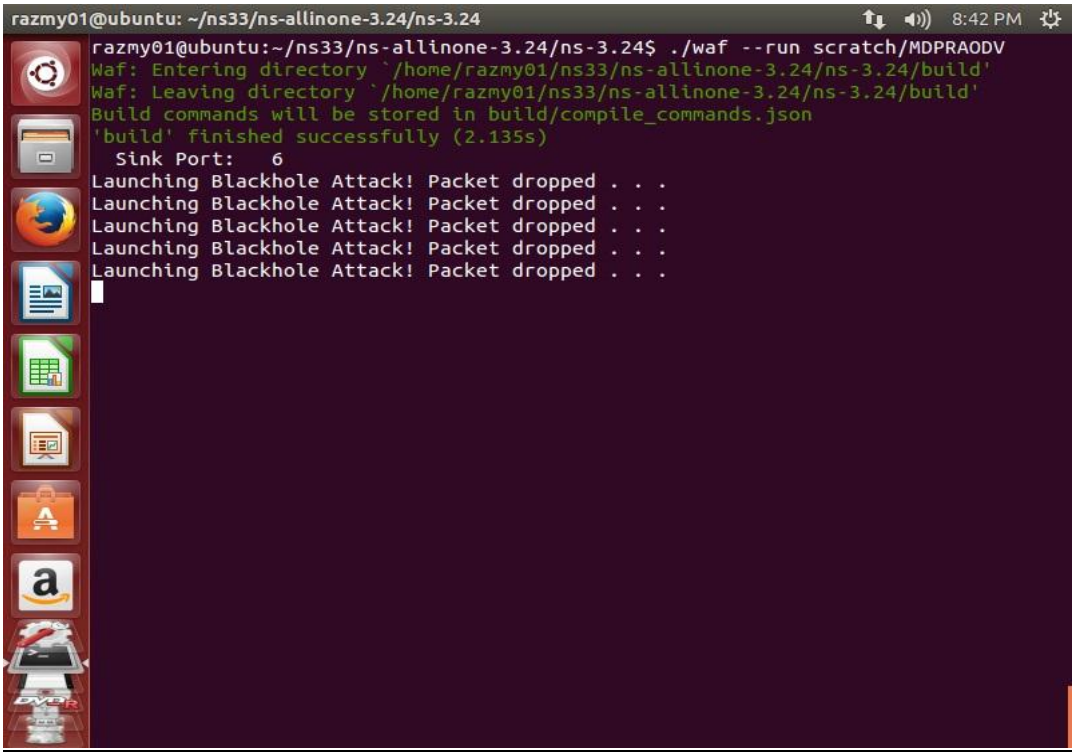


Figure 34. Output of Modified AODV

If MDPRAODV file compiled in simulation, Packet dropped message displayed on the screen. Nearly 8 minutes displayed Overall output of AODV file.

6.10. Overall output of Modified AODV

The final step, MDPRAODV takes 8 minutes to display the Overall Output, which is shown below.

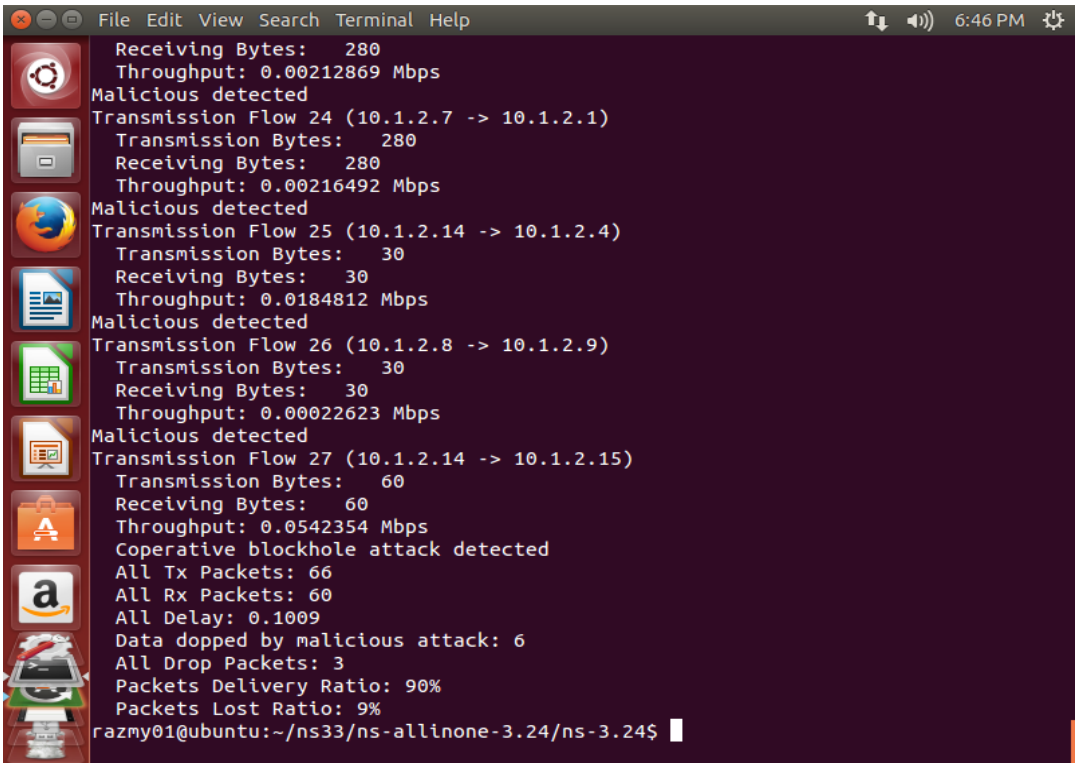


Figure 35. Overall output of Modified AODV Program

In the above output display Packet delivery ratio 90% as percentage and 9% displayed as Packet Lost Ratio for MDPRAODV in Mobile Ad-hoc Network.

6.11. Network Animator

The Network Animator is a kind of Program that is freely available in the network simulator version3. This program displays the movement of the packet through the network. The researcher has the three xml files known as Black hole, AODV and MDPRAODV. Each of the files is displayed below.

6.11.1 Blachole.xml File

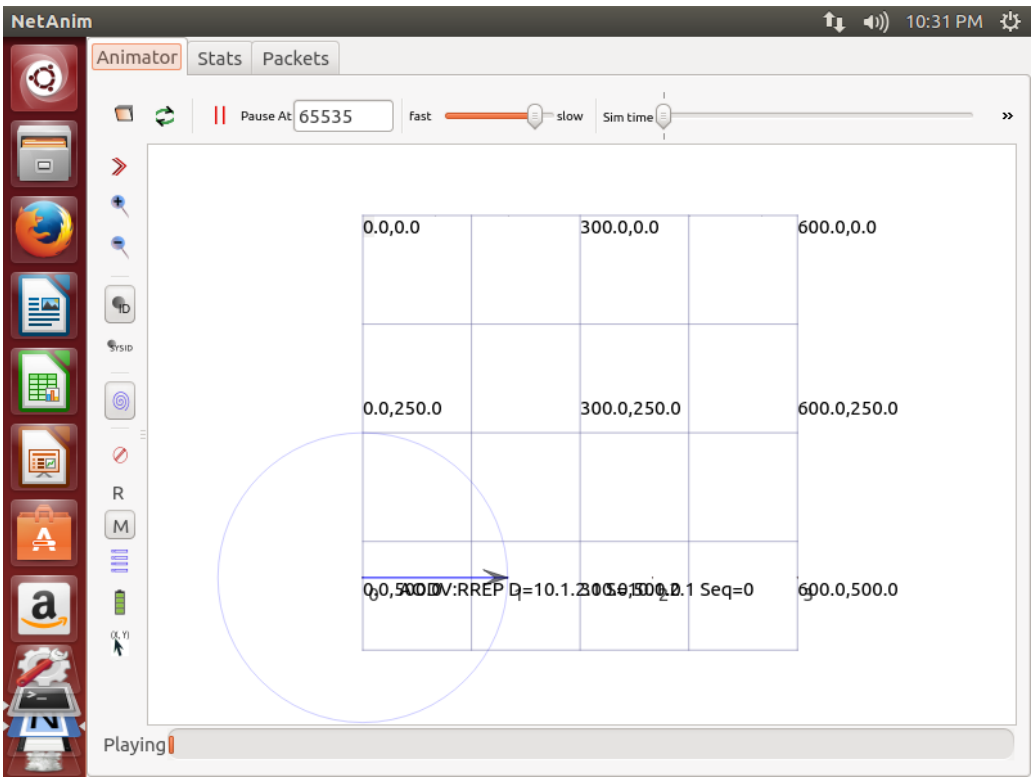


Figure 36. Blackhole.xml

The Above figure shows the movements of data packets in Black hole attack in the AODV Protocol in MANET Environment.

6.11.2 AODV.xml

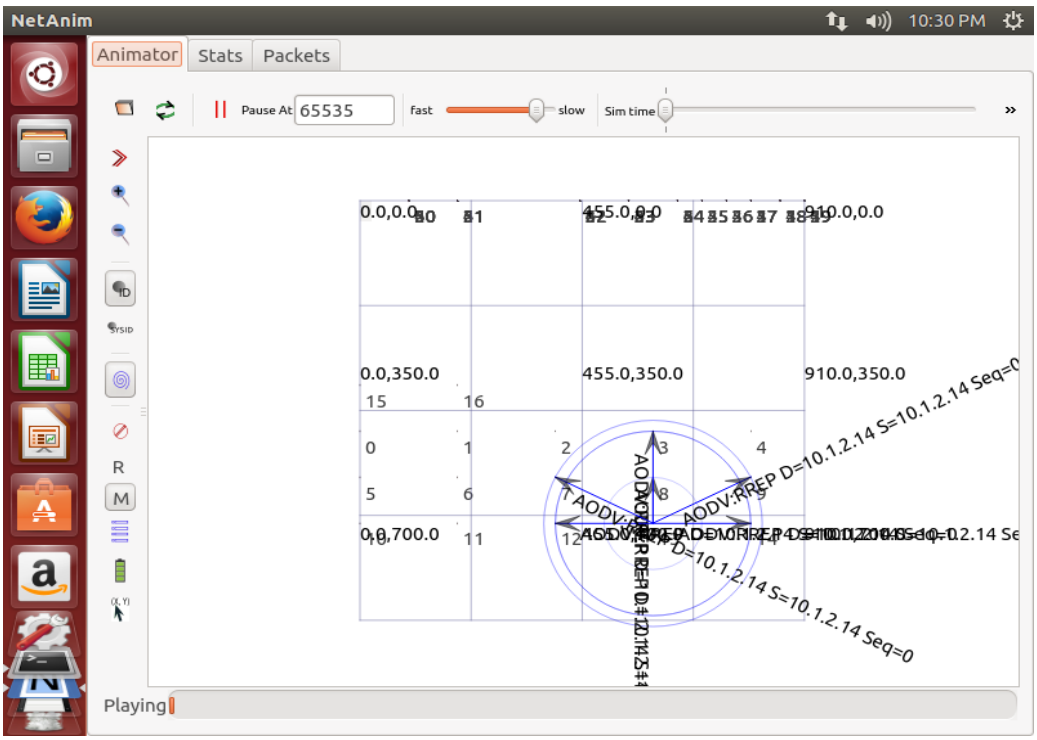


Figure 37.AODV.xml

The Above figure shows the movements of data packets in the AODV Protocol in MANET Environment. Which display the AODV Route Request and Route Response messages.

6.11.3 MDPRAODV.xml

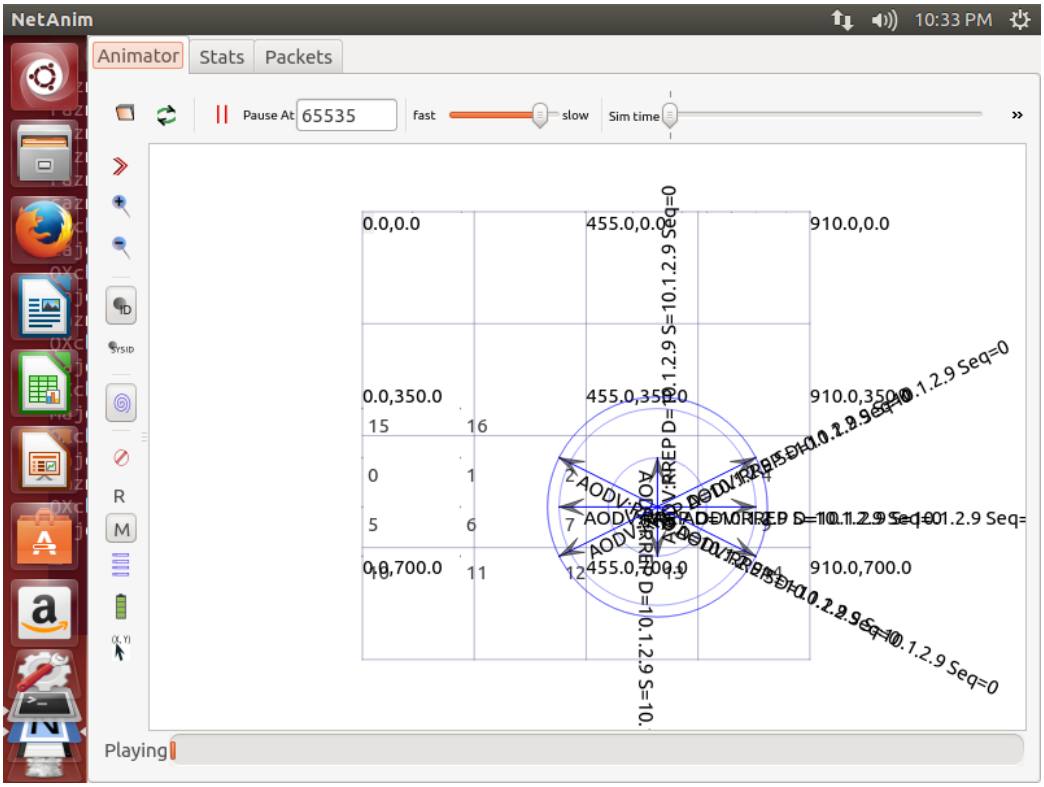


Figure 38. MDPRAODV.xml

The Above figure shows the movements of data packets in the MDPRAODV Protocol in MANET Environment. Route Request and Route Response messages also displayed in the above figure.

6.12. Summary

Chapter 06 described the Software Requirement and Implementation of the Project. For this purpose researcher compiles the Black hole code, Original AODV code and Modified AODV code. The researcher also uses the Network Animator tool to display the movement of the data packet. The movement of the black hole, AODV and MDPRAODV files, data packets are also displayed as Blackhole.xml, AODV.xml and MDPRAODV.xml. Next chapter will address the Research Result and Evaluation.

Chapter 07

Research Results and Evaluation

7.0. Introduction

Chapter 06 has stated the Software Requirement and Implementation of the Research. Chapter 07 presents the Research Results and Evaluation. For this scenario researcher has used four metrics for comparing the Original AODV Protocol with Proposed Protocol, namely Packet delivery Ratio, Packet Loss Ratio, throughput and end-to-end delay.

7.1 Packet Delivery Ratio

Packet delivery ratio describes the percentage between total number of packets received by destination and generated packets by from the source. Researcher's aim to compare the Packet delivery ratio of Present AODV Protocol and improved Modified AODV Protocol.

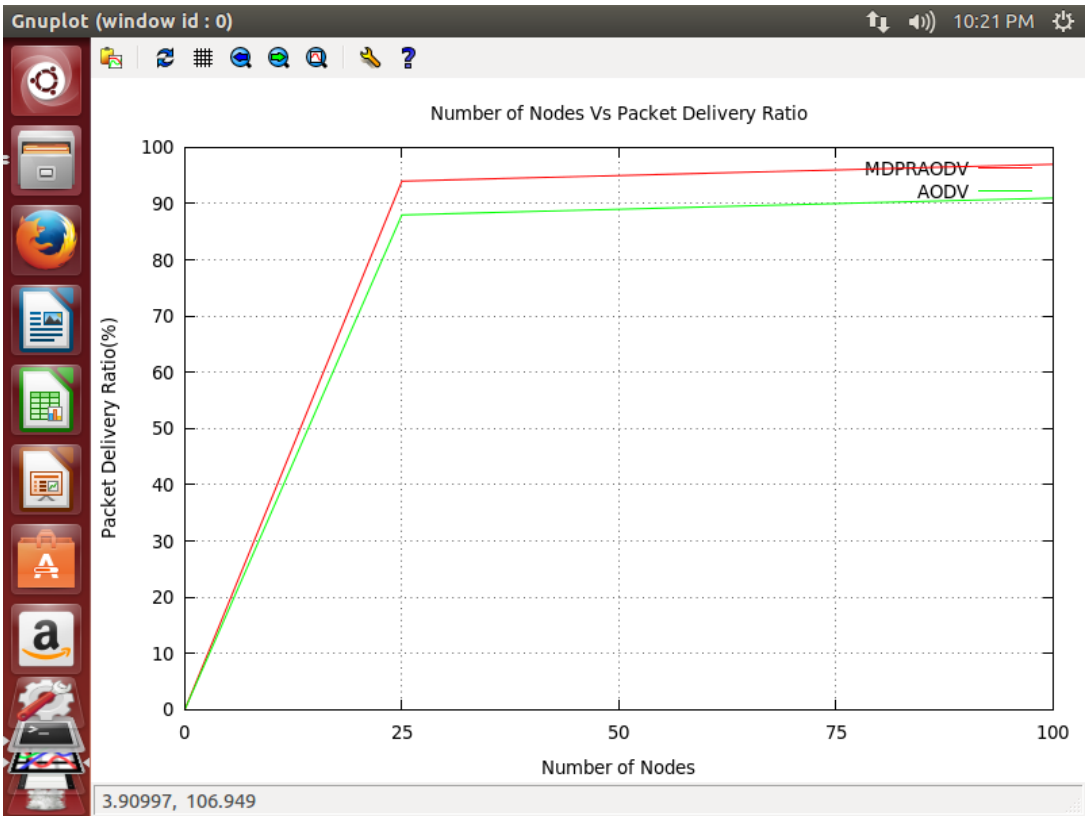


Figure 39. Packet Delivery Ratio

In the above figure shows a comparison of Packet delivery Ratio of Present AODV and Proposed Modified AODV Protocol. The Modified AODV Protocol's Packet Delivery Ratio is higher than the existing AODV Protocol.

7.2 Packet Loss Ratio

Packet loss describes the failure of more transferred packets to come to their particular destination. Packet loss ratio measured by total number of Packets loss respect to total number of packets sent.

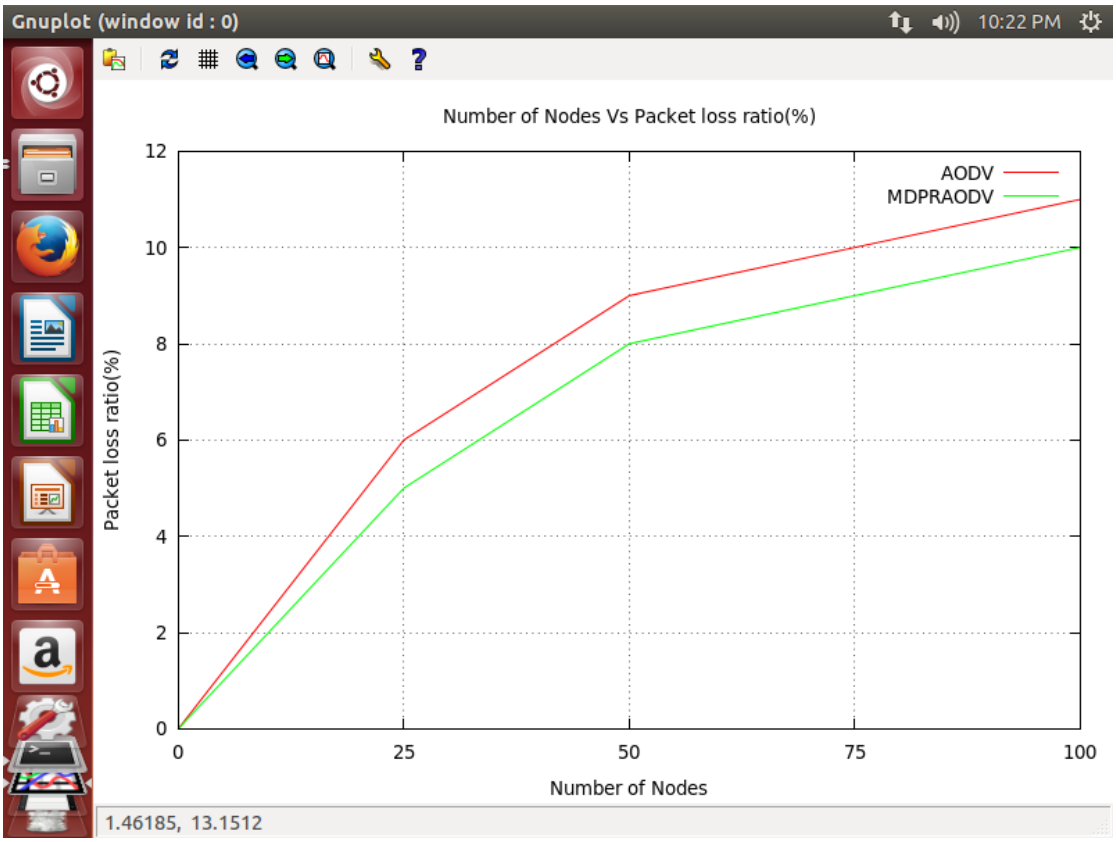


Figure 40. Packet Loss Ratio

In the above figure shows a comparison of the Packet Loss Ratio of Pure AODV and Proposed Modified AODV Protocol. The Modified AODV Protocol's Packet Loss Ratio is lower than the existing AODV Protocol.

7.3 End-to-end delay

Packet End-to-end delay describe the amount of time taken for a packet to be delivered through the network from source node to the destination node. It is a specific word used in network communication and different from round trip time, it only measures the one direction between source node to the destination node.

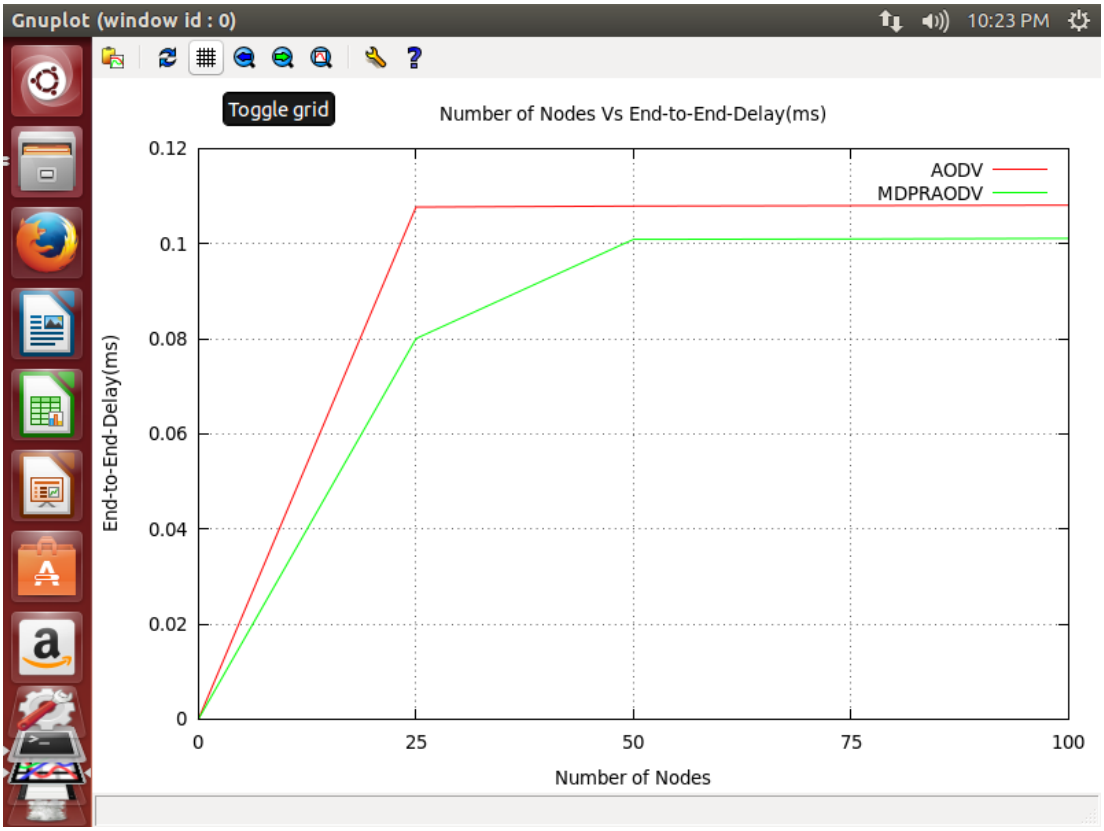


Figure 41. End to end Delay

In the above figure shows a comparison of End to end delay of Pure AODV and Proposed Modified AODV Protocol. The Modified AODV Protocol end to end delay is lower than the existing AODV Protocol.

7.4 Throughput

In Data communication, Throughput stated amount of data packets successfully transferred source node to a destination node in a specific time period and is measured in bits per second. The throughput is calculated by total number of data packets is divided by the specified time period.

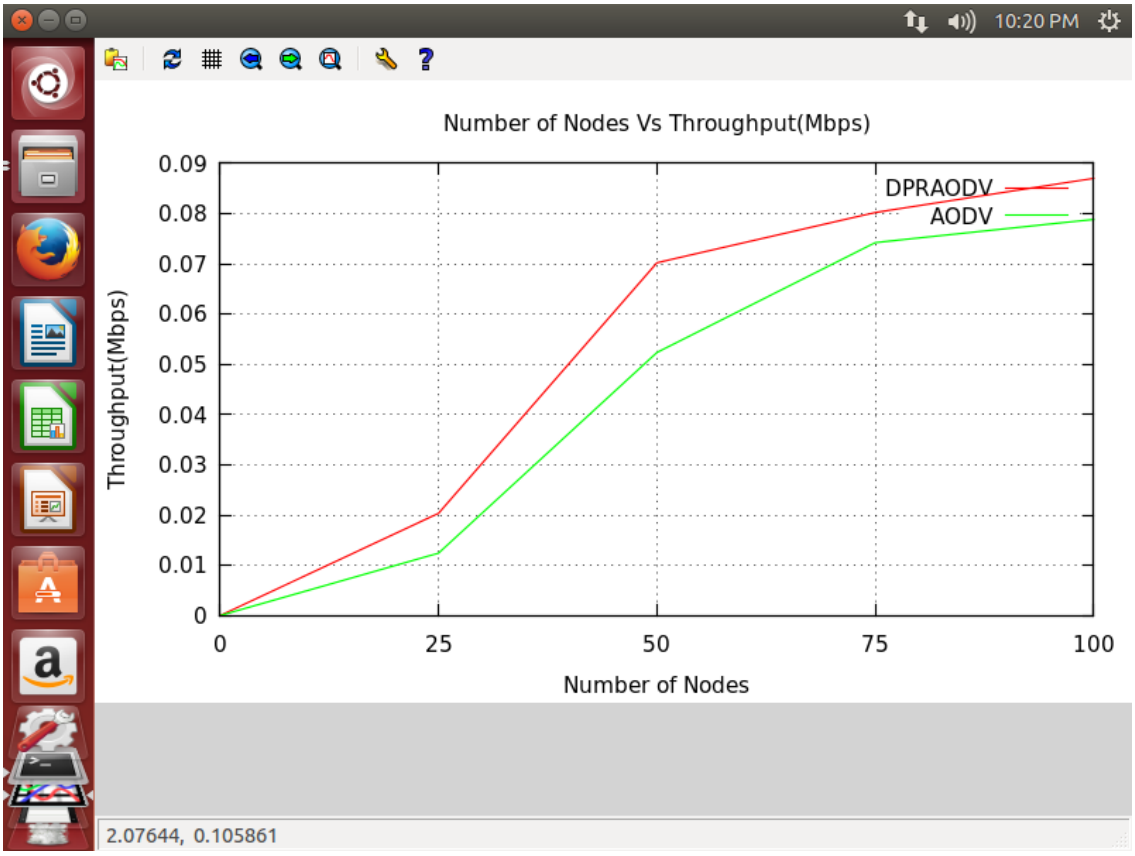


Figure 42. Throughput

In the above figure shows a comparison of network throughput of Pure AODV and Proposed Modified AODV Protocol. The Modified AODV Protocol's through is higher than the existing AODV Protocol.

7.5 Summary

This Chapter described the Research Results and Evaluation. Implementing the particular research, researcher has used four metrics namely Packet Delivery Ratio, Packet Loss Ratio, End-to-end delay and throughput. Next chapter will address Conclusion & Future Work of the dissertation.

Chapter 08

Conclusion and Future work

8.0. Introduction

Chapter 07 has sated the Research Results and Evaluation of the Thesis. The researcher has evaluated four metrics namely Packet Delivery Ratio, Packet Loss Ratio, End-to-end delay and throughput. Chapter 08 presents the Conclusion and Future work of the dissertation.

8.1. Conclusion

The cooperative Black hole attack is the one of the major challenges in the MANET. The Proposed MDPRAODV implementation, researcher detects the many numbers of cooperative Black hole nodes in the AODV Protocol in MANET and it was clear to detect many cooperative malicious nodes accurately. However, implementing the proposed solution, researcher added 100 nodes for the Original AODV code and improved AODV code. If the nodes are increased more than 100 nodes the proposed program will take more than 5 minutes to detect the malicious cooperative Black hole nodes in the Modified AODV and it clearly to display Modified AODV file has high packet delivery Ratio, high throughput and decrease the Packet Lost ratio and Delay than the Original AODV protocol in MANET. The proposed technique detects the cooperative Black nodes rather than securely secure way of transmitting data source to destination which is the main drawbacks of the proposed solution.

8.2. Future Work

Mobile Ad-Hoc networks are broadly used networks due to their flexible environment. These networks are wide-open to both external and internal attacks and there is no control centralized mechanism. There is a need to detect the cooperative Black Hole attack in other MANETs routing protocols such as DSR, TORA and GRP. Other types of attacks such as Wormhole, Jellyfish and Sybil attacks are needed to be studied in comparison with the black hole attack.

8.3 Summary

Chapter08 stated the conclusion work which has been implemented by the researcher and future work on Mobile Ad-hoc network is presented.

References

[1] C. E. Perkins Editor *et al.*, “Ad Hoc Networking Addison-Wesley,” 2008.

[2] A. Saini and H. Kumar, “Comparison between various black hole detection techniques in MANET, Akanksha Saini , Harish Kumar,” *Instrumentation*, no. pp. 19–20, 2010.

[3] A. Hinds, M. Ngulube, S. Zhu, and H. Al-Aqrabi, “A Review of Routing Protocols for Mobile Ad-Hoc NETworks (MANET),” *Int. J. Inf. Educ. Technol.*, vol. 3, no. 1, pp. 1–5, 2013.

[4] K. Majumder and S. K. Sarkar, “Performance analysis of AODV and DSR routing protocols in hybrid network scenario,” *Proc. INDICON 2009 - An IEEE India Counc. Conf.*, vol. 2, no. 2, 2009.

[5] H. Deng, W. Li, and D. P. Agrawal, “Routing security in wireless ad hoc networks,” *IEEE Commun. Mag.*, vol. 40, no. 10, pp. 70–75, 2002.

[6] P. Manickam and T. G. Baskar, “Pefromance comparioson of routing protocols in Mobile adhoc networks,” vol. 3, no. 1, pp. 98–106, 2011.

[7] Kumar Rai *et al.*, “Different Types of Attacks on Integrated MANET-Internet Communication,” no. 4, pp. 265–274, 2010.

[8] K. Pandey and A. Swaroop, “A Comprehensive Performance Analysis of Proactive, Reactive and Hybrid MANETs Routing Protocols,” *IJCSI Int. J. Comput. Sci. Issues ISSN*, vol. 8, no. 3, pp. 1694–814, 2011.

[9] N. Kalia, “Detection of Multiple Black hole nodes attack in MANET by modifying AODV protocol,” vol. 8, no. 5, pp. 160–174.

[10] N. Mistry, D. C. Jinwala, and M. Zaveri, “Improving AODV Protocol against Blackhole Attacks,” *Int. Multiconference Eng. Comput. Sci. (Imecs 2010), Vols I-Iii*, vol. II, pp. 1034–1039, 2010.

[11] P. N. Raj and P. B. Swadas, “Dpraodv: a Dyanamic Learning System Against,” *Int. J. Comput. Sci. Issues*, vol. 2, pp. 54–59, 2009.

[12] A. K. Sahoo, “Prevention of Black hole Attack in AODV protocols for Mobile Ad Hoc Network by Key Authentication Prevention of Black hole Attack in AODV protocols for Mobile Ad Hoc Network by Key Authentication,” 2012, 2014.

[13] S. L. Dhende, “A-98. A Mechanism for Detection of Black Hole Attack in Mobile Ad Hoc Networks,” vol. 1, no. 6, pp. 1–4, 2012.

[14] L. Tamilselvan and V. Sankaranarayanan, “Prevention of co-operative black hole attack in MANET,” *J. Networks*, vol. 3, no. 5, pp. 13–20, 2008.

[15] L. Himral, V. Vig, and N. Chand, “Preventing AODV Routing Protocol from Black Hole Attack,” *Int. J. Eng. Sci. Technol.*, vol. 3, no. 5, pp. 3927–3932, 2011.

[16] W. Saetang and S. Charoenpanyasak, “CAODV Free Blackhole Attack in Ad Hoc Networks,” vol. 35, no. Cncs, pp. 63–68, 2012.

[17] S. Univercity, “Security attacks I . INTRODUCTION,” vol. 2, no. 3, pp. 651–657, 2012.

[18] K. Vishnu and A. J. Paul, “Detection and Removal of Cooperative Black/Gray hole attack in Mobile AdHoc Networks,” *Int. J. Comput. Appl.*, vol. 1, no. 22, pp. 40–44, 2010.

[19] P. Goyal, V. Parmar, and R. Rishi, “MANET: Vulnerabilities, Challenges, Attacks, Application,” *IJCEM Int. J. Comput. Eng. Manag. ISSN*, vol, pp. 2230–7893, 2011.

[20] M. Chitkara and M. W. Ahmad, “Review on MANET: Characteristics, Challenges, Imperatives and Routing Protocols,” *Int. J. Comput. Sci. Mob. Comput.*, vol. 32, no. 2, pp. 432–437, 2014.

[21] S. A. K. A. Omari and P. Sumari, “An Overview of Mobile Ad Hoc Networks for the Existing Protocols and Applications,” *Int. J. Appl. Graph Theory Wirel. Ad Hoc Networks Sens. Networks*, vol. 2, no. 1, pp. 87–110, 2010.

[22] S. Ade and P. Tijare, “Performance comparison of AODV, DSDV, OLSR and DSR routing protocols in mobile ad hoc networks,” *Int. J. Inf. Technol. Knowl. Manag.*, vol. 2, no. 2, pp. 545–548, 2010.

[23] A. S. Roy, M. Borah, and A. Banerjee, “Study of Fisheye Routing Protocol in Ns3 and Its Comparative Analysis With Aodv ,” pp. 82–94.

[24] K. A. Adoni, “Optimization of Energy Consumption for OLSR Routing Protocol in MANET,” *Int. J. Wirel. Mob. Networks*, vol. 4, no. 1, pp. 251–262, 2012.

[25] V. D. Park and M. S. Corson, “A performance comparison of the temporally-ordered routing algorithm and ideal link-state routing,” *Proc. - 3rd IEEE Symp. Comput. Commun. ISCC 1998*, pp. 592–598, 1998.

[26] N. Beijar, “Zone Routing Protocol (ZRP),” *Netw. Lab. Helsinki Univ. Technol. Finl. 9*, pp. 1–12, 2002.

[27] A. Huhtonen, “Comparing AODV and OLSR Routing Protocols 2 Ad hoc On

Demand Distance Vector," *Telecommun. Softw. Multimed.*, pp. 1–9, 2004.

[28] I. D. Chakeres and E. M. Belding-Royer, "AODV routing protocol implementation design," pp. 698–703, 2004.

[29] P. M. Jawandhiya, M. M. Ghonge, M. S. Ali, and P. J. S. Deshpande, "A Survey of Mobile Ad Hoc Network Attacks," *Int. J. Eng. Sci. Technol.*, vol. 2, no. 9, pp. 4063–4071, 2010.

[30] K. I. Lakhtaria, "Next generation wireless network security and privacy," *Next Gener. Wirel. Netw. Secur. Priv.*, 2015.

[31] S. Alzahrani and L. Hong, "Generation of DDoS Attack Dataset for Effective IDS Development and Evaluation," *J. Inf. Secur.*, vol. 09, no. 04, pp. 225–241, 2018.

[32] S. Dixit, P. Pathak, and S. Gupta, "A novel approch for gray hole and black hole detection and prevention," *2016 Symp. Colossal Data Anal. Networking, CDAN 2016*.

[33] A. Nadeem and M. Howarth, "Adaptive intrusion detection & prevention of denial of service attacks in MANETs," p. 926, 2009.

[34] H. Bakiler and A. Şafak, "Analysis of Current Routing Attacks in Mobile Ad Hoc Networks," *Int. J. Appl. Math. Electron. Comput.*, vol. 3, no. 2, p. 127, 2015.

[35] R. H. Jhaveri, A. D. Patel, J. D. Parmar, and B. I. Shah, "MANET routing protocols and wormhole attack against AODV," *Int. J. Comput. Sci. Netw. Secur.*, vol. 10, no. 4, pp. 12–18, 2010.

[36] J. Kumar, M. Kulkarni, and D. Gupta, "Effect of Black Hole Attack on MANET Routing Protocols," *Int. J. Comput. Netw. Inf. Secur.*, vol. 5, no. 5, pp. 64–72, 2013.

[37] V. Shanmuganathan and M. T. Anand, "A Survey on Gray Hole Attack in MANET," *IRACST-International J. Comput. Networks Wirel. Commun.*, vol. 2, pp. 647–650, 2012.

[38] P. Casas, J. Mazel, and P. Owezarski, "Knowledge-independent traffic monitoring: Unsupervised detection of network attacks," *IEEE Netw.*, vol. 26, no. 1, pp. 13–21, 2012.

[39] S. Yadav, "Attacks in MANET," vol. 1, no. 3, pp. 123–126, 2012.

[40] M. P. A. Patil, M. E. H. V. P. M. Coet, and A. P. H. V. P. M. Coet, "Available Online at www.ijarcs.info Network Traffic Monitoring with IDS," vol. 4, no. 6, pp. 214–219, 2013.

APPENDIX-A

Glossary of Terms

- Node: A node in a mobile ad hoc network is the actual device that communicates with other devices using wireless transmission. It refers host or hop as well.
- Data Packets: A Data Packet is a small unit of data transmits through the network.
- DOS: A Denial-of-service is a kind of attack where the intruder will block the authorized users to using the available resources. In this attack, an intruder or attacker will send unnecessary messages to the network which include the invalid addresses.
- Protocol: A Protocol is a set of rules and procedure for transferring data over the network.
- Black hole attack: A Black hole attack is kind of Denial-of-service attack in which a router that is supposed to spread packets instead of rejects them.
- Cooperative Black hole: Cooperative Black hole is kind of black hole attack where the attacker nodes worked as a unit or group.
- Network Simulator: in Networking Research, network simulation is software that forecast or analyze of a computer network.
- RREQ: Route Request Message, the function of Route Request message is creating a route to a destination when the nodes need to transmit a data.
- RREP: Route Reply message, the destination node sends the reply messages to the source node that there is a valid path.
- Route Error: Route Error message, when there is a link failure in the network the Route Error message is generated and informed it to node in the network.
- Security Attack: it is unauthorized access to services cause to damage or modify the network.
- Virtual Private Network: it is a kind of network, which is a secure tunnel two or more devices on the network.

- Throughput: in Data communication throughput is the amount of data transferred successfully from source node to destination. Throughput is measured in bits per second.
- Packet Delivery Ratio: Packet delivery ratio is the percentage of packets successfully expected to total sent. Packet delivery Ratio Calculated by total number of data packets received is divided by data packets produced by each source.
- Packet Loss Ratio: Data Packets Loss arises when data packets transferring over the computer network unable to reach their destination. It is measured as a Percentage of Packets lost with respect to total data packets.
- End-to-end delay: End to end delay is the amount of time taken for a particular data packet transmitted from source node to the destination node.

APPENDIX-B

Installation guide line for Network Simulator Version 3.24

1. Introduction NS 3.24

Network Simulator Version 3 is the famous tool designed for Wireless communication. It is the successor of Network Simulator Version 2. The tool is written in C++ and python language. And also Network Simulator 3 is mainly developed on Linux platforms.

There are many versions available for Network Simulator 3. Research has interesting to use network simulator version 3.24 version it includes many modules. Implementing the Research researcher discuss two modules namely,

i. Network Simulator
ii. Network Animator

1.1. Installing the network simulator 3.24 on Ubuntu 14.04

It is not very easy to install the network simulator into the Windows Platform. Therefore the user has to install the VMwhere workstation to the windows. After that install the Linux version. Implementing the researcher, researcher installed the Ubuntu 14.04 Version. If the Ubuntu installed successfully user has to install the network simulator version 3.24 for the Ubuntu.

1.2. Using NS3

1. Network simulator version 3 has many dependencies. Therefore, before the ns3 installation researcher needs to install those dependencies. Open the new terminal and user has to type the following commands in command line.

$] sudo apt update
$] sudo apt upgrade

2. After that Researcher needs the run the combination of following command, it will take some time to complete the installation

$] sudo apt-get install build-essential autoconf automake libxmu-dev python-pygoocanvas python-pygraphviz cvs mercurial bzr git cmake p7zip-full

python-matplotlib python-tk python-dev python-kiwi python-gnome2 python-gnome2-desktop-dev python-rsvg qt4-dev-tools qt4-qmake qt4-qmake qt4-default gnuplot-x11 wireshark

3. If the above commands run successfully, then Researcher has to install the Network simulator 3.24. The researcher needs to open the terminal and type the following commands listed below,

   $] tar jxvf ns-allione-3.24.tar.bz2

   $] cd ns-allinone-3.24/

   $] ./build.py --enable-examples --enable-tests

   Above commands take some time to complete. If the installation is successful, the following researcher will get the following screen on the terminal

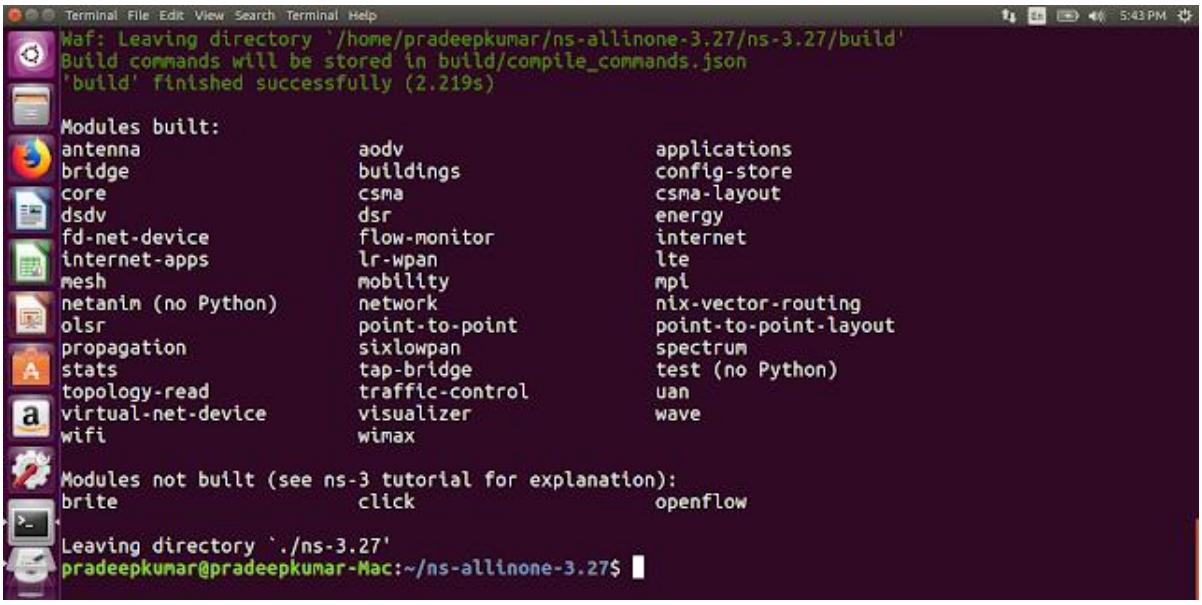


Figure 43. NS3 installation

APPENDIX-C

Flow Charts:

C1. Route Discovery of AODV Protocol Flow Chart:

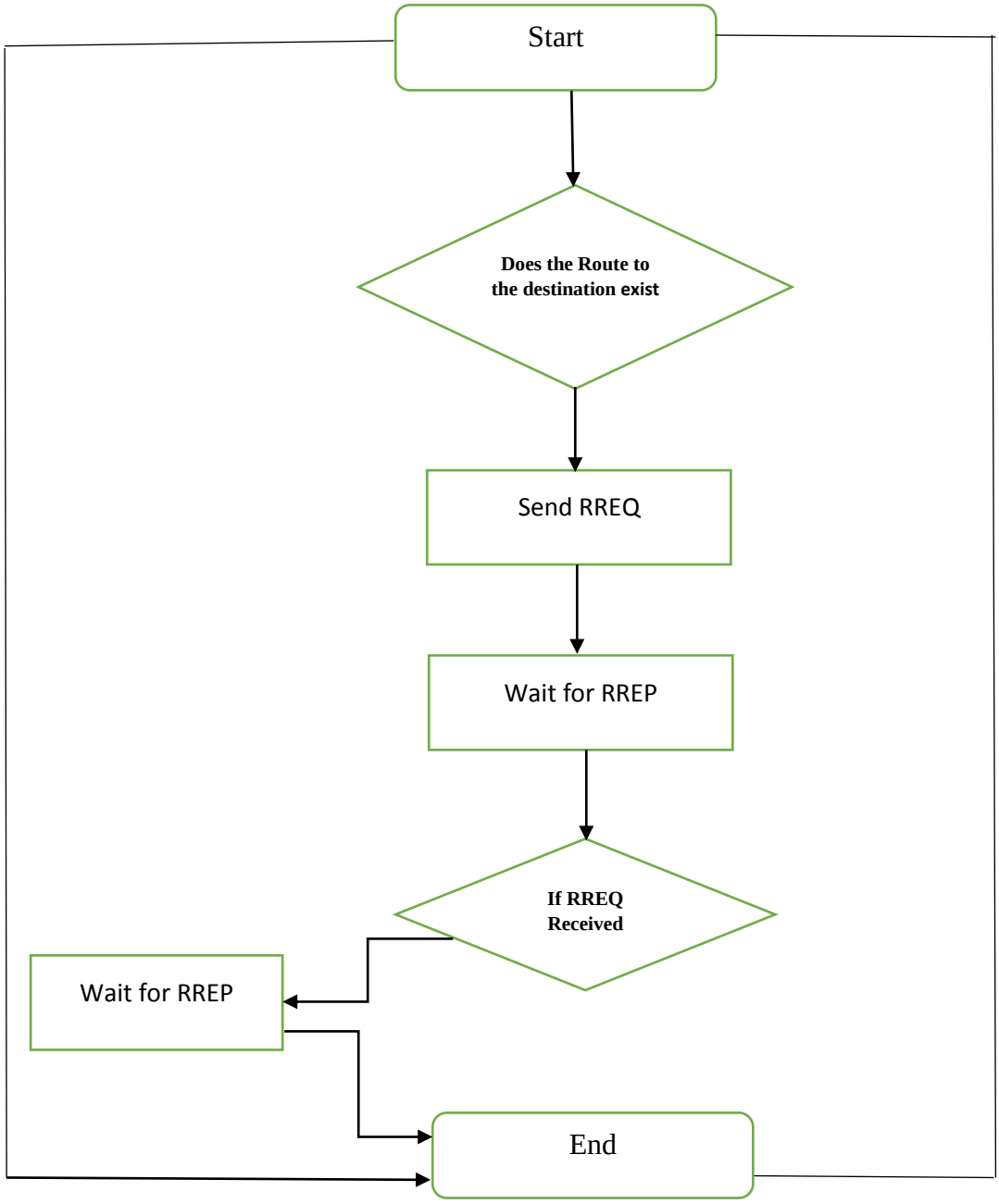


Figure 44. Route Discovery of AODV

C2. Route Reply of AODV Protocol Flow Chart:

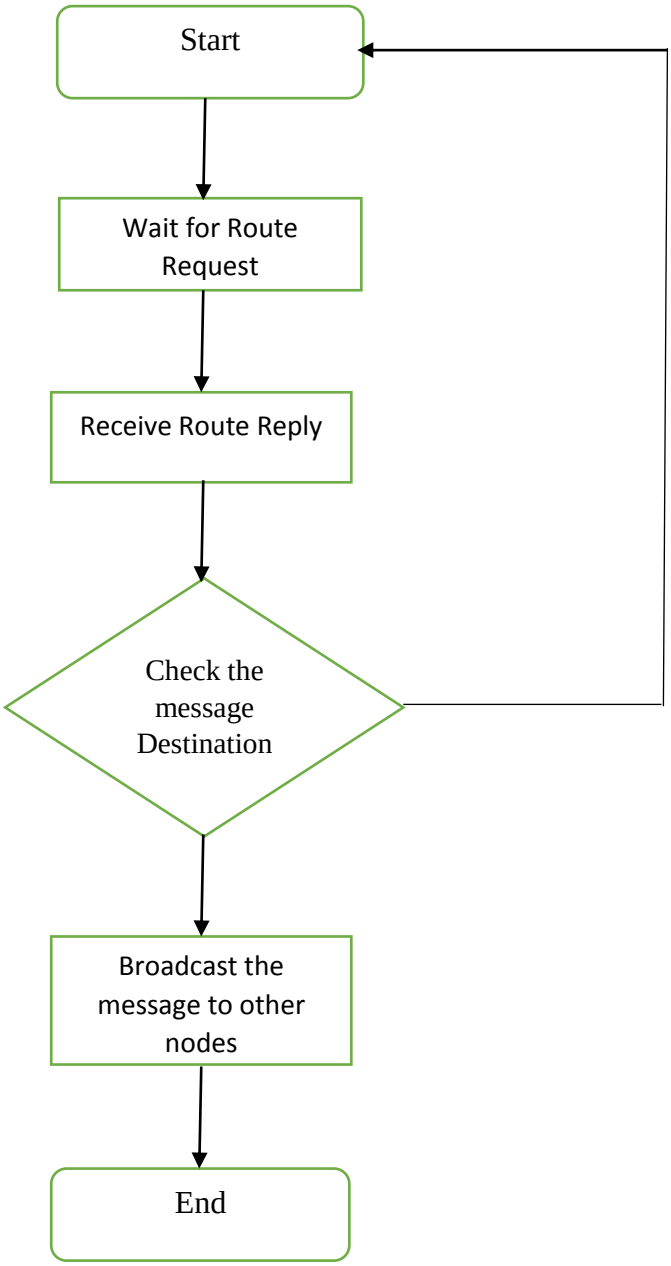


Figure 45. Route Reply of AODV

C3. AODV Protocol Flow Chart

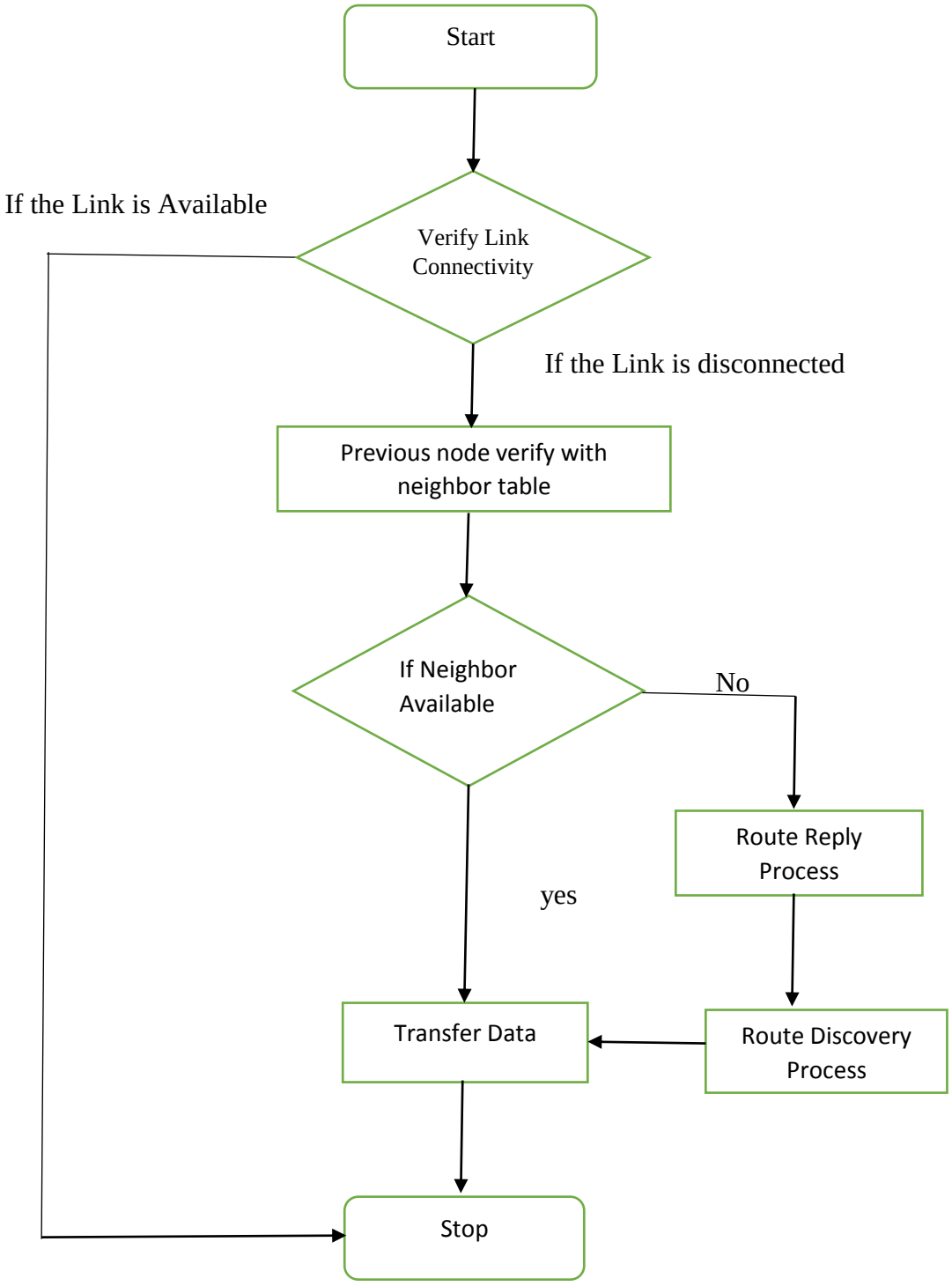


Figure 46. AODV Protocol Flow Chart

C4. Black hole attack Flow Chart

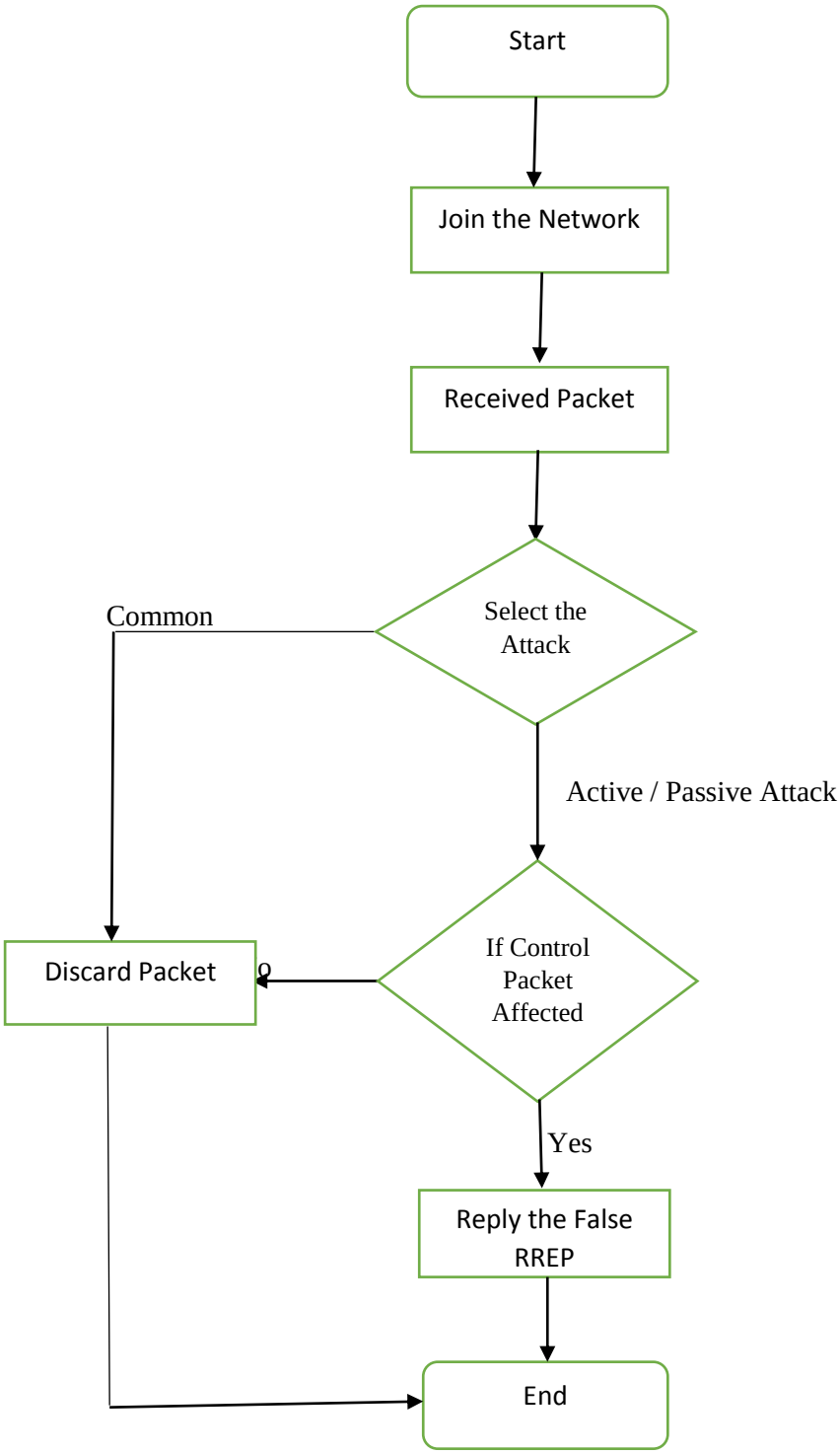


Figure 47. Black hole attack Flow Chart

C5. Cooperative Black hole Detection Flow Chart

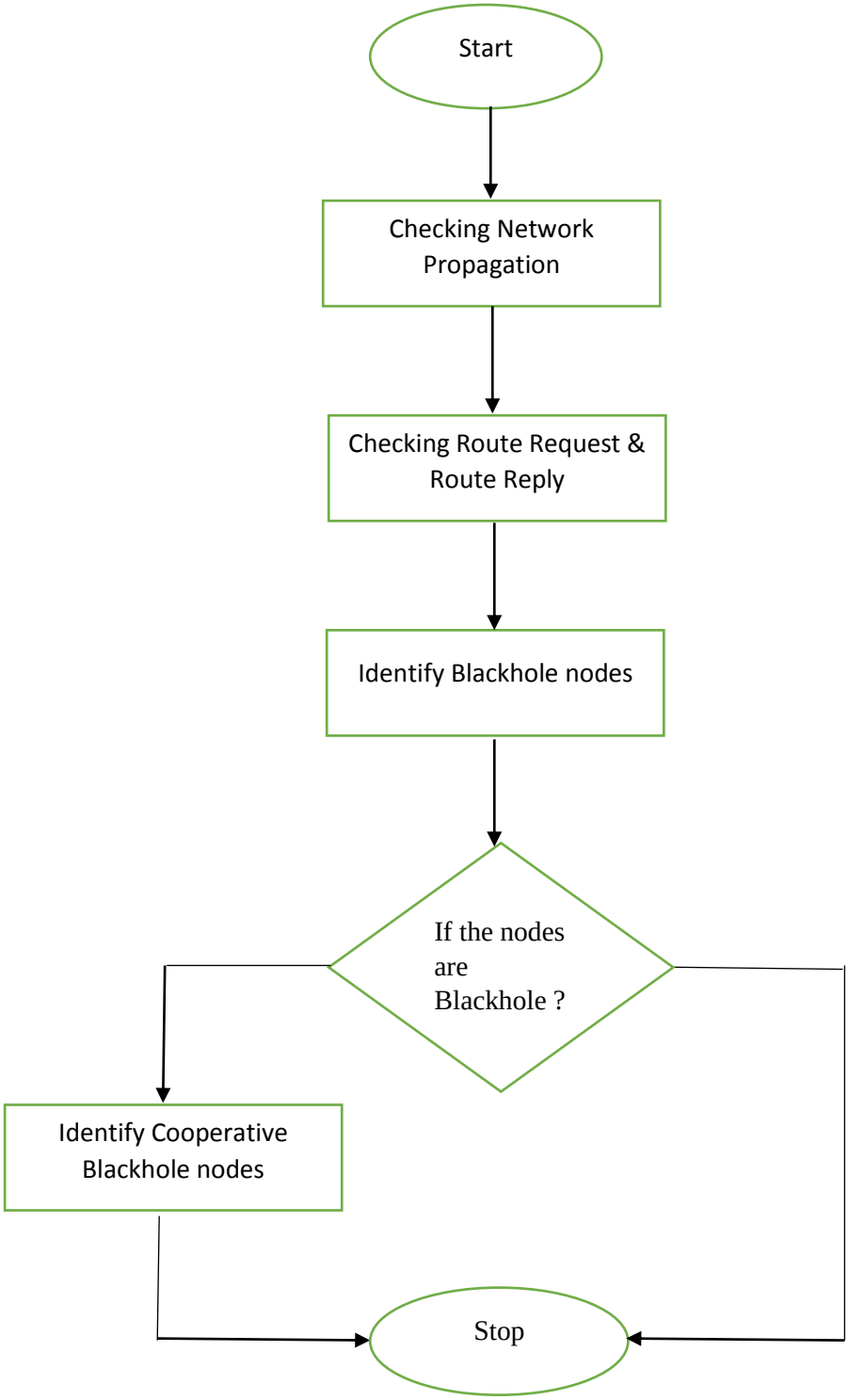


Figure 48.Cooperative Black hole Detection Flow Chart

C6. Flow Diagram of Implementation of the Proposed Method in Network Simulator Version 3

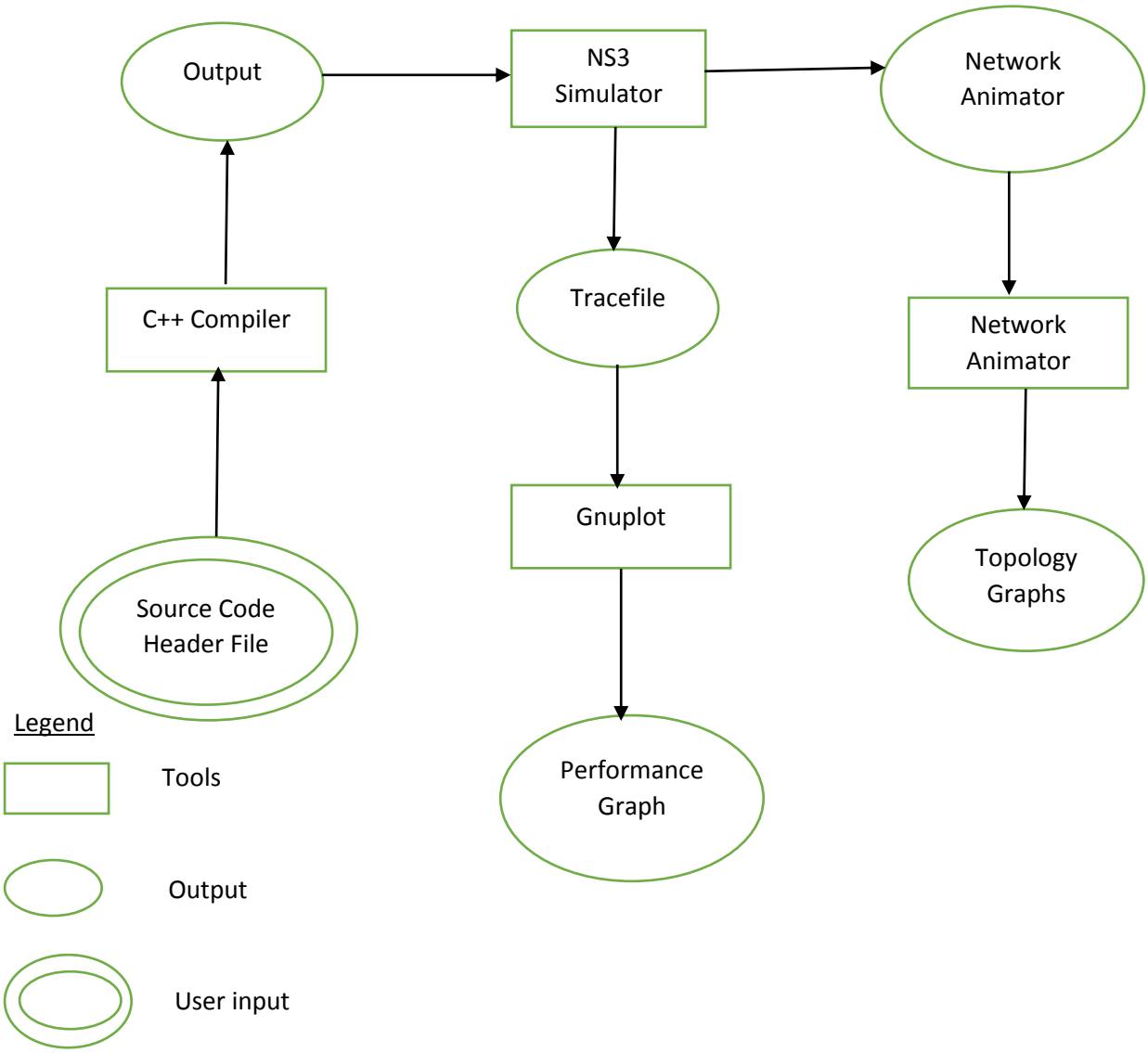


Figure 49. Overall Diagram of Proposed Method

Appendix D- UML Diagram

D1. AODV Message Parsing

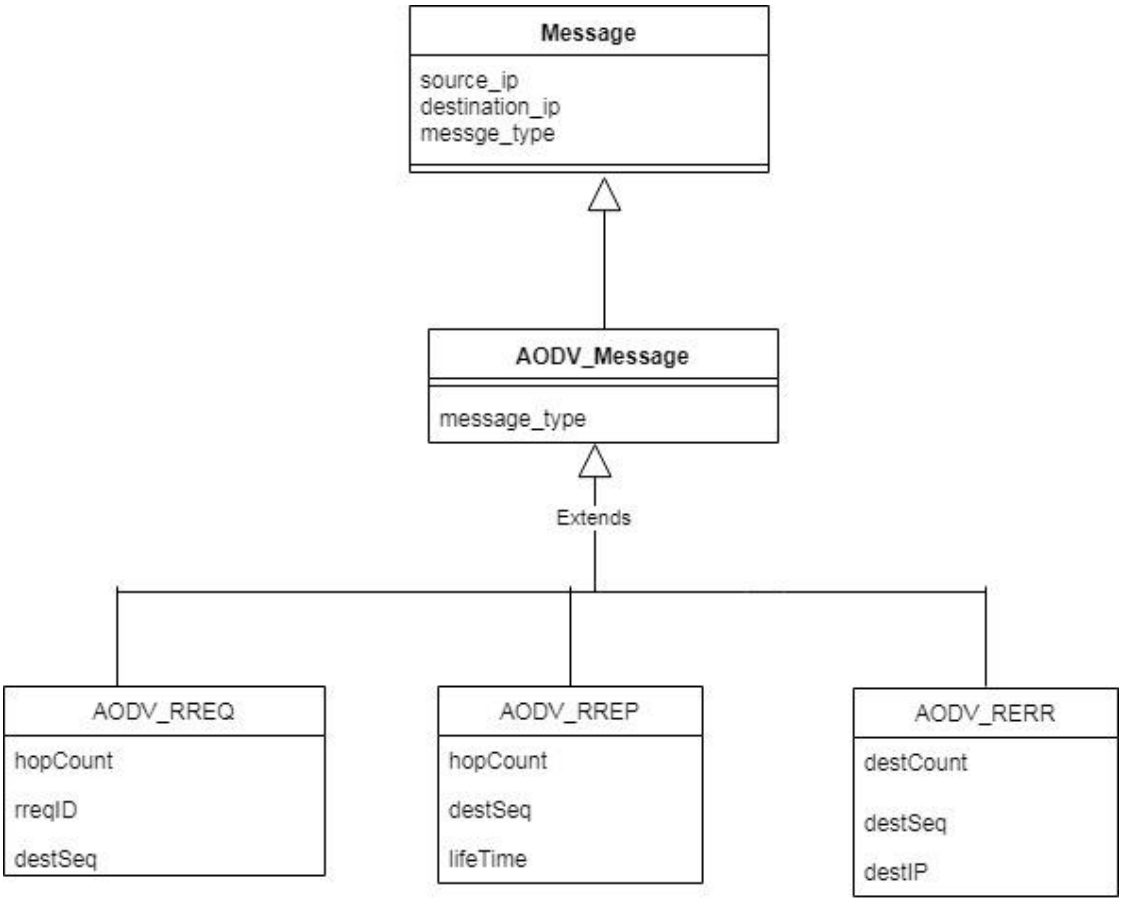


Figure 50. AODV Message parsing

D2. AODV Protocol Class Diagram

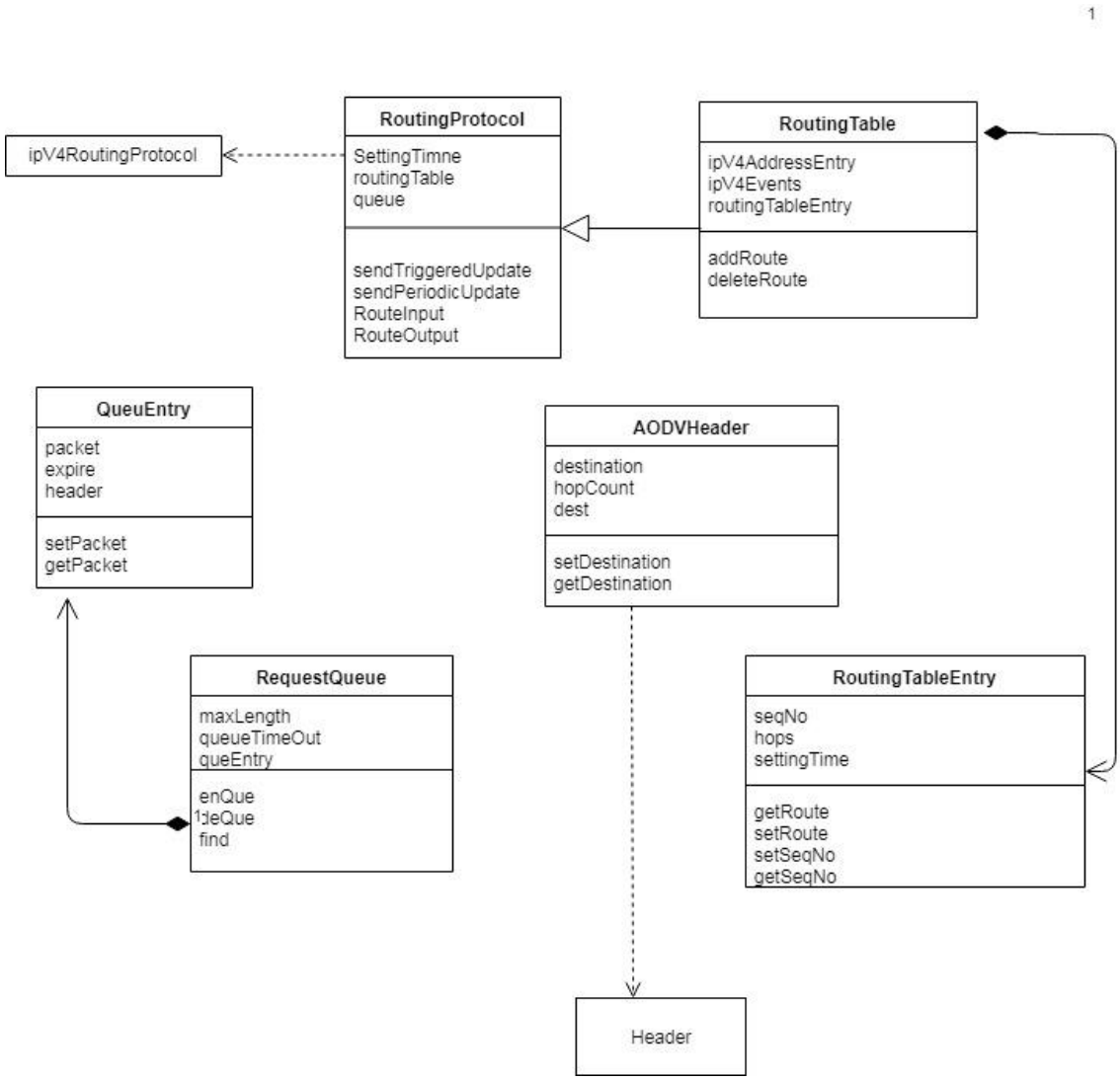


Figure 51. AODV Class Diagram

D3. AODV Protocol Message Passing Sequence Diagram

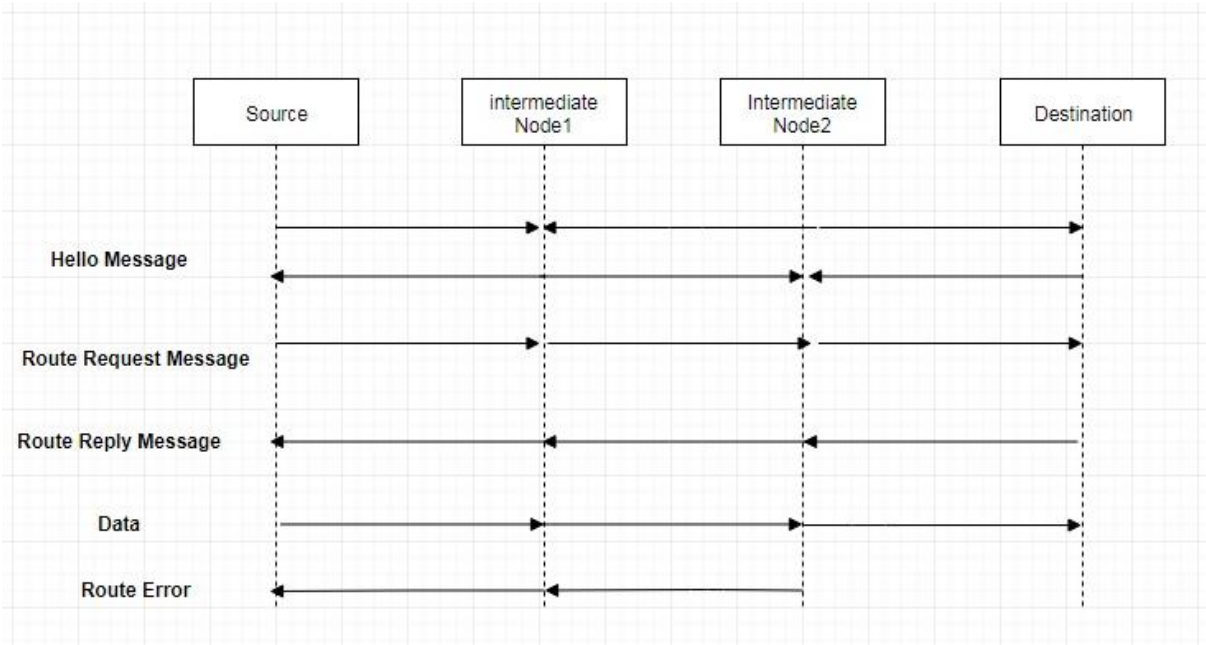


Figure 52. AODV Protocol Message Parsing

D4. AODV Message Passing for Route Discovery

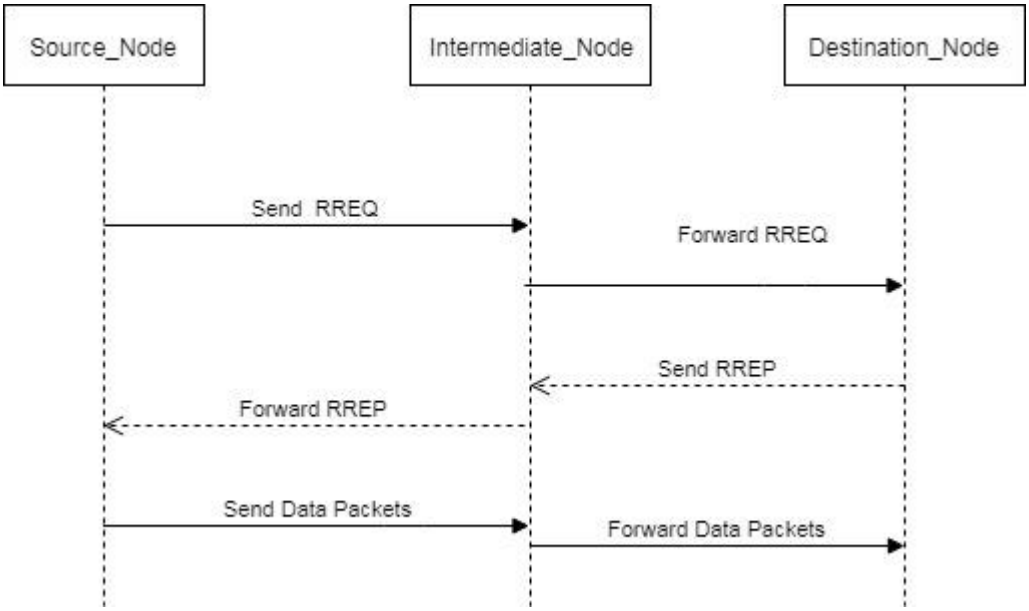


Figure 53. AODV Message Passing for Route Discovery

Appendix E

Source Code

Source codes which are written in C++ Language implemented in Network simulator 3.24 for the tasks which are listed in this appendix.

E1.The Complete Source code for the Block-hole attack source code known as blackhole.cc

```
/* Blackhole Attack Simulation with AODV Routing Protocol - Sample
Program
 *
 * Network topology
 *
 *    n0 ------------> n1 ------------> n2 -------------> n3
 *
 * Each node is in the range of its immediate adjacent.
 * Source Node: n1
 * Destination Node: n3
 * Malicious Node: n0
 *
 * Output of this file:
 * 1. Generates blackhole.routes file for routing table information and
 * 2. blackhole.xml file for viewing animation in NetAnim.
 *
 */
#include "ns3/aodv-module.h"
#include "ns3/netanim-module.h"
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/applications-module.h"
#include "ns3/mobility-module.h"
#include "ns3/wifi-module.h"
```

```
#include "ns3/netanim-module.h"
#include "ns3/flow-monitor-module.h"
#include "ns3/mobility-module.h"
#include "myapp.h"
NS_LOG_COMPONENT_DEFINE ("Blackhole");
using namespace ns3;
void
ReceivePacket(Ptr<const Packet> p, const Address & addr)
{
        std::cout << Simulator::Now ().GetSeconds () << "\t" << p-
>GetSize() <<"\n";
}

int main (int argc, char *argv[])
{
  bool enableFlowMonitor = false;
  std::string phyMode ("DsssRate1Mbps");
  CommandLine cmd;
  cmd.AddValue ("EnableMonitor", "Enable Flow Monitor",
enableFlowMonitor);
  cmd.AddValue ("phyMode", "Wifi Phy mode", phyMode);
  cmd.Parse (argc, argv);
// Explicitly create the nodes required by the topology (shown above).
  NS_LOG_INFO ("Create nodes.");
  NodeContainer c; // ALL Nodes
  NodeContainer not_malicious;
  NodeContainer malicious;
  c.Create(4);
 not_malicious.Add(c.Get(1));
  not_malicious.Add(c.Get(2));
  not_malicious.Add(c.Get(3));
  malicious.Add(c.Get(0));
  // Set up WiFi
  WifiHelper wifi;
```

```
  YansWifiPhyHelper wifiPhy =  YansWifiPhyHelper::Default ();
  wifiPhy.SetPcapDataLinkType
(YansWifiPhyHelper::DLT_IEEE802_11);
  YansWifiChannelHelper wifiChannel ;
  wifiChannel.SetPropagationDelay
("ns3::ConstantSpeedPropagationDelayModel");
  wifiChannel.AddPropagationLoss
("ns3::TwoRayGroundPropagationLossModel",
  "SystemLoss", DoubleValue(1),
  "HeightAboveZ", DoubleValue(1.5));
  // For range near 250m
  wifiPhy.Set ("TxPowerStart", DoubleValue(33));
  wifiPhy.Set ("TxPowerEnd", DoubleValue(33));
  wifiPhy.Set ("TxPowerLevels", UintegerValue(1));
  wifiPhy.Set ("TxGain", DoubleValue(0));
  wifiPhy.Set ("RxGain", DoubleValue(0));
  wifiPhy.Set ("EnergyDetectionThreshold", DoubleValue(-61.8));
  wifiPhy.Set ("CcaMode1Threshold", DoubleValue(-64.8));
  wifiPhy.SetChannel (wifiChannel.Create ());
  // Add a non-QoS upper mac
  NqosWifiMacHelper wifiMac = NqosWifiMacHelper::Default ();
  wifiMac.SetType ("ns3::AdhocWifiMac");


  // Set 802.11b standard
  wifi.SetStandard (WIFI_PHY_STANDARD_80211b);
  wifi.SetRemoteStationManager ("ns3::ConstantRateWifiManager",
                        "DataMode",StringValue(phyMode),
                        "ControlMode",StringValue(phyMode));
  NetDeviceContainer devices;
  devices = wifi.Install (wifiPhy, wifiMac, c);
//  Enable AODV
```

```
  AodvHelper aodv;
  AodvHelper malicious_aodv;
   // Set up internet stack
  InternetStackHelper internet;
  internet.SetRoutingHelper (aodv);
  internet.Install (not_malicious);
    malicious_aodv.Set("IsMalicious",BooleanValue(true)); // putting
*false* instead of*true* would disable the malicious behavior of the
node
internet.SetRoutingHelper (malicious_aodv);
  internet.Install (malicious);
  // Set up Addresses
  Ipv4AddressHelper ipv4;
  NS_LOG_INFO ("Assign IP Addresses.");
  ipv4.SetBase ("10.1.2.0", "255.255.255.0");
  Ipv4InterfaceContainer ifcont = ipv4.Assign (devices);
  NS_LOG_INFO ("Create Applications.");
  // UDP connection from N1 to N3
  uint16_t sinkPort = 6;
  Address sinkAddress (InetSocketAddress (ifcont.GetAddress (3),
sinkPort)); // interfaceof n3
PacketSinkHelper packetSinkHelper ("ns3::UdpSocketFactory",
InetSocketAddress (Ipv4Address::GetAny (), sinkPort));
  ApplicationContainer sinkApps = packetSinkHelper.Install (c.Get (3));
//n3 as sink
  sinkApps.Start (Seconds (0.));
  sinkApps.Stop (Seconds (100.));
  Ptr<Socket> ns3UdpSocket = Socket::CreateSocket (c.Get (1),
UdpSocketFactory::GetTypeId ()); //source at n1
  // Create UDP application at n1
  Ptr<MyApp> app = CreateObject<MyApp> ();
```

```
  app->Setup (ns3UdpSocket, sinkAddress, 1040, 5, DataRate
("250Kbps"));
  c.Get (1)->AddApplication (app);
  app->SetStartTime (Seconds (40.));
  app->SetStopTime (Seconds (100.));
// Set Mobility for all nodes

  MobilityHelper mobility;
  Ptr<ListPositionAllocator> positionAlloc = CreateObject
<ListPositionAllocator>();
  positionAlloc ->Add(Vector(100, 0, 0)); // node0
  positionAlloc ->Add(Vector(200, 0, 0)); // node1
  positionAlloc ->Add(Vector(450, 0, 0)); // node2
  positionAlloc ->Add(Vector(550, 0, 0)); // node3
  mobility.SetPositionAllocator(positionAlloc);
  mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
  mobility.Install(c);
  AnimationInterface anim ("blackhole.xml"); // Mandatory
  AnimationInterface::SetConstantPosition (c.Get (0), 0, 500);
  AnimationInterface::SetConstantPosition (c.Get (1), 200, 500);
  AnimationInterface::SetConstantPosition (c.Get (2), 400, 500);
  AnimationInterface::SetConstantPosition (c.Get (3), 600, 500);
  anim.EnablePacketMetadata(true);
     Ptr<OutputStreamWrapper> routingStream =
Create<OutputStreamWrapper> ("blackhole.routes", std::ios::out);
     aodv.PrintRoutingTableAllAt (Seconds (45), routingStream);
  // Trace Received Packets
Config::ConnectWithoutContext("/NodeList/*/ApplicationList/*/$ns3::
PacketSink/Rx",MakeCallback (&ReceivePacket));
//
// Calculate Throughput using Flowmonitor
//
  FlowMonitorHelper flowmon;
  Ptr<FlowMonitor> monitor = flowmon.InstallAll();
```

```
//// Now, do the actual simulation.
  NS_LOG_INFO ("Run Simulation.");
  Simulator::Stop (Seconds(100.0));
  Simulator::Run ();
  monitor->CheckForLostPackets ();
  Ptr<Ipv4FlowClassifier> classifier =
DynamicCast<Ipv4FlowClassifier> (flowmon.GetClassifier ());
  std::map<FlowId, FlowMonitor::FlowStats> stats = monitor-
>GetFlowStats ();
  for (std::map<FlowId, FlowMonitor::FlowStats>::const_iterator i =
stats.begin (); i != stats.end (); ++i)
    {
          Ipv4FlowClassifier::FiveTuple t = classifier->FindFlow (i-
>first);
     if ((t.sourceAddress=="10.1.2.2" && t.destinationAddress ==
"10.1.2.4"))
     {
        std::cout << "Flow " << i->first  << " (" << t.sourceAddress << " -
> " <<t.destinationAddress << ")\n";
    std::cout << "  Tx Bytes:   " << i->second.txBytes << "\n";
        std::cout << "  Rx Bytes:   " << i->second.rxBytes << "\n";
          std::cout << "  Throughput: " << i->second.rxBytes * 8.0 / (i-
>second.timeLastRxPacket.GetSeconds() - i-
>second.timeFirstTxPacket.GetSeconds())/1024/1024  << " Mbps\n";
     }
    }
  monitor->SerializeToXmlFile("lab-4.flowmon", true, true);
}
```

E2.The Complete code for myApp.h header file known as myApp.h

```
#include "ns3/applications-module.h"
#include "ns3/core-module.h"
```

```
using namespace ns3;
class MyApp : public Application
{
public:
MyApp ();
virtual ~MyApp();
void Setup (Ptr<Socket> socket, Address address, uint32_t packetSize,
uint32_t nPackets, DataRate dataRate);
private:
virtual void StartApplication (void);
virtual void StopApplication (void);
 void ScheduleTx (void);
 void SendPacket (void);
Ptr<Socket>     m_socket;
Address         m_peer;
uint32_t        m_packetSize;
uint32_t        m_nPackets;
DataRate        m_dataRate;
EventId         m_sendEvent;
bool            m_running;
uint32_t        m_packetsSent;
};
MyApp::MyApp ()
: m_socket (0),
 m_peer (),
  m_packetSize (0),
   m_nPackets (0),
 m_dataRate (0),
m_sendEvent (),
m_running (false),
m_packetsSent (0)
{
```

```
}
MyApp::~MyApp()
{
m_socket = 0;
}
void
MyApp::Setup (Ptr<Socket> socket, Address address, uint32_t
packetSize, uint32_t nPackets, DataRate dataRate)
{
m_socket = socket;
m_peer = address;
m_packetSize = packetSize;
m_nPackets = nPackets;
  m_dataRate = dataRate;
void
MyApp::StartApplication (void)
{
m_running = true;
m_packetsSent = 0;
m_socket->Bind ();
m_socket->Connect (m_peer);
SendPacket ();
}
Void MyApp::StopApplication (void)
{
m_running = false;
if (m_sendEvent.IsRunning ())
{
Simulator::Cancel (m_sendEvent);
    }
 if (m_socket)
   {
     m_socket->Close ();
  }
}
```

```
void
MyApp::SendPacket (void)
{
 Ptr<Packet> packet = Create<Packet> (m_packetSize);
  m_socket->Send (packet);
  if (++m_packetsSent < m_nPackets)
  {
  ScheduleTx ();
  }
}
void
MyApp::ScheduleTx (void)
{
  if (m_running)
{
     Time tNext (Seconds (m_packetSize * 8 / static_cast<double>
(m_dataRate.GetBitRate ())));
 m_sendEvent = Simulator::Schedule (tNext, &MyApp::SendPacket,
this);
  }
}
```

E3.The complete Source code for the OAODV protocol source code known as OAODV.cc

```
    #include "ns3/aodv-module.h"
    #include "ns3/netanim-module.h"
    #include "ns3/core-module.h"
    #include "ns3/network-module.h"
    #include "ns3/internet-module.h"
    #include "ns3/applications-module.h"
    #include "ns3/mobility-module.h"
```

```
#include "ns3/wifi-module.h"
#include "ns3/netanim-module.h"
#include "ns3/flow-monitor-module.h"
#include "ns3/mobility-module.h"
#include "myapp.h"
NS_LOG_COMPONENT_DEFINE ("Blackhole");
using namespace ns3;
void
ReceivePacket(Ptr<const Packet> p, const Address & addr)
{
    std::cout << Simulator::Now ().GetSeconds () << "\t" << p-
>GetSize() <<"\n";
}
int main (int argc, char *argv[])
{
  bool enableFlowMonitor = false;
  std::string phyMode ("DsssRate1Mbps");
  CommandLine cmd;
  cmd.AddValue ("EnableMonitor", "Enable Flow Monitor",
 enableFlowMonitor);
cmd.AddValue ("phyMode", "Wifi Phy mode", phyMode);
cmd.Parse (argc, argv);
    NS_LOG_INFO ("Create nodes.");
    NodeContainer c; // ALL Nodes
    NodeContainer not_malicious;
      NodeContainer malicious;
    c.Create(100);
     not_malicious.Add(c.Get(1));
    not_malicious.Add(c.Get(2));
    not_malicious.Add(c.Get(3));
    not_malicious.Add(c.Get(4));
    not_malicious.Add(c.Get(5));
```

```
not_malicious.Add(c.Get(6));
not_malicious.Add(c.Get(7));
not_malicious.Add(c.Get(8));
not_malicious.Add(c.Get(9));
not_malicious.Add(c.Get(10));
not_malicious.Add(c.Get(11));
not_malicious.Add(c.Get(12));
not_malicious.Add(c.Get(13));
not_malicious.Add(c.Get(14));
not_malicious.Add(c.Get(15));
not_malicious.Add(c.Get(16));
not_malicious.Add(c.Get(17));
not_malicious.Add(c.Get(18));
not_malicious.Add(c.Get(19));
not_malicious.Add(c.Get(20));
not_malicious.Add(c.Get(21));
not_malicious.Add(c.Get(22));
not_malicious.Add(c.Get(23));
not_malicious.Add(c.Get(24));
not_malicious.Add(c.Get(25));
not_malicious.Add(c.Get(26));
not_malicious.Add(c.Get(27));
not_malicious.Add(c.Get(28));
not_malicious.Add(c.Get(29));
not_malicious.Add(c.Get(30));
not_malicious.Add(c.Get(31));
not_malicious.Add(c.Get(32));
not_malicious.Add(c.Get(33));
not_malicious.Add(c.Get(34));
not_malicious.Add(c.Get(35));
not_malicious.Add(c.Get(36));
not_malicious.Add(c.Get(37));
```

```
not_malicious.Add(c.Get(38));
not_malicious.Add(c.Get(39));
not_malicious.Add(c.Get(40));
not_malicious.Add(c.Get(41));
not_malicious.Add(c.Get(42));
not_malicious.Add(c.Get(43));
not_malicious.Add(c.Get(44));
not_malicious.Add(c.Get(45));
not_malicious.Add(c.Get(46));
not_malicious.Add(c.Get(47));
not_malicious.Add(c.Get(48));
not_malicious.Add(c.Get(49));
not_malicious.Add(c.Get(50));
not_malicious.Add(c.Get(51));
not_malicious.Add(c.Get(52));
not_malicious.Add(c.Get(53));
not_malicious.Add(c.Get(54));
not_malicious.Add(c.Get(55));
not_malicious.Add(c.Get(56));
not_malicious.Add(c.Get(57));
not_malicious.Add(c.Get(58));
not_malicious.Add(c.Get(59));
not_malicious.Add(c.Get(60));
not_malicious.Add(c.Get(61));
not_malicious.Add(c.Get(62));
not_malicious.Add(c.Get(63));
not_malicious.Add(c.Get(64));
not_malicious.Add(c.Get(65));
not_malicious.Add(c.Get(66));
not_malicious.Add(c.Get(67));
not_malicious.Add(c.Get(68));
not_malicious.Add(c.Get(69));
```

```
not_malicious.Add(c.Get(70));
not_malicious.Add(c.Get(71));
not_malicious.Add(c.Get(72));
not_malicious.Add(c.Get(73));
not_malicious.Add(c.Get(74));
not_malicious.Add(c.Get(75));
not_malicious.Add(c.Get(76));
not_malicious.Add(c.Get(77));
not_malicious.Add(c.Get(78));
not_malicious.Add(c.Get(79));
not_malicious.Add(c.Get(80));
not_malicious.Add(c.Get(81));
not_malicious.Add(c.Get(82));
not_malicious.Add(c.Get(83));
not_malicious.Add(c.Get(84));
not_malicious.Add(c.Get(85));
not_malicious.Add(c.Get(86));
not_malicious.Add(c.Get(87));
not_malicious.Add(c.Get(88));
not_malicious.Add(c.Get(89));
not_malicious.Add(c.Get(90));
not_malicious.Add(c.Get(91));
not_malicious.Add(c.Get(92));
not_malicious.Add(c.Get(93));
not_malicious.Add(c.Get(94));
not_malicious.Add(c.Get(95));
not_malicious.Add(c.Get(96));
not_malicious.Add(c.Get(97));
not_malicious.Add(c.Get(98));
not_malicious.Add(c.Get(99));
 malicious.Add(c.Get(0));
 // Set up WiFi
```

```
  WifiHelper wifi;
  YansWifiPhyHelper wifiPhy =  YansWifiPhyHelper::Default ();
wifiPhy.SetPcapDataLinkType(YansWifiPhyHelper::DLT_IEEE802
_11)
  YansWifiChannelHelper wifiChannel ;
  wifiChannel.SetPropagationDelay
("ns3::ConstantSpeedPropagationDelayModel");
  wifiChannel.AddPropagationLoss
("ns3::TwoRayGroundPropagationLossModel",
   "SystemLoss", DoubleValue(1),
       "HeightAboveZ", DoubleValue(1.5));
  // For range near 250m
  wifiPhy.Set ("TxPowerStart", DoubleValue(33));
  wifiPhy.Set ("TxPowerEnd", DoubleValue(33));
  wifiPhy.Set ("TxPowerLevels", UintegerValue(1));
  wifiPhy.Set ("TxGain", DoubleValue(0));
  wifiPhy.Set ("RxGain", DoubleValue(0));
  wifiPhy.Set ("EnergyDetectionThreshold", DoubleValue(-61.8));
  wifiPhy.Set ("CcaMode1Threshold", DoubleValue(-64.8));
  wifiPhy.SetChannel (wifiChannel.Create ());
  // Add a non-QoS upper mac
  NqosWifiMacHelper wifiMac = NqosWifiMacHelper::Default ();
  wifiMac.SetType ("ns3::AdhocWifiMac");
  // Set 802.11b standard
wifi.SetStandard (WIFI_PHY_STANDARD_80211b);
  wifi.SetRemoteStationManager ("ns3::ConstantRateWifiManager",
                        "DataMode",StringValue(phyMode),
                         "ControlMode",StringValue(phyMode));
  NetDeviceContainer devices;
  devices = wifi.Install (wifiPhy, wifiMac, c);
//  Enable AODV
  AodvHelper aodv;
```

```
  AodvHelper malicious_aodv;
  // Set up internet stack
  InternetStackHelper internet;
  internet.SetRoutingHelper (aodv);
  internet.Install (not_malicious);
  malicious_aodv.Set("IsMalicious",BooleanValue(true)); // putting
*false* instead of *true* would disable the malicious behavior of the
node
  internet.SetRoutingHelper (malicious_aodv);
  internet.Install (malicious);
  // Set up Addresses
  Ipv4AddressHelper ipv4;
  NS_LOG_INFO ("Assign IP Addresses.");
  ipv4.SetBase ("10.1.2.0", "255.255.255.0");
  Ipv4InterfaceContainer ifcont = ipv4.Assign (devices);
  NS_LOG_INFO ("Create Applications.");
  // UDP connection from N1 to N3
  uint16_t sinkPort = 6;
 std::cout << "  Sink Port:   " << sinkPort << "\n";
  Address sinkAddress (InetSocketAddress (ifcont.GetAddress (4),
sinkPort)); // interface of n3
  PacketSinkHelper packetSinkHelper ("ns3::UdpSocketFactory",
InetSocketAddress (Ipv4Address::GetAny (), sinkPort));
  ApplicationContainer sinkApps = packetSinkHelper.Install (c.Get
(4)); //n3 as sink
  sinkApps.Start (Seconds (0.));
  sinkApps.Stop (Seconds (100.));
  Ptr<Socket> ns3UdpSocket = Socket::CreateSocket (c.Get (1),
UdpSocketFactory::GetTypeId ()); //source at n1
  Ptr<MyApp> app = CreateObject<MyApp> ();
  app->Setup (ns3UdpSocket, sinkAddress, 1040, 5, DataRate
("250Kbps"));
```

```
  c.Get (1)->AddApplication (app);
  app->SetStartTime (Seconds (40.));
  app->SetStopTime (Seconds (100.));
// Set Mobility for all nodes
  MobilityHelper mobility;
  Ptr<ListPositionAllocator> positionAlloc = CreateObject
<ListPositionAllocator>();
  positionAlloc ->Add(Vector(100, 0, 0)); // node0
  positionAlloc ->Add(Vector(200, 0, 0)); // node1
  positionAlloc ->Add(Vector(450, 0, 0)); // node2
  positionAlloc ->Add(Vector(550, 0, 0)); // node3
  positionAlloc ->Add(Vector(650, 0, 0)); // node4
  positionAlloc ->Add(Vector(700, 0, 0)); // node5
  positionAlloc ->Add(Vector(750, 0, 0)); // node6
  positionAlloc ->Add(Vector(800, 0, 0)); // node7
  positionAlloc ->Add(Vector(860, 0, 0)); // node8
  positionAlloc ->Add(Vector(910, 0, 0)); // node9
  mobility.SetPositionAllocator(positionAlloc);

mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
  mobility.Install(c);
  AnimationInterface anim ("OAodv.xml"); // Mandatory
  AnimationInterface::SetConstantPosition (c.Get (0), 0, 500);
  AnimationInterface::SetConstantPosition (c.Get (1), 200, 500);
  AnimationInterface::SetConstantPosition (c.Get (2), 400, 500);
  AnimationInterface::SetConstantPosition (c.Get (3), 600, 500);
  AnimationInterface::SetConstantPosition (c.Get (4), 800, 500);
  AnimationInterface::SetConstantPosition (c.Get (5), 0, 600);
  AnimationInterface::SetConstantPosition (c.Get (6), 200, 600);
  AnimationInterface::SetConstantPosition (c.Get (7), 400, 600);
  AnimationInterface::SetConstantPosition (c.Get (8), 600, 600);
  AnimationInterface::SetConstantPosition (c.Get (9), 800, 600);
```

```
  AnimationInterface::SetConstantPosition (c.Get (10), 0, 700);
  AnimationInterface::SetConstantPosition (c.Get (11), 200, 700);
  AnimationInterface::SetConstantPosition (c.Get (12), 400, 700);
  AnimationInterface::SetConstantPosition (c.Get (13), 600, 700);
  AnimationInterface::SetConstantPosition (c.Get (14), 800, 700);
  AnimationInterface::SetConstantPosition (c.Get (15), 0, 400);
  AnimationInterface::SetConstantPosition (c.Get (16), 200, 400);
    anim.EnablePacketMetadata(true);
  Ptr<OutputStreamWrapper> routingStream =
Create<OutputStreamWrapper> ("blackhole.routes", std::ios::out);
  aodv.PrintRoutingTableAllAt (Seconds (45), routingStream);
  PcapHelper pcapHelper;
  Ptr<PcapFileWrapper> file = pcapHelper.CreateFile
("blockhole.pcap", std::ios::out, PcapHelper::DLT_PPP);
Config::ConnectWithoutContext("/NodeList/*/ApplicationList/*/$ns
3::PacketSink/Rx", MakeCallback (&ReceivePacket));
  FlowMonitorHelper flowmon;
  Ptr<FlowMonitor> monitor = flowmon.InstallAll();
  NS_LOG_INFO ("Run Simulation.");
  Simulator::Stop (Seconds(100.0));
  Simulator::Run ();
  monitor->CheckForLostPackets ();
  Ptr<Ipv4FlowClassifier> classifier =
DynamicCast<Ipv4FlowClassifier> (flowmon.GetClassifier ());
  std::map<FlowId, FlowMonitor::FlowStats> stats = monitor-
>GetFlowStats ();
uint32_t txPacketsum = 0;
        uint32_t rxPacketsum = 0;
        uint32_t DropPacketsum = 0;
        uint32_t LostPacketsum = 0;
        double Delaysum = 0;
```

```
  for (std::map<FlowId, FlowMonitor::FlowStats>::const_iterator i =
stats.begin (); i != stats.end (); ++i)
    {
            txPacketsum += i->second.txPackets;
              rxPacketsum += i->second.rxPackets;
              LostPacketsum += i->second.lostPackets;
              DropPacketsum += i->second.packetsDropped.size();
              Delaysum += i->second.delaySum.GetSeconds();
        Ipv4FlowClassifier::FiveTuple t = classifier->FindFlow (i-
>first);
         std::cout << "Transmission Flow " << i->first  << " (" <<
t.sourceAddress << " -> " << t.destinationAddress << ")\n";
          std::cout << "  Transmission Bytes:   " << i->second.txBytes
<< "\n";
          std::cout << "  Receiving Bytes:   " << i->second.rxBytes <<
"\n";
                 std::cout << "  Throughput: " << i->second.rxBytes * 8.0
/ (i->second.timeLastRxPacket.GetSeconds() - i-
>second.timeFirstTxPacket.GetSeconds())/1024/1024  << "
Mbps\n";
 std::cout << "  All Tx Packets: " << txPacketsum << "\n";
        std::cout << "  All Rx Packets: " << rxPacketsum << "\n";
        std::cout << "  All Delay: " << Delaysum / txPacketsum <<"\n";
        std::cout << "  All Lost Packets: " << LostPacketsum << "\n";
        std::cout << "  All Drop Packets: " << DropPacketsum << "\n";
        std::cout << "  Packets Delivery Ratio: " << ((rxPacketsum
*100) / txPacketsum) << "%" << "\n";
        std::cout << "  Packets Lost Ratio: " << ((LostPacketsum *100)
/ txPacketsum) << "%" << "\n";
      if ((t.sourceAddress=="10.1.2.2" && t.destinationAddress ==
"10.1.2.4"))
      {
```

```
        std::cout << "Transmission Flow " << i->first  << " (" <<
    t.sourceAddress << " -> " << t.destinationAddress << ")\n";
         std::cout << "  Transmission Bytes:   " << i->second.txBytes
    << "\n";
         std::cout << "  Receiving Bytes:   " << i->second.rxBytes <<
    "\n";


                std::cout << "  Throughput: " << i->second.rxBytes * 8.0
    / (i->second.timeLastRxPacket.GetSeconds() - i-
    >second.timeFirstTxPacket.GetSeconds())/1024/1024  << "
    Mbps\n";
     std::cout << "  All Tx Packets: " << txPacketsum << "\n";
          std::cout << "  All Rx Packets: " << rxPacketsum << "\n";
          std::cout << "  All Delay: " << Delaysum / txPacketsum <<"\n";
          std::cout << "  All Lost Packets: " << LostPacketsum << "\n";
          std::cout << "  All Drop Packets: " << DropPacketsum << "\n";
          std::cout << "  Packets Delivery Ratio: " << ((rxPacketsum
    *100) / txPacketsum) << "%" << "\n";
          std::cout << "  Packets Lost Ratio: " << ((LostPacketsum *100)
    / txPacketsum) << "%" << "\n";
        }
        }
      monitor->SerializeToXmlFile("lab-4.flowmon", true, true);
    }
```

E4.The complete Source code for the Modified OADV protocol source code known as MDPROADV.cc

```
#include "ns3/aodv-module.h"
#include "ns3/netanim-module.h"
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
```

```
#include "ns3/applications-module.h"
#include "ns3/mobility-module.h"
#include "ns3/wifi-module.h"
#include "ns3/netanim-module.h"
#include "ns3/flow-monitor-module.h"
#include "ns3/mobility-module.h"
#include "myapp.h"
NS_LOG_COMPONENT_DEFINE ("Blackhole");
using namespace ns3;
void
ReceivePacket(Ptr<const Packet> p, const Address & addr)
{
        std::cout << Simulator::Now ().GetSeconds () << "\t" << p-
>GetSize() <<"\n";
}
int main (int argc, char *argv[])
{
bool enableFlowMonitor = false;
std::string phyMode ("DsssRate1Mbps");
CommandLine cmd;
cmd.AddValue ("EnableMonitor", "Enable Flow Monitor",
enableFlowMonitor);
cmd.AddValue ("phyMode", "Wifi Phy mode", phyMode);
cmd.Parse (argc, argv);
  NS_LOG_INFO ("Create nodes.");
NodeContainer c; // ALL Nodes
NodeContainer not_malicious;
NodeContainer malicious;
  c.Create(100);  //17
  not_malicious.Add(c.Get(1));
not_malicious.Add(c.Get(2));
  not_malicious.Add(c.Get(3));
```

```
not_malicious.Add(c.Get(4));
 //not_malicious.Add(c.Get(5));
//not_malicious.Add(c.Get(6));
not_malicious.Add(c.Get(7));
not_malicious.Add(c.Get(8));
not_malicious.Add(c.Get(9));
not_malicious.Add(c.Get(10));
not_malicious.Add(c.Get(11));
not_malicious.Add(c.Get(12));
not_malicious.Add(c.Get(13));
not_malicious.Add(c.Get(14));
not_malicious.Add(c.Get(15));
not_malicious.Add(c.Get(16));
not_malicious.Add(c.Get(17));
not_malicious.Add(c.Get(18));
not_malicious.Add(c.Get(19));
//not_malicious.Add(c.Get(20));
not_malicious.Add(c.Get(21));
not_malicious.Add(c.Get(22));
not_malicious.Add(c.Get(23));
not_malicious.Add(c.Get(24));
not_malicious.Add(c.Get(25));
not_malicious.Add(c.Get(26));
not_malicious.Add(c.Get(27));
not_malicious.Add(c.Get(28));
not_malicious.Add(c.Get(29));
not_malicious.Add(c.Get(30));
not_malicious.Add(c.Get(31));
not_malicious.Add(c.Get(32));
not_malicious.Add(c.Get(33));
not_malicious.Add(c.Get(34));
not_malicious.Add(c.Get(35));
```

```
not_malicious.Add(c.Get(36));
not_malicious.Add(c.Get(37));
not_malicious.Add(c.Get(38));
not_malicious.Add(c.Get(39));
//not_malicious.Add(c.Get(40));
not_malicious.Add(c.Get(41));
not_malicious.Add(c.Get(42));
not_malicious.Add(c.Get(43));
not_malicious.Add(c.Get(44));
//not_malicious.Add(c.Get(45));
//not_malicious.Add(c.Get(46));
not_malicious.Add(c.Get(47));
not_malicious.Add(c.Get(48));
not_malicious.Add(c.Get(49));
not_malicious.Add(c.Get(50));
not_malicious.Add(c.Get(51));
not_malicious.Add(c.Get(52));
not_malicious.Add(c.Get(53));
not_malicious.Add(c.Get(54));
not_malicious.Add(c.Get(55));
not_malicious.Add(c.Get(56));
not_malicious.Add(c.Get(57));
not_malicious.Add(c.Get(58));
not_malicious.Add(c.Get(59));
//not_malicious.Add(c.Get(60));
not_malicious.Add(c.Get(61));
not_malicious.Add(c.Get(62));
not_malicious.Add(c.Get(63));
not_malicious.Add(c.Get(64));
//not_malicious.Add(c.Get(65));
//not_malicious.Add(c.Get(66));
not_malicious.Add(c.Get(67));
```

```
not_malicious.Add(c.Get(68));
not_malicious.Add(c.Get(69));
not_malicious.Add(c.Get(70));
not_malicious.Add(c.Get(71));
not_malicious.Add(c.Get(72));
not_malicious.Add(c.Get(73));
not_malicious.Add(c.Get(74));
not_malicious.Add(c.Get(75));
not_malicious.Add(c.Get(76));
not_malicious.Add(c.Get(77));
not_malicious.Add(c.Get(78));
not_malicious.Add(c.Get(79));
//not_malicious.Add(c.Get(80));
not_malicious.Add(c.Get(81));
not_malicious.Add(c.Get(82));
not_malicious.Add(c.Get(83));
not_malicious.Add(c.Get(84));
not_malicious.Add(c.Get(85));
not_malicious.Add(c.Get(86));
not_malicious.Add(c.Get(87));
not_malicious.Add(c.Get(88));
not_malicious.Add(c.Get(89));
not_malicious.Add(c.Get(90));
not_malicious.Add(c.Get(91));
not_malicious.Add(c.Get(92));
not_malicious.Add(c.Get(93));
not_malicious.Add(c.Get(94));
//not_malicious.Add(c.Get(95));
//not_malicious.Add(c.Get(96));
not_malicious.Add(c.Get(97));
not_malicious.Add(c.Get(98));
not_malicious.Add(c.Get(99));
```

```
      //not_malicious.Add(c.Get(100));
        malicious.Add(c.Get(0));
       malicious.Add(c.Get(5));
      malicious.Add(c.Get(6));
       malicious.Add(c.Get(20));
       malicious.Add(c.Get(45));
      malicious.Add(c.Get(46));
      malicious.Add(c.Get(60));
      malicious.Add(c.Get(65));
      malicious.Add(c.Get(66));
      malicious.Add(c.Get(80));
        // Set up WiFi
       WifiHelper wifi;
        YansWifiPhyHelper wifiPhy =  YansWifiPhyHelper::Default ();
      wifiPhy.SetPcapDataLinkType
(YansWifiPhyHelper::DLT_IEEE802_11);
        YansWifiChannelHelper wifiChannel ;
      wifiChannel.SetPropagationDelay
      ("ns3::ConstantSpeedPropagationDelayModel");
        wifiChannel.AddPropagationLoss
      ("ns3::TwoRayGroundPropagationLossModel",
        "SystemLoss", DoubleValue(1),
         "HeightAboveZ", DoubleValue(1.5));
      // For range near 250m
  wifiPhy.Set ("TxPowerStart", DoubleValue(33));
  wifiPhy.Set ("TxPowerEnd", DoubleValue(33));
  wifiPhy.Set ("TxPowerLevels", UintegerValue(1));
  wifiPhy.Set ("TxGain", DoubleValue(0));
  wifiPhy.Set ("RxGain", DoubleValue(0));
  wifiPhy.Set ("EnergyDetectionThreshold", DoubleValue(-61.8));
  wifiPhy.Set ("CcaMode1Threshold", DoubleValue(-64.8));
  wifiPhy.SetChannel (wifiChannel.Create ());
```

```
// Add a non-QoS upper mac
NqosWifiMacHelper wifiMac = NqosWifiMacHelper::Default ();
wifiMac.SetType ("ns3::AdhocWifiMac");
// Set 802.11b standard
wifi.SetStandard (WIFI_PHY_STANDARD_80211b);
wifi.SetRemoteStationManager ("ns3::ConstantRateWifiManager",
                              "DataMode",StringValue(phyMode),
                              "ControlMode",StringValue(phyMode));
    NetDeviceContainer devices;
   devices = wifi.Install (wifiPhy, wifiMac, c);
   //  Enable AODV
    AodvHelper aodv;
    AodvHelper malicious_aodv;
    // Set up internet stack
    InternetStackHelper internet;
    internet.SetRoutingHelper (aodv);
    internet.Install (not_malicious);
 malicious_aodv.Set("IsMalicious",BooleanValue(true)); // putting *false*
instead of *true* would disable the malicious behavior of the node
 internet.SetRoutingHelper (malicious_aodv);
 internet.Install (malicious);
 // Set up Addresses
 Ipv4AddressHelper ipv4;
 NS_LOG_INFO ("Assign IP Addresses.");
 ipv4.SetBase ("10.1.2.0", "255.255.255.0");
 Ipv4InterfaceContainer ifcont = ipv4.Assign (devices);
 NS_LOG_INFO ("Create Applications.");
 // UDP connection from N1 to N3
     uint16_t sinkPort = 6;
  std::cout << "  Sink Port:   " << sinkPort << "\n";
     Address sinkAddress (InetSocketAddress (ifcont.GetAddress (4),
    sinkPort)); // interface of n3
```

```
  PacketSinkHelper packetSinkHelper ("ns3::UdpSocketFactory",
InetSocketAddress (Ipv4Address::GetAny (), sinkPort));
  ApplicationContainer sinkApps = packetSinkHelper.Install (c.Get (4));
//n3 as sink
 sinkApps.Start (Seconds (0.));
sinkApps.Stop (Seconds (100.));
  Ptr<Socket> ns3UdpSocket = Socket::CreateSocket (c.Get (1),
UdpSocketFactory::GetTypeId ()); //source at n1
  Ptr<MyApp> app = CreateObject<MyApp> ();
app->Setup (ns3UdpSocket, sinkAddress, 1040, 5, DataRate
("250Kbps"));
c.Get (1)->AddApplication (app);
app->SetStartTime (Seconds (40.));
app->SetStopTime (Seconds (100.));
  MobilityHelper mobility;
Ptr<ListPositionAllocator> positionAlloc = CreateObject
<ListPositionAllocator>();
positionAlloc ->Add(Vector(100, 0, 0)); // node0
positionAlloc ->Add(Vector(200, 0, 0)); // node1
positionAlloc ->Add(Vector(450, 0, 0)); // node2
positionAlloc ->Add(Vector(550, 0, 0)); // node3
positionAlloc ->Add(Vector(650, 0, 0)); // node4
positionAlloc ->Add(Vector(700, 0, 0)); // node5
positionAlloc ->Add(Vector(750, 0, 0)); // node6
positionAlloc ->Add(Vector(800, 0, 0)); // node7
positionAlloc ->Add(Vector(860, 0, 0)); // node8
positionAlloc ->Add(Vector(910, 0, 0)); // node9
 mobility.SetPositionAllocator(positionAlloc);
 mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
 mobility.Install(c);
  AnimationInterface anim ("MDPRAODV.xml"); // Mandatory
AnimationInterface::SetConstantPosition (c.Get (0), 0, 500);
```

```
      AnimationInterface::SetConstantPosition (c.Get (1), 200, 500);
      AnimationInterface::SetConstantPosition (c.Get (2), 400, 500);
      AnimationInterface::SetConstantPosition (c.Get (3), 600, 500);
      AnimationInterface::SetConstantPosition (c.Get (4), 800, 500);
      AnimationInterface::SetConstantPosition (c.Get (5), 0, 600);
       AnimationInterface::SetConstantPosition (c.Get (6), 200, 600);
      AnimationInterface::SetConstantPosition (c.Get (7), 400, 600);
      AnimationInterface::SetConstantPosition (c.Get (8), 600, 600);
      AnimationInterface::SetConstantPosition (c.Get (9), 800, 600);
      AnimationInterface::SetConstantPosition (c.Get (10), 0, 700);
  AnimationInterface::SetConstantPosition (c.Get (11), 200, 700);
  AnimationInterface::SetConstantPosition (c.Get (12), 400, 700);
  AnimationInterface::SetConstantPosition (c.Get (13), 600, 700);
  AnimationInterface::SetConstantPosition (c.Get (14), 800, 700);
  AnimationInterface::SetConstantPosition (c.Get (15), 0, 400);
  AnimationInterface::SetConstantPosition (c.Get (16), 200, 400);
AnimationInterface::SetConstantPosition (c.Get (17), 0, 800);
AnimationInterface::SetConstantPosition (c.Get (18), 200, 800);
AnimationInterface::SetConstantPosition (c.Get (19), 400, 800);
AnimationInterface::SetConstantPosition (c.Get (20), 600, 800);
AnimationInterface::SetConstantPosition (c.Get (21), 800, 800);
      anim.EnablePacketMetadata(true);
  Ptr<OutputStreamWrapper> routingStream =
Create<OutputStreamWrapper> ("Cooperative_blackhole.routes",
std::ios::out);
  aodv.PrintRoutingTableAllAt (Seconds (45), routingStream);
  PcapHelper pcapHelper;
  Ptr<PcapFileWrapper> file = pcapHelper.CreateFile
("Cooperative_blockhole.pcap", std::ios::out, PcapHelper::DLT_PPP);
Config::ConnectWithoutContext("/NodeList/*/ApplicationList/*/$ns3::Pack
etSink/Rx", MakeCallback (&ReceivePacket));
  FlowMonitorHelper flowmon;
```

```
  Ptr<FlowMonitor> monitor = flowmon.InstallAll();
  NS_LOG_INFO ("Run Simulation.");
  Simulator::Stop (Seconds(100.0));
  Simulator::Run ();
  monitor->CheckForLostPackets ();
  Ptr<Ipv4FlowClassifier> classifier = DynamicCast<Ipv4FlowClassifier>
(flowmon.GetClassifier ());
  std::map<FlowId, FlowMonitor::FlowStats> stats = monitor-
>GetFlowStats ();
uint32_t txPacketsum = 0;
                uint32_t rxPacketsum = 0;
                uint32_t DropPacketsum = 0;
                uint32_t LostPacketsum = 0;
                double Delaysum = 0;
for (std::map<FlowId, FlowMonitor::FlowStats>::const_iterator i =
stats.begin (); i != stats.end (); ++i)
       {
txPacketsum += i->second.txPackets;
             rxPacketsum += i->second.rxPackets;
             LostPacketsum += i->second.lostPackets;
             DropPacketsum += i->second.packetsDropped.size();
             Delaysum += i->second.delaySum.GetSeconds();
       Ipv4FlowClassifier::FiveTuple t = classifier->FindFlow (i->first);
std::cout << "Malicious detected \n";
std::cout << "Transmission Flow " << i->first  << " (" << t.sourceAddress
<< " -> " << t.destinationAddress << ")\n";
        std::cout << "  Transmission Bytes:   " << i->second.txBytes << "\n";
        std::cout << "  Receiving Bytes:   " << i->second.rxBytes << "\n";
std::cout << "  Throughput: " << i->second.rxBytes * 8.0 / (i-
>second.timeLastRxPacket.GetSeconds() - i-
>second.timeFirstTxPacket.GetSeconds())/1024/1024  << " Mbps\n";
```

```
    if ((t.sourceAddress=="10.1.2.2" && t.destinationAddress ==
"10.1.2.4"))
 {
        std::cout << "Transmission Flow " << i->first  << " (" <<
t.sourceAddress << " -> " << t.destinationAddress << ")\n";
        std::cout << "  Transmission Bytes:   " << i->second.txBytes << "\n";
        std::cout << "  Receiving Bytes:   " << i->second.rxBytes << "\n";
std::cout << "  Throughput: " << i->second.rxBytes * 8.0 / (i-
>second.timeLastRxPacket.GetSeconds() - i-
>second.timeFirstTxPacket.GetSeconds())/1024/1024  << " Mbps\n";
         }
        }
 std::cout << "  Coperative blockhole attack detected\n";
      std::cout << "  All Tx Packets: " << txPacketsum << "\n";
              std::cout << "  All Rx Packets: " << rxPacketsum << "\n";
              std::cout << "  All Delay: " << Delaysum / txPacketsum <<"\n";
              std::cout << "  Data dopped by malicious attack: " <<
LostPacketsum << "\n";
               std::cout << "  All Drop Packets: " << DropPacketsum << "\n";
std::cout << "  Packets Delivery Ratio: " << ((rxPacketsum *100) /
txPacketsum) << "%" << "\n";
std::cout << "  Packets Lost Ratio: " << ((LostPacketsum *100) /
txPacketsum) << "%" << "\n";
  monitor->SerializeToXmlFile("lab-4.flowmon", true, true);
}
```