DENIAL OF SERVICES IN MANET USING NS2.0

Dr. Sudipta Majumder

Kripa Drishti Publications, Pune.

DENIAL OF SERVICES IN MANET USING NS2.0

Dr. Sudipta Majumder

Kripa-Drishti Publications, Pune.

Book Title: **Denial of Services in MANET using NS2.0**

Author by: **Dr. Sudipta Majumder**

1st Edition

ISBN: **978-93-90847-25-9**

9 789390 847259

Published: **January 2021**

Publisher:

KRIPA DRISHTI
PUBLICATIONS

Kripa-Drishti Publications

A-503 Poorva Heights, Pashan-Sus Road, Near Sai Chowk,

Pune – 411021, Maharashtra, India.

Mob: +91-8007068686

Email: editor@kdpublications.in

Web: https://www.kdpublications.in

[The responsibility for the facts stated, conclusions reached, etc., is entirely that of the author. The publisher is not responsible for them, whatsoever.]

This book is dedicated to

Maa, baba, Wife

And beloved daughter Sakshi

Acknowledgement

This book is based on research conducted on peer-to-peer based Video on demand service. I am grateful for a number of friends and colleagues encouraging me to start the work, persevere with it, and finally to publish it.

I would like to express my deepest sense of gratitude to my mentor Prof. Md. Anwar Hussain for his valuable suggestions, guidance, and immense moral support throughout my carrier. I am highly obliged to him for sharing his vast knowledge and experience throughout the tenure of my study.

Also, I would like to thank all the faculty members, friends and administration of north eastern Regional Institute of Science and Technology (NERIST) for encouraging me and giving me the opportunity to do research in the area.

I am also thankful to Dibrugarh University Institute of Engineering and Technology and Dibrugarh University for supporting me in every step.

Finally, I would like to acknowledge with gratitude, the support and love of my family- my parents, my brother, my wife and my daughter

PREFACE

MANET stands for Mobile ad-hoc Network also called a wireless ad-hoc network or ad-hoc wireless network that usually has a routable networking environment on top of a Link Layer ad hoc network. They consist of a set of mobile nodes connected wirelessly in a self-configured, self-healing network without having a fixed infrastructure. MANET nodes are free to move randomly as the network topology changes frequently. Each node behaves as a router as they forward traffic to other specified nodes in the network. The newcomers in this field of research, sometimes face Challenges regarding understanding new emergent Technologies. The book has chronological explanation of types of wireless networks, its vulnerabilities and NS2.0 implementation of attacks in MANET.

Features of the book: Several book features are designed to make it particularly easy for a student to understand the basics of various types of attack in mobile ad-hoc networks. The book also gives the reader a detailed insight into how black hole and grey attack implementation.

Structure: We have you read the book into a number of chapters. these chapters are logically arranged so that reader doesn't find it difficult to understand the concepts.

Visual approach: The highly technical subject matter is presented in this book. Complex numerical expressions and formulas are being avoided to make the book e more user-friendly for a new reader in the area. The book contains a balanced mix of text and figures. The book helps a lot in explaining the networking concepts which are obviously based on connections and transmissions.

Recommended reading: The book provides a detailed reference to the areas described in it. These references may help the reader to understand the matter more clearly. Each topic in the book is provided with a reference number. The reader may refer to the reference paper for more elaborative study

Chapters.

The book has got four main chapters namely introduction, literature review, attacks in AODV routing protocol and network simulator & a new routing protocol. In chapter one, we have discussed what is a wireless network and mobile ad-hoc network. In the second chapter, we have discussed various types of networks, IEEE 802.11 standards and routing in ad-hoc network and in the third chapter, the black hole and the grey hole attack in AODV routing protocol were studied. Finally in chapter fourth, we have shown implementation of a new protocol in NS 2.0 so that grey hole and black attacks can be implemented

INDEX

ISBN: 978-93-90847-25-9

Chapter 1

Introduction

1.1 Introduction:

Wireless ad-hoc networks are composed of autonomous nodes that are self- managed without any infrastructure. In this way, ad-hoc networks have a dynamic topology such that nodes can easily join or leave the network at any time. They have many potential applications, especially, in military and rescue areas such as connecting soldiers on the battlefield or establishing a new network in place of a network which collapsed after a disaster like an earthquake. Ad-hoc networks are suitable for areas where it is not possible to set up a fixed infrastructure. Since the nodes communicate with each other without an infrastructure, they provide the connectivity by forwarding packets over themselves.

To support this connectivity, nodes use some routing protocols such as AODV (Ad-hoc On-Demand Distance Vector), DSR (Dynamic Source Routing) and DSDV (Destination-Sequenced Distance-Vector). Besides acting as a host, each node also acts as a router to discover a path and forward packets to the correct node in the network. As wireless ad-hoc networks lack an infrastructure, they are exposed to a lot of attacks. One of these attacks is the Black Hole attack.

In the Black Hole attack, a malicious node absorbs all data packets in itself, similar to a hole which sucks in everything in. In this way, all packets in the network are dropped. A malicious node dropping all the traffic in the network makes use of the vulnerabilities of the route discovery packets of the on demand protocols, such as AODV. In route discovery process of AODV protocol, intermediate nodes are responsible to find a fresh path to the destination, sending discovery packets to the neighbor nodes. Malicious nodes do not use this process and instead, they immediately respond to the source node with false information as though it has fresh enough path to the destination.

Therefore source node sends its data packets via the malicious node to the destination assuming it is a true path. Black Hole attack may occur due to a malicious node which is deliberately misbehaving, as well as a damaged node interface. In any case, nodes in the network will constantly try to find a route for the destination, which makes the node consume its battery in addition to losing packets.

In our study, we simulated the Black Hole, Grey Hole and Worm Hole attack in wireless ad-hoc networks and evaluated its damage in the network. We made our simulations using NS-2 (Network Simulator version 2) simulation program that consists of the collection of all network protocols to simulate many of the existing network topologies. Even though NS-2 contains wireless ad-hoc routing protocols, it does not have any modules to simulate malicious protocols.

Thus, to simulate Black Hole attacks And Gray Hole Attack, we first added a new Black Hole and Gray Hole protocol into the NS-2. We started our study by writing a new AODV protocol using C++, to simulate the Black Hole attack.

Having implemented a new routing protocol which simulates the black hole we performed tests on different topologies to compare the network performance with and without black holes in the network. As expected, the throughput in the network was deteriorated considerably in the presence of a black hole.

Similarly, after implementing the gray whole attack, we performed tests on different topologies to compare the network performance with and without gray holes in the network. As a result we found out the throughput in the network was deteriorated in the presence of a gray hole.

And for worm whole attack, we made modification at the Mac layer. All the Mac layer protocol are available in the Mac directory under ns 2.32 directory. There we made modification in the Mac.cc and Mac.h file.

We have also implemented the impersonation attack and verified its existence from the trace file generated.

Chapter 2

Literature Review

2.1 Wireless Networks:

Wireless communication is used to transfer data among users without a wired infrastructure. Using electromagnetic waves, mobile users transmit and receive data over the air. Wireless communication spreads from home RF to satellites, from cellular phones to walkie-talkies.

Its mobility, simplicity and cost saving installation advantages make the wireless communication more popular, especially in recent decades Increasing user mobility needs and developments in the use of laptop computers and PDA's is one of the main reasons of the popularity of wireless networks.

2.2 Convenience Offered by Wireless Networks:

a. Mobility:

This is one of the obvious advantages of the wireless networks. Mobile users can connect to the existing networks while roaming freely and enjoying independence.

Simplicity we can translate simplicity into rapid development. It is easy to install a wireless infrastructure, compared to a wired network [10].

b. Flexibility:

Wireless network coverage area can reach where wire cannot go. It is very useful for moving vehicles or for the places where running cable is not possible like historical buildings [10].

2.3 Types of Networks:

According to coverage area, three type of wireless interconnection have been defined. Personal Area Networks (PANs), Local Area Networks (LANs) and Wide Area Networks (WANs).

2.3.1 Personal Area Networks (PAN):

PAN is a computer network used for communication among computer devices (including telephones, PDAs, etc.) close to one person. Typical PAN networks are Bluetooth, Sensor networks and zigbees.

The Standards Board of the IEEE approved the standard 802.15, as MAC and PHY Specifications for Wireless PANs (WPANs).

2.3.2 Local Area Networks (LAN):

In this type of network, devices are communicating with each other in a local coverage area that can be a building or a campus. Wireless LANs (WLANs) are alternatives of conventional wired LANs.

In a wired network nodes are communicating over physical environments such as cables. On the other hand, in a WLAN nodes use air as the medium. WLANs are standardized by Institute of Electrical and Electronics Engineers (IEEE).

2.3.3 Wide Area Networks (WAN):

WANs spread a relatively larger geographical area. Typically a WAN includes more than one LANs. 2G and 3G Mobile Cellular Networks, Satellite Systems and Paging Networks are examples of Wireless WANs (WWANs).

2.4 Wireless Local Area Networks (WLAN):

WLANs are alternative of conventional LANs that connect nodes in wired environments. WLANs transmit information over wireless medium instead of wire. A Wireless Local Area Networks (WLAN) is a shared medium communication network that broadcast information over wireless links to be received by all stations (e.g. computing devices).

WLANs are used mainly to connect to the Internet. Wireless internet access points are known as “hot spots” and are already available in coffeehouse and other public places such as airports, stations and hotels.

Thanks to these benefits, WLANs have gained significant popularity among mobile users to access real-time information.

Actually WLANs are implemented in mobile devices such as laptops, PDAs etc. to communicate with each other without using wired Ethernet (IEEE 802.3). In a WLAN, instead of wired Ethernet protocol, IEEE 802.3, wireless Ethernet protocol, IEEE 802.11 is used.

2.5 IEEE 802.11 Standards, Specifications and Technologies:

IEEE 802.11 is a member of the IEEE 802 protocol family, which defines specifications of Local Area Network (LAN) technologies. IEEE 802 specifications are focused on two lowest layers of the OSI model, the MAC and the physical (PHY) component that incorporate each other [7].

In the IEEE 802 series, individual specifications are determined after the point. 802.3, for example design Carrier Sense Multiple Access network with Collision Detection (CSMA/CD) and 802.5 is the Token-Ring specification.

Figure 2.1 shows the various components of the 802 family and their relation with the ISO models.

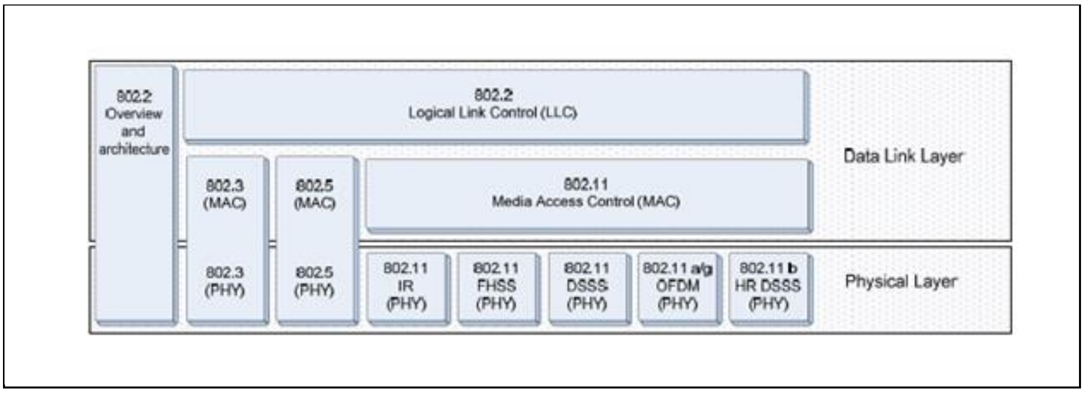


Figure 2.1. The Various Components of the 802 Family and their Relation with the ISO Models.

2.6 Ad-Hoc Network:

This network is called Independent Basic Service Set (IBSS) Stations in an IBSS communicate directly with each other and do not use an access point. Because of the mobility associated with ad-hoc networks, they are commonly called MANET (Mobile Ad-hoc NETwork). MANETs are self-organized networks whose nodes are free to move randomly while being able to communicate with each other without the help of an existing network infrastructure. MANETs are suitable for use in situations where any wired or wireless infrastructure is inaccessible, overloaded, damaged or destroyed such as emergency or rescue missions, disaster relief efforts and tactical battlefields, as well as civilian MANET situations, such as conferences and classrooms or in the research area like sensor networks. MANETs eliminate this dependence on a fixed network infrastructure where each station acts as an intermediate switch [7].

2.6.1 Comparison of Infrastructure and Ad-Hoc Networks:

In an infrastructure network, stations are required to be in the coverage area of access point. Therefore, mobility is limited with the distance between the access point and the station. But, in an Ad-hoc network, a station can transmit data to another one as long as there is a third station that can cover both of them. Data is forwarded via intermediate station/s using one of the ad-hoc network routing protocols.

This approach supports a larger working area but physical layer complexity increases. Stations may search for the target station that is out of range by flooding the network with broadcasts that are forwarded by each station. In an infrastructure network, access points can handle battery optimization for its stations.

While a station is in the power saving mode, access point can buffer frames for it. But in an ad-hoc network, power consumption is higher, since stations transmit frames that do not concern themselves. When transmitting packets over mobile nodes, hop count is changing in MANETs although in infrastructure WLANs, communication must have two hops. It is possible to include rapidly changing, random, multi-hop topologies in the routing function of MANET [5].

2.7 Routing In MANETs:

MANETs have special limitation and properties such as limited bandwidth and power, highly dynamic topology, high error rates etc., explained in the preceding sections [4]. Moreover, compared to infrastructure based networks, in a MANET, all nodes are mobile and can be connected dynamically in an arbitrary manner.

Nodes of MANET behave as router and take part in discovery and maintenance to establish a reliable route of each other. Therefore, routing protocols for wired networks cannot be directly used in wireless networks and numerous protocols have been developed for MANETs.

These routing protocols are divided into two categories based on management of routing tables. These categories are Table Driven Routing Protocols and On-Demand Routing Protocols, shown in the Table 2.1 and they are explained below:

MANET Routing Protocols	
Table Driven Routing Protocols	**On – Demand Routing Protocols**
Destination – Sequenced distance vector routing protocol (DSDV)	Ad – Hoc On – Demand distance vector routing (AODV)
Wireless routing protocol (WRP)	Cluster based routing protocols (CBRP)
Global state routing (GSR)	Dynamic source routing protocol (DSRP)
Fisheye state routing (FSR)	Temporally ordered routing algorithm (TORA)
Hierarchical state routing (HSR)	Associativity based routing (ABR)
Zone – based hierarchical link state routing protocol (ZHLS)	Signal stability routing (SSR)
Cluster head gateway switch routing protocol (CGSR)	

Figure 2.2. Categories of Routing Protocols

2.7.1 Table Driven Routing Protocols:

In Table Driven Routing Protocols, each node has to keep up-to-date routing tables. To maintain reliable routing tables, every node propagates the update messages to the network when the network topology changes. Because every node has information about network topology, Table Driven Routing Protocols present several problems.

- Periodically updating the network topology increases bandwidth overhead,
- Periodically updating route tables keeps the nodes awake and quickly exhaust their batteries,

Many redundant route entries to the specific destination needlessly take place in the routing tables. Destination-Sequenced Distance Vector Routing Protocol (DSDV), Wireless Routing Protocol (WRP), Global State Routing (GSR), Fisheye State Routing (FSR), Hierarchical

State Routing (HSR), Zone-based Hierarchical Link State Routing Protocol (ZHLS) and Cluster head Gateway Switch Routing Protocol (CGSR) are Table Driven Routing Protocols [10].

2.7.2 On-Demand Routing Protocols:

These protocols take a lazy approach to routing. [5] Compared to Table Driven Routing Protocols; On-Demand Routing Protocols are not maintained periodically, route tables are created when required.

When the source node wants to connect to the destination node, it propagates the route request packet to its neighbors. Just as neighbors of the source node receive the broadcasted request packet, they forward the packet to their neighbors and this action is happen until the destination is found.

Afterward, the destination node sends a replay packet the source node in the shortest path. The route remains in the route tables of the nodes through shortest path until the route is no longer needed [10].

Cluster based Routing Protocols (CBRP), Ad-Hoc On-Demand Distance Vector Routing (AODV), Dynamic Source Routing Protocol (DSRP), Temporally Ordered Routing Algorithm (TORA), Associatively Based Routing (ABR), Signal Stability Routing (SSR) are On-Demand Routing protocols. In our work, we have used Ad-Hoc On-Demand Distance Vector Routing (AODV) and implemented Black Hole attack to this protocol. AODV protocol and Black Hole Attack are detailed in next chapter.

2.8 Security Issues for MANETs:

Vulnerabilities of operating systems and upper layer applications that belong to user programs such as databases, browsers or client-server applications are not considered as a security issue for ad-hoc networks. General attack types are the threats against the routing layer of the ad-hoc networks; such as physical, MAC and network layer which is the most important function of wireless ad-hoc network for the routing mechanism, orienting the packets after a route discovery process. Other vulnerabilities are application security, network security, database security which are studied in different works which are not explained in detail here. Attacks to the wireless ad-hoc network in the networking layer usually have two purposes: not forwarding packets or adding and changing some parameters of routing messages; such as sequence number and IP addresses.

These will be detailed in the subsequent sections. Using one of the key mechanisms such as cryptography or authentication, or both in a network, serves as a preventive approach and can be employed against ‘attackers’.

However, these mechanisms protect the network against attacks that come from outside, malicious ‘insiders’ which use one of the critical keys can also threaten the security. For instance, in a battle field where ad-hoc networks are used, even if keys are protected by temper proof hardware that are used in the vehicles in the network, it is difficult to say that these vehicles exhibit the same behavior if the enemy captures them.

On the other hand, a node may un-deliberately misbehave as if it is damaged. A node with a failed battery which is unable to perform network operations may be perceived as an attack. Another malicious behavior of the nodes is selfishness. Selfish nodes refrain from consuming its resources; such as battery, by not participating in network operations.

Therefore; failed and selfish nodes also affect the network performance as they do not correctly process network packets, such as in routing mechanism. We should, therefore ensure that everything is correctly working in the network to support overall security and know how an insider is able to attack the wireless ad-hoc network. Wireless ad-hoc networks should be protected with an intrusion detection system that can understand the possible actions of attackers and can produce a solution against these attacks [2].

2.8.1 Attack Types:

A. Passive Eavesdropping:

An attacker can listen to any wireless network to know what is going on in the network. It first listens to control messages to infer the network topology to understand how nodes are located or are communicating with another. Therefore, it can gather intelligent information about the network before attacking. It may also listen to the information that is transmitted using encryption although it should be confidential belonging to upper layer applications.

Eavesdropping is also a threat to location privacy. An unauthorized node can notice a wireless network that exists within a geographical area, just by detecting radio signals. To combat this, traffic engineering techniques have been developed [3].

B. Selective Existence (Selfish Nodes):

This malicious node which is also known as selfish node and which is not participating in the network operations, use the network for its advantage to enhance performance and save its own resources such as power. To achieve that, selfish node puts forth its existence whenever personal cost is involved. Therefore these selfish node behaviors are known as selective existence attacks. [7]. For instance, selfish nodes do not even send any HELLO messages and drop all packets even if they are sent to itself, as long as it does not start the transmission. When a selfish node wants to start a connection with another node, it performs a route discovery and then sends him necessary packets. When the node no longer needs to use the network, it returns to the “silent mode” After a while, neighboring nodes invalidate their own route entries to this node and selfish node becomes invisible on the network. Actually, dropping packets may be divided into two categories according to the aims of the attacking node. Attacker may want to drop the packets of only the other nodes that it will attack later. To do that it must look at the packet to see whether it comes from this node. If attacker looks at the content of all packets aggregating from the network, it spends CPU resource and naturally battery life. This is not desirable behavior for selfish nodes because it spends battery life. Therefore, attackers are not interested in the content of the packets if its aim is not to consume its own resources. First category of dropping packets cannot be evaluated as a selfish node behavior. Thus, selectively dropping messages is not a selfish node behavior mentioned in [8]. Selective existence is kind of a passive attack, nodes just do not participate in the network operations and they do not change the content of packets [6].

C. Gray Hole Attack (Routing Misbehavior):

Gray whole attacks is an active attack type, which lead to dropping of messages. Attacking node first agrees to forward packets and then fails to do so. Initially the node behaves correctly and replays true RREP messages to nodes that initiate RREQ message. This way, it takes over the sending packets. Afterwards, the node just drops the packets to launch a (DoS) denial of service attack. If neighboring nodes that try to send packets over attacking nodes lose the connection to destination then they may want to discover a route again, broadcasting RREQ messages. Attacking node establishes a route, sending RREP messages. This process goes on until malicious node succeeds its aim (e.g. network resource consumption, battery consumption). This attack is known as routing misbehavior. Dropping packets is also one of the behaviors of failed or overloading nodes. One should not evaluate every dropping packet action as a selective existence, gray or black hole attack. Actually most routing protocols have no mechanism to detect whether data packets have been forwarded, DSR being the only exception [7].

D. Black Hole Attack:

The difference of Black Hole Attacks compared to Gray Hole Attacks is that malicious nodes never send true control messages initially. To carry out a black hole attack, malicious node waits for neighboring nodes to send RREQ messages.

When the malicious node receives an RREQ message, without checking its routing table, immediately sends a false RREP message giving a route to destination over itself, assigning a high sequence number to settle in the routing table of the victim node, before other nodes send a true one. Therefore requesting nodes assume that route discovery process is completed and ignore other RREP messages and begin to send packets over malicious node. Malicious node attacks all RREQ messages this way and takes over all routes. Therefore all packets are sent to a point when they are not forwarding anywhere. This is called a black hole akin to real meaning which swallows all objects and matter. To succeed a black hole attack, malicious node should be positioned at the center of the wireless network. If malicious node masquerades false RREP message as if it comes from another victim node instead of itself, all messages will be forwarded to the victim node. By doing this, victim node will have to process all incoming messages and is subjected to a sleep deprivation attack [8].

E. Impersonation:

Due to lack of authentication in ad-hoc networks, only MAC or IP addresses uniquely identify hosts. These addresses are not adequate to authenticate the sender node. Therefore non-repudiation is not provided for ad-hoc network protocols. MAC and IP spoofing are the simplest methods to pretend as another node or hide in the network. Malicious nodes achieve impersonation only by changing the source IP address in the control message. Another reason for impersonation is to persuade nodes to change their routing tables pretending to be a friendly node, such as attacks against routing table. One of the interesting impersonations is Man-in-the-middle attack. Malicious node performs this attack by combining spoofing and dropping attacks. Physically, it must be placed as the only node within the range for destination, in the middle of the route or victim node must be prevented from receiving any other route information to the destination.

Malicious node may also change the routing tables of the victim node to redirect its packets, using attacks against the routing table. At this point, malicious node waits for an RREQ message to the destination node from source node. When source node sends an RREQ message, malicious node drops the RREQ and replays a spoofed RREP message to source node as if it is coming from the destination node.

At the same time, malicious node sends a RREQ message to the destination node and drops the RREP message from the destination node. By doing this; malicious node manages to establish a route both to the source and the destination node and attacker controls the communication between the source and destination. If the communication is encrypted or entails an authentication as to MAC or IP address, malicious node can easily get the up layer communication [11].

F. Modification Attack:

Control massages are used to establish the shortest and true path between two nodes. But malicious nodes want to route packets to the direction that they want, modifying content of the control messages (e.g. RREQ, RREP and RERR). Modification means that the message does not carry out its normal functions. Route information such as hop count, sequence number, life time etc. are carried along with control messages.

This information has a big role in establishing a true route. Modifying these fields in the control messages, malicious node can perform its own attacks. Impersonation is not one of these kinds of attacks; impersonation is only performed by modifying source address to pretend as another node in the network. But changing route information in control messages is performed to mislead the victim or intermediate node and this modification is generally against the replay messages.

For example; by changing hop count or sequence number in the RREP messages, malicious node wants to change route information of victim node. In this attack type; malicious node decreases its sequence number in the RREP message, first capturing it, and finally sending it to the claimed node.

When victim node receives this false message it chooses the costly route in the network. Malicious node intends to perform this attack to affect the network performance, or its intension may be selfish, it does not want to route the packet. This attack can be performed by adding a number of virtual nodes and decreasing hop count field of the RREP messages. This attack is also known as detour attack. Another attack is performed by changing destination IP field in any control message. Thus, messages are not forwarded to relevant node and the communication is broken.

At the same time the malicious node may send all messages to the victim node to perform denial of service (DoS) attack or to another malicious node to collect the aggregated network dump.

To perform the latter one; more than one malicious node should be located in the network and one of them should be located in the middle of the network to collect messages. This way; collaborative malicious nodes can obtain all information about the network [11].

G. Attack against the Routing Tables:

Every node has its own routing table to find other nodes easily in the network. At the same time, this routing table draws the network topology for each node for a period (max. 3 seconds, duration of ACTIVE_ROUTE_TIMEOUT constant value of AODV protocol). If malicious node attacks against this table, attacked nodes do not find any route to other nodes that it wants to connect. This attack is always performed by fabricating a new control message. Therefore it is also named fabricating attack. There are many attacks against routing tables. Each one is done by fabricating false control messages.

For example; to attempt a black hole attack, malicious node first invades into the routing table of the victim, sending false RREP message. Malicious node also spreads false RERR messages to the network so that valid working links are marked as broken. Another attack type against the routing table is to attempt to create lots of route entries for non-existent nodes, using RREQ messages. As a result, routing table of the attacked node is full and does not have enough entry to create a new one. This attack type is known as routing table overflow. Attacks against the routing tables also affect the network integrity, changing the network topology established in the routing tables. Incorrect control messages are disseminated quickly in the network due to route discovery process and influence the network integrity in a wide area. Therefore attacks against the routing table are known as Network Integrity Attacks [6].

H. Sleep Deprivation Torture Attack (Battery Exhaustion):

Many techniques are used to maximize the battery life and mobile nodes prefer to stay at the sleep mode, when they are not used. Sleep Deprivation Torture is one of the serious types of Denial of Service Attacks, which affects only nodes, especially handheld devices that have limited resources. In a period time, attacker can propagate some control messages through the network, in which other nodes are interested. Other nodes pass to the operation mode from the sleep mode and start processing these unnecessary packets until their batteries completely run out [6].

I. Worm Hole Attack:

The wormhole attack [6] is quite severe, and consists in recording traffic from one region of the network and replaying it in a different region. It is carried out by an intruder node X located within transmission range of legitimate nodes A and B, where A and B are not themselves within transmission range of each other. Intruder node X merely tunnels control traffic between A and B (and vice versa), without the modification presumed by the routing protocol – e.g. without stating its address as the source in the packets header – so that X is virtually invisible.

Chapter 3

Attacks in AODV Routing Protocol

3.1 Black Hole, Gray Hole and Worm Hole Attack in AODV Routing Protocol:

Initially, we should take into account Ad-hoc On-Demand Distance Vector (AODV) Routing Protocol and then we shall explain Black Hole Attack.

3.2 Ad-Hoc on-Demand Distance Vector (AODV) Routing Protocol:

Ad-hoc On-Demand Distance Vector (AODV) Routing Protocol is used for finding a path to the destination in an ad-hoc network. To find the path to the destination all mobile nodes work in cooperation using the routing control messages. Thanks to these control messages, AODV Routing Protocol offers quick adaptation to dynamic network conditions, low processing and memory overhead, low network bandwidth utilization with small size control messages. The most distinguishing feature of AODV compared to the other routing protocols is that it uses a destination sequence number for each route entry. The destination sequence number is generated by the destination when a connection is requested from it. Using the destination sequence number ensures loop freedom. AODV makes sure the route to the destination does not contain a loop and is the shortest path. Route Requests (RREQs), Route Replay (RREPs), Route Errors (RERRs) are control messages used for establishing a path to the destination, sent using UDP/IP protocols. Header information of these control messages are explained in [11]. When the source node wants to make a connection with the destination node, it broadcasts an RREQ message. This RREQ message is propagated from the source, received by neighbors (intermediate nodes) of the source node. The intermediate nodes broadcast the RREQ message to their neighbors. This process goes on until the packet is received by destination node or an intermediate node that has a fresh enough route entry for the destination. Figure 3 shows how the RREQ message is propagated in an ad-hoc network.

Fresh enough means that the intermediate node has a valid route to destination formed a period of time ago, lower than the threshold. While the RREQ packet travels through the network, every intermediate node increases the hop count by one. If an RREQ message with the same RREQ ID is received, the node silently discards the newly received RREQs, controlling the ID field of the RREQ message. When the destination node or intermediate node that has fresh enough route to the destination receive the RREQ message they create an RREP message and update their routing tables with accumulated hop count and the sequence number of the destination node.

Afterwards the RREP message is unicasted to the source node. The difference between the broadcasting an RREQ and unicasting RREP can be seen from Figures 4.4 and 4.5. While the RREQ and the RREP messages are forwarded by intermediate nodes, intermediate nodes update their routing tables and save this route entry for 3 seconds, which is the ACTIVE_ROUTE_TIMEOUT constant value of AODV protocol. The default constant values of the AODV protocol are listed in appendix of RFC – 3561 [13].

Thus the node knows over which neighbor to reach at the destination. In terminology, the neighbor list for destination is labeled as "Precursor List". Figure 3.1 shows how the RREP message is unicasted and how the route entries in the intermediate nodes are updated.

3.3 Sequence Numbers:

Sequence Numbers serve as time stamps and allow nodes to compare how fresh their information on the other node is. However when a node sends any type of routing control message, RREQ, RREP, RERR etc., it increases its own sequence number. Higher sequence number is more accurate information and whichever node sends the highest sequence number, its information is considered and route is established over this node by the other nodes.

The sequence number is a 32-bit unsigned integer value (i.e., 4294967295). If the sequence number of the node reaches the possible highest sequence number, 4294967295, then it will be reset to zero (0). If the results of subtraction of the currently stored sequence number in a node and the sequence number of incoming AODV route control message is less than zero, the stored sequence number is changed with the sequence number of the incoming control message.

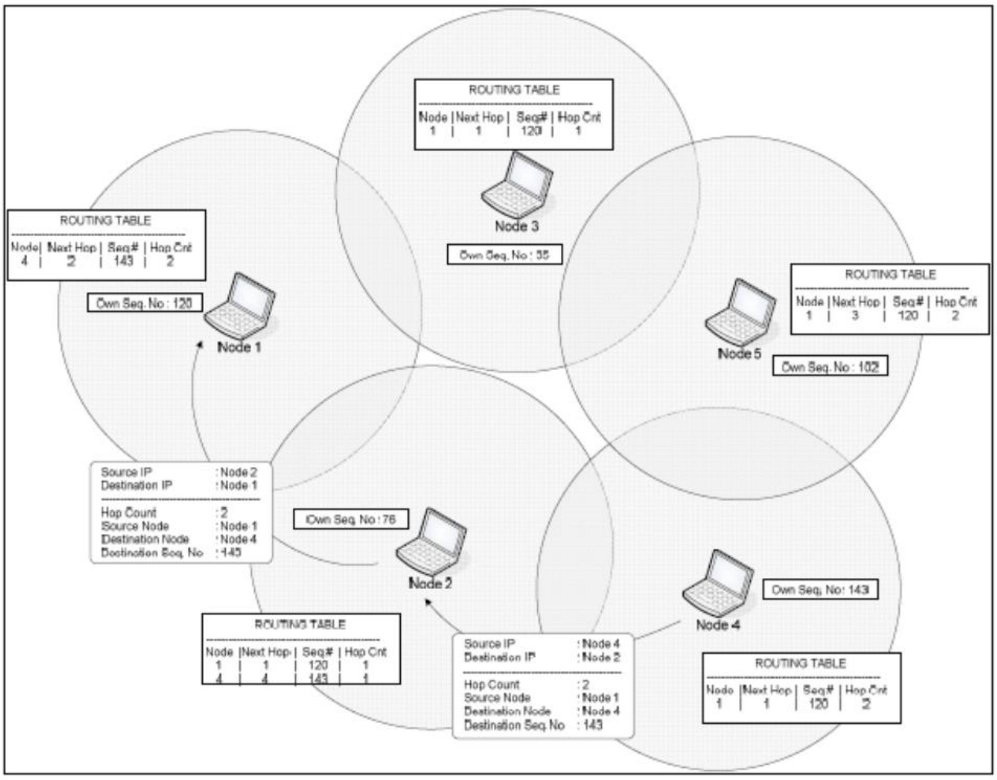


Figure 3.1 Showing How Sequence Number Works

In Figure 3.1, while Node 2 forwards the RREP message coming from Node 3, it compares its own previously stored sequence number with that of Node 3. If it notices that the sequence number is newer than its own, then it changes its route table entry as necessary.

3.4 Attacks:

a. Black Hole Attack:

Black hole attack is briefly explained in the previous Chapter. In this Chapter we shall explain it in more detail as we have already explained the AODV protocol. In an ad-hoc network that uses the AODV protocol, a Black Hole node absorbs the network traffic and drops all packets.

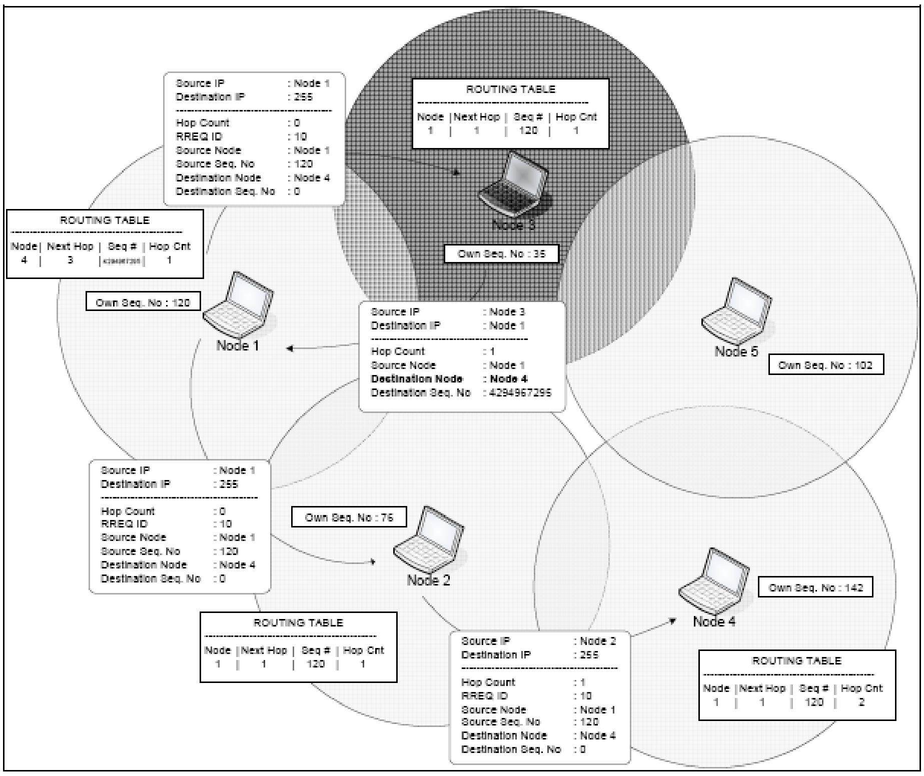


Figure 3.2.Working of Black Hole Attack

To explain the Black Hole Attack we added a malicious node that exhibits Black Hole behavior in the scenario of the figures of the previous section.

In this scenario shown in Figure 3.2, we assume that Node 3 is the malicious node. When Node 1 broadcasts the RREQ message for Node 4, Node 3 immediately responds to Node 1 with an RREP message that includes the highest sequence number of Node 4, as if it is coming from Node 4. Node 1 assumes that Node 4 is behind Node 3 with 1 hop and discards the newly received RREP packet come from Node 2. Afterwards Node 1 starts to send out its data packet to the node 3 trusting that these packets will reach Node 4 but Node 3 will drop all data packets. In a Black Hole Attack, after a while, the sending node understands that there is a link error because the receiving node does not send TCP ACK packets. If it sends out new TCP data packets and discovers a new route for the destination, the malicious node still manages to deceive the sending node. If the sending node sends out UDP data packets the problem is not detected because the UDP data connections do not wait for the ACK packets.

In our scenarios we use UDP data packets and we will explain our scenarios and their results in Chapter 5. Before Chapter 5 we will describe how Black Hole and Gray hole behavior how implemented in the simulator program, NS (Network Simulator).

b. Wormhole Attack:

For launching a wormhole attack, an adversary connects two distant points in the network using a direct low-latency communication link called as the wormhole ink.

The wormhole link can be established by a variety of means, e.g., by using an Ethernet cable etc, Long- range wireless transmission, or an optical link. Once the wormhole link is established, the adversary captures wireless transmissions on one end, sends them through the wormhole link and replays them at the other end.

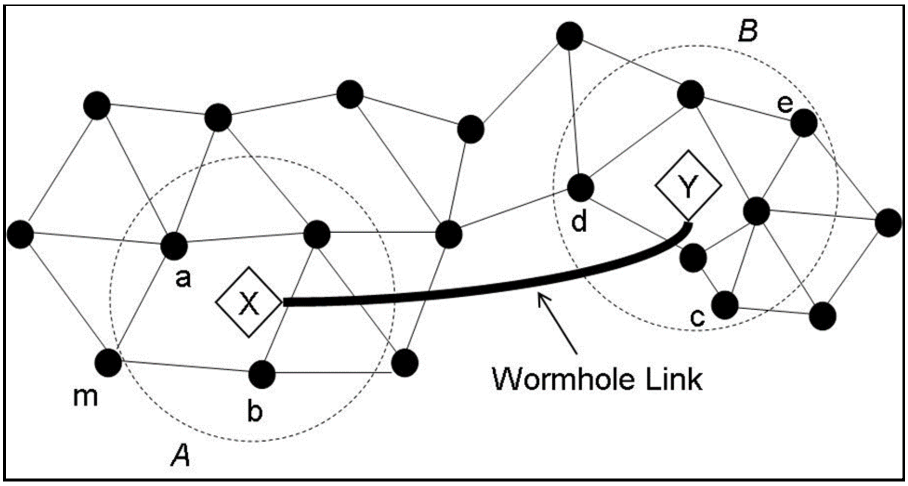


Figure 3.3. Working of Worm Whole Attack

An example is shown in the above figure. Here X and Y are the two end-points of the wormhole link (called as wormholes). X replays in its neighborhood (in area A) everything that Y hears in its own neighborhood (area B) and vice versa.

The net effect of such an attack is that all the nodes in area A assume that nodes in area B are their neighbors and vice versa. This, as a result, affects routing and other connectivity based protocols in the network.

Once the new routes are established and the traffic in the network starts using the X-Y shortcut, the wormhole nodes can start dropping packets and cause network disruption.

They can also spy on the packets going through and use the large amount of collected information to break any network security.

The wormhole attack will also affect connectivity-based localization algorithms and protocols based on localization, like geographic routing, will find many inconsistencies resulting in further network disruption [17].

c. Message Tampering:

In this attack a malicious node will involve in the routing as an intermediate node. It will add (or) delete some bytes of data packets received by it and forward to the destination node. Due to this abnormality or destruction of network takes place.

d. Byzantine Attack:

Here an intermediate node works alone or a set of intermediate nodes work in collusion and carry out attacks. These attacker nodes create routing loops, forwarding packets through non-optimal paths or selectively dropping packets which results in disruption or degradation of routing services [12].

e. Flooding Attack:

Flooding attack can be launched by flooding the network with fake RREQ's (or) data packets leading to the congestion of the network and reduces the probability of data transmission of the authorized nodes [13]. The detection of attack is very hard and it exhausts the network resources.

3.5 Information Disclosure:

Here authorized node acts as malicious node. It reveals the information regarding the location of node or sometimes structure of network.

It gathers the node location information such as route map, passwords, private keys and then plan further attacks. The leakage of information results in catastrophic situation in security sensitive scenarios [13].

Sleep Deprivation (or) Resource Consumption: Here a malicious node can attempt to consume battery life by requesting excessive routing discovery or by forwarding unnecessary packets to the victim node. Due to wastage of resources the performance of the network degrades.

3.6 Attacks on Routing:

Here a malicious node will get into the path between the source and destination nodes it then controls the flow of network traffic. These types of attacks can change the behavior of the routing protocol in the network. There are different types of routing attacks as

a. Routing Table Overflow Attack:

This occurs in proactive routing in which updating of routing information takes place periodically. Here the malicious node creates routes between unauthorized node and authorized nodes present in the network.

It tries to make the target systems routing table to overflow. The target is to have more routes such that it can prevent creation of new routes [13].

b. Routing Table Poisoning:

Here a malicious node in the network send fake routing updates or modify correct route update the data packets sent to other authorized nodes. This causes congestion in portions of the network, suboptimal routing and makes the network inaccessible [13].

c. Packet Replication:

Here a malicious node replicates the data packets as a result additional consumption of bandwidth and battery power resources takes place. This causes chaos in routing process [13].

d. Rushing Attack:

Here the aim is to control as much network traffic [14] as possible. The compromised node which receives a RREQ packet from the source node try to distribute the packet early throughout the network before the same RREQ packet reaches the other nodes. Nodes which receive the RREQ packet from the source node consider those packets as duplicates of RREQ packets which are already received through the compromised node and drop (or) discard them. As a result whenever source node discovers route the malicious nodes will become as the intermediate nodes of the route. Finding secure routes will become difficult task. Detection of this kind of attacks is very difficult

e. Route Cache Poisoning:

The information regarding known routes can be maintained by each node in route cache. The information in the route cache of the node can be altered or deleted by the malicious node. This is called as route cache poisoning. It results in congestion of some part of network or inaccessibility of some part of network [13].

Chapter 4

Network Simulator (NS) and a New Routing Protocol

4.1 Network Simulator (NS) and a New Routing Protocol:

In this work, we have tried to evaluate the effects of the Black Hole attacks in the wireless Ad-hoc Networks. To achieve this we have simulated the wireless ad-hoc network scenarios which includes Black Hole node using NS Network Simulator program. To simulate the Black Hole node in a wireless ad-hoc network we have implemented a new protocol that drops data packets after attracting them to itself. In this chapter we present NS and our contribution to this software.

4.2 NS Network Simulator:

NS is an event driven network simulator program, developed at the University of California Berkley, which includes many network objects such as protocols, applications and traffic source behavior. The NS is a part of software of the VINT project [15] that is supported by DARPA since 1995.

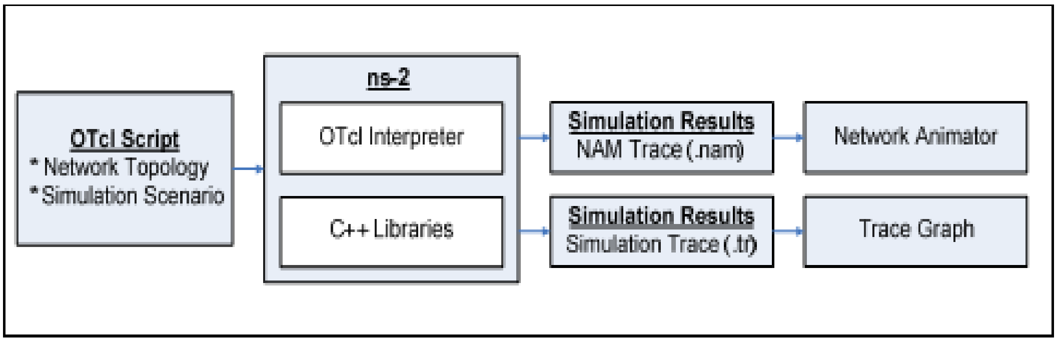


Figure 4.1. NS-2 Schema

At the simulation layer NS uses OTcl (Object oriented Tool Command Language) programming language to interpret user simulation scripts. OTcl language is in fact an object oriented extension of the Tcl Language. The Tcl language is fully compatible with the C++ programming language. At the top layer, NS is an interpreter of Tcl scripts of the users, they work together with C++ nodes. In Chapter 5 the usage of the Tcl Language will be explained in detail. As shown in Figure 4.1 [16], an OTcl script written by a user is interpreted by NS. While OTcl script is being interpreted, NS creates two main analysis reports simultaneously. One of them is NAM (Network Animator) object that shows the visual animation of the simulation. The other is the trace object that consists of the behavior of all objects in the simulation. Both of them are created as a file by NS.

Former is .nam file used by NAM software that comes along with NS. Latter is a “.tr” file that includes all simulation traces in the text format.

NS project is normally distributed along with various packages (ns, nam, tcl, otcl etc.) named as “all-in-one package”, but they can also be found and downloaded separately. In this study we have used version 2.29 of ns all-in-one package and installed the package in the Windows environment using Cygwin. After version 2, NS is commonly using a NS-2 and in our thesis we shell refer to it as NS-2.

We have written the “.tcl” files in text editor and analyzed the results of the “.tr” file using “cat”, “awk”, and “wc” and “grep” commands of Unix Operating System. The implementation phase of the Black hole behavior to the AODV protocol is written using C++.

Why two languages? Ns uses two languages because simulator has two different kinds of things it needs to do. On one hand, detailed simulations of protocols requires a systems programming language which can efficiently manipulate bytes, packet headers, and implement algorithms that run over large data sets. For these tasks run-time speed is important and turn-around time (run simulation, find bug, fix bug, recompile, re-run) is less important.

On the other hand, a large part of network research involves slightly varying parameters or configurations, or quickly exploring a number of scenarios. In these cases, iteration time (change the model and re-run) is more important. Since configuration runs once (at the beginning of the simulation), run-time of this part of the task is less important.

Ns meets both of these needs with two languages, C++ and OTcl. C++ is fast to run but slower to change, making it suitable for detailed protocol implementation. OTcl runs much slower but can be changed very quickly (and interactively), making it ideal for simulation configuration. Ns (via tclcl) provides glue to make objects and variables appear on both languages [10].

For more information about the idea of scripting languages and split-language programming, see Ousterhout's article in IEEE Computer [15]. For more information about split level programming for network simulation, see the ns paper [16].

Which language for what? Having two languages raises the question of which language should be used for what purpose.

Our basic advice is to use OTcl:

- For configuration, setup, and ``one-time" stuff.
- If you can do what you want by manipulating existing C++ objects.

And use C++:

- If you are doing anything that requires processing each packet of a flow.
- If you have to change the behavior of an existing C++ class in ways that weren't anticipated.

For example, links are OTcl objects that assemble delay, queuing, and possibly loss modules. If your experiment can be done with those pieces, great. If instead you want do something fancier (a special queuing discipline or model of loss), then you'll need a new C++ object [10].

There are certainly grey areas in this spectrum: most routing is done in OTcl (although the core Dijkstra algorithm is in C++).

We've had HTTP simulations where each flow was started in OTcl and per-packet processing was all in C++. This approach worked OK until we had 100s of flows starting per second of simulated time. In general, if you're ever having to invoke Tcl many times.

4.2.1 The Class Simulator:

The overall simulator is described by a Tcl Simulator./ns-2/ns-lib.h. It provides a set of interfaces for configuring a simulation and for choosing the type of event scheduler used to drive the simulation.

A simulation script generally begins by creating an instance of this class and calling various methods to create nodes, topologies, and configure other aspects of the simulation. A subclass of Simulator called OldSim is used to support ns v1 backward compatibility.

The procedures and functions described in this chapter can be found in ~ns/tcl/lib/ns-lib.tcl, ~ns/scheduler.{cc, h}, and, ~ns/heap.h.

4.2.2 Schedulers and Events:

The simulator is an event-driven simulator. There are presently four schedulers available in the simulator, each of which is implemented using a different data structure: a simple linked-

List, heap, calendar queue (default), and a special type called ``real-time".

Each of these are described below. The scheduler runs by selecting the next earliest event, executing it to completion, and returning to execute the next event.

Unit of time used by scheduler is seconds. Presently, the simulator is single-threaded, and only one event in execution at any given time.

If more than one event are scheduled to execute at the same time, their execution is performed on the first scheduled - first dispatched manner. Simultaneous events are not re-ordered anymore by schedulers (as it was in earlier versions) and all schedulers should yield the same order of dispatching given the same input.

No partial execution of events or pre-emption is supported.

An event generally comprises a “firing time" and a handler function. The actual definition of an event is found in ~ns/scheduler.h:

```
class Event {

            public:

                  Event* next_;         /* event list /

                  Handler* handler_;    /* handler to call when event ready /

                  double time_;         /* time at which event is ready /

                  int uid_;             /* unique ID /

                        Event (): time_ (0), uid_ (0) {}

                  };

            /*

            * The base class for all event handlers.  When an event's scheduled

            * time arrives, it is passed to handle which must consume it.

            * {\i.e. if it needs to be freed it, it must be freed by the handler.}

            */

      class Handler {
                  public:
                        virtual void handle (Event* event);
                  };
```

Two types of objects are derived from the base Event./ns-2/scheduler.cc: packets and ``at-events".

Packets are described in detail in the next chapterChaptersec: packetclass. An at-event is a tcl procedure execution scheduled to occur at a particular time. This is frequently used in simulation scripts. A simple example of how it is used is as follows:

```
\ldots

      Set ns_ [new Simulator]

      $ns_ use-scheduler Heap

      $ns_ at 300.5 "$self complete_sim"

      \ldots
```

This tcl code fragment first creates a simulation object, then changes the default scheduler implementation to be heap-based (see below), and finally schedules the function $self complete_sim to be executed at time 300.5 (seconds) (Note that this particular code fragment expects to be encapsulated in an object instance procedure, where the appropriate reference to $self is correctly defined.). At-events are implemented as events where the handler is effectively an execution of the tcl interpreter.

4.2.3 The List Scheduler:

The list scheduler (Scheduler/List.../ns-2/scheduler.cc) implements the scheduler using a simple linked-list structure. The list is kept in time-order (earliest to latest), so event insertion and deletion require scanning the list to find the appropriate entry. Choosing the next event for execution requires trimming the first entry off the head of the list. This implementation preserves event execution in a FIFO manner for simultaneous events.

4.2.4 The Heap Scheduler:

The heap scheduler (Scheduler/Heap.../ns-2/scheduler.cc) implements the scheduler using a heap structure. This structure is superior to the list structure for a large number of events, as insertion and deletion times are in for events.

This implementation in ns v2 is borrowed from the MaRS-2.0 simulator; it is believed that MaRS itself borrowed the code from NetSim, although this lineage has not been completely verified.

4.2.5 The Calendar Queue Scheduler:

The calendar queue scheduler (Scheduler/Calendar.../ns-2/scheduler.cc) uses a data structure analogous to a one-year desk calendar, in which events on the same month/day of multiple years can be recorded in one day. It is formally described in, and informally described in Jain. The implementation of Calendar queues in ns v2 was contributed by David Wetherill (presently at MIT/LCS).

The calendar queue scheduler since ns v2.33 is improved by the following three algorithms:

- A heuristic improvement that changes the linear search direction in enqueue operations. The original implementation searches the events in a bucket in chronological order to find the in-order spot for the event that is being inserted.
- The new implementation searches the bucket in reverse chronological order because the event being inserted is usually later than most of the events that are already in the bucket.
- A new bucket width estimation that uses the average interval of dequeued events as the estimation of bucket width. It is stated in that the optimal bucket width should be the average interval of all events in the future. The original implementation uses the average interval of future events currently in the most crowded bucket as the estimation. This estimation is unstable because it is very likely that many future events will be inserted into the bucket after this estimation, significantly changing the averaged event interval in the bucket. The new implementation uses the observed event interval in the past, which will not change, to estimate the event interval in future.

- SNOOPy Calendar Queue: a Calendar queue variant that dynamically tunes the bucket width according to the cost trade-off between enqueue operations and dequeue operation.

  The SNOOPy queue improvement is described in. In this implementation, there is one tcl parameter adjust_new_width_interval_ specifying the interval with which the SNOOPy queue should re-calculate the bucket width.

  Setting this parameter to 0 turns off the SNOOPy queue algorithm and degrades the scheduler back to the original Calendar Queue. In general, normal simulation users are not expected to change this parameter.

4.3. Implementing a New Routing Protocol in NS to Simulate Black Hole / Grey Hole Behavior:

In [17] Implementation of a New Manet Unicast Routing Protocol in NS-2 is described. To implement our contribution we have used the details explained in this paper.

In our work, we have used the nodes that exhibit black hole behavior in wireless ad-hoc network that use AODV protocol. Since the nodes behave as a Black Hole they have to use a new routing protocol that can participate in the AODV messaging. Implementation of this new routing protocol is explained below in detail:

All routing protocols in NS are installed in the directory of “ns-2.29”. We start the work by duplicating AODV protocol in this directory and change the name of directory as “blackholeaodv”.

Names of all files that are labeled as “aodv” in the directory are changed to “blackholeaodv” such as blackholeaodv.cc, blackholeaodv.h, blackholeaodv.tcl, blackholeaodv_rqueue.cc, blackholeaodv_rqueue.h etc. in this new directory except for “aodv_packet.h”.

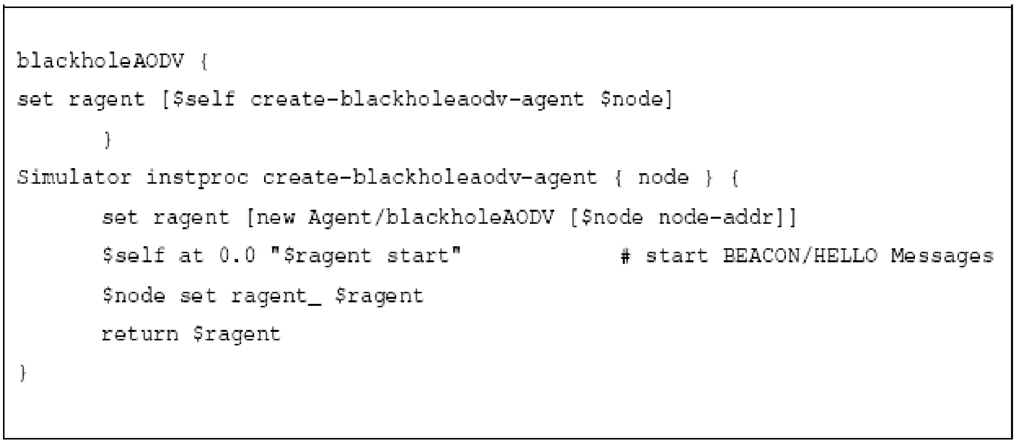
The key point in our work is that AODV and Black Hole AODV protocol will send each other the same AODV packets. Therefore, we did not copy “aodv_packet.h” file into the blackholeaodv directory.

We have changed all classes, functions, structs, variables and constants names in all the files in the directory except struct names that belong to AODV packet.h code.

We have designed aodv and blackholeoadv protocols to send each other aodv packets. These two protocols are actually the same.

After the above changes, we have changed two common files that are used in NS-2 globally to integrate new blackholeaodv protocol to the simulator. In [17] more files are changed to add new routing protocol and this new protocol uses its own packets.

But in our implementation we do not need to add a new packet. Therefore we have changed only two files. The changes are explained below.



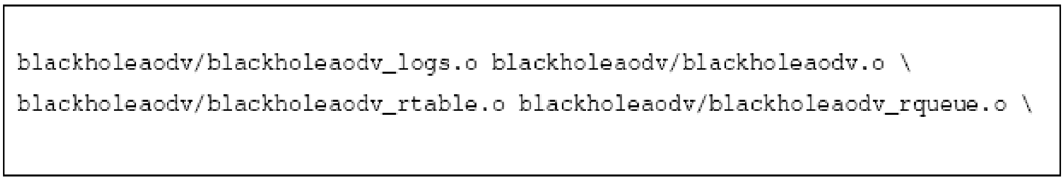
```
blackholeAODV {
set ragent [$self create-blackholeaodv-agent $node]
       }
Simulator instproc create-blackholeaodv-agent { node } {
       set ragent [new Agent/blackholeAODV [$node node-addr]]
       $self at 0.0 "$ragent start"          # start BEACON/HELLO Messages
       $node set ragent_ $ragent
       return $ragent
}
```

Figure 4.2 Blackhole "Aodv" Protocol Agent is added in "\Tcl\Lib\ NS-Lib.Tcl"

The First file modified is "\tcl\lib\ ns-lib.tcl" where protocol agents are coded as a procedure. When the nodes use black hole aodv protocol, this agent is scheduled at the beginning of the simulation and it is assigned to the nodes that will use black hole adodv protocol.

The agent procedure for blackholeaodv is shown in Figure 4.2. Second file which is adapted is "\makefile" in the root directory of the "ns-2.29".

After all implementations are ready, we have to compile NS-2 again to create object files. We have added the below lines in Figure 4.3 to the "\makefile".



```
blackholeaodv/blackholeaodv_logs.o blackholeaodv/blackholeaodv.o \
blackholeaodv/blackholeaodv_rtable.o blackholeaodv/blackholeaodv_rqueue.o \
```

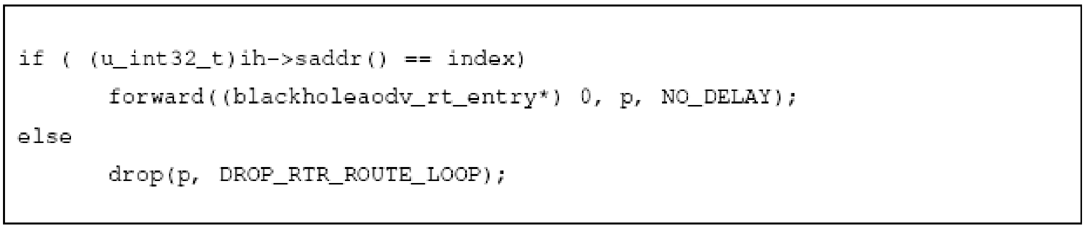
Figure 4.3 Addition to the "Makefile"

So far, we have implemented a new routing protocol which is labeled as blackholeaodv. But Black Hole behaviors have not yet been implemented in this new routing protocol.

To add Black Hole behavior into the new AODV protocol we made same changes in blackholeaodv/blackholeaodv.cc C++ file. We will describe these changes we made in blackholeaodv/blackholeaodv.cc file explaining working mechanism of the AODV and Black Hole AODV protocols below.

When a packet is received by the "recv" function of the "aodv/aodv.cc", it processes the packets based on its type.

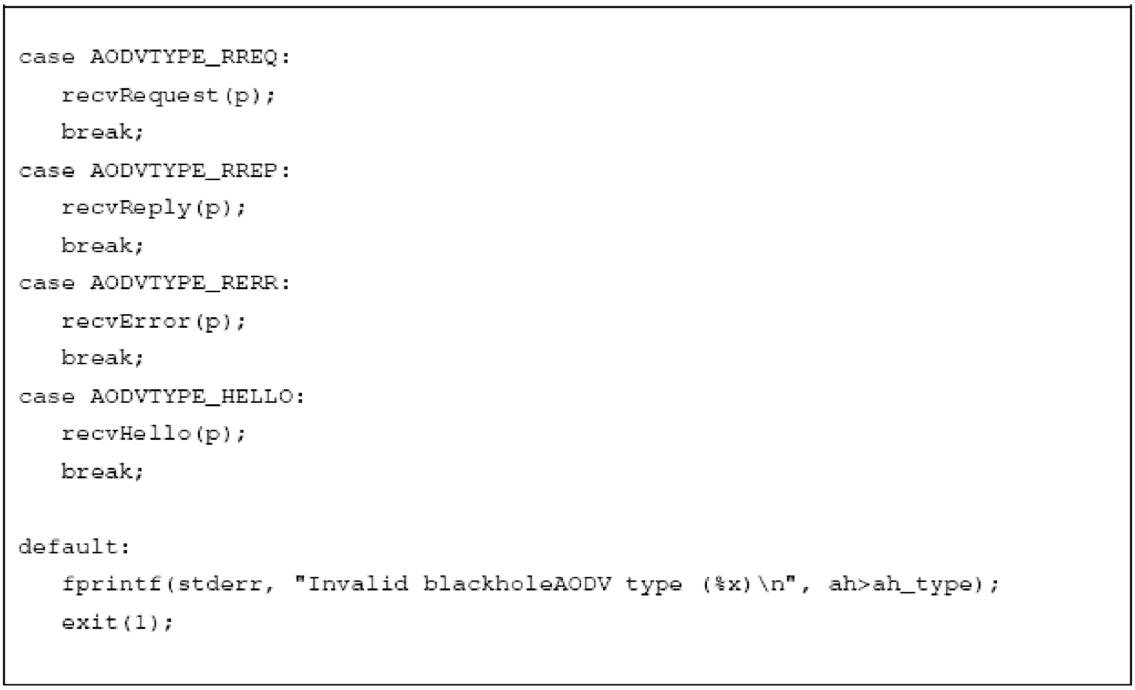
If packet type is any of the many AODV route management packets, it sends the packet to the "recvAODV" function that we will explain below.



```
if ( (u_int32_t)ih->saddr() == index)
       forward((blackholeaodv_rt_entry*) 0, p, NO_DELAY);
else
       drop(p, DROP_RTR_ROUTE_LOOP);
```

Figure 4.4 If" Statement for Dropping or Accepting the Packets

If the received packet is a data packet, normally AODV protocol sends it to the destination address, but behaving as a Black Hole it drops all data packets as long as the packet does not come to itself. In the code below, the first "if" condition provides the node to receive data packets if it is the destination. The "else" condition drops all remaining packets. If statement is shown in Figure 4.4. If the packet is an AODV management packet, "recv" function sends it to "recvblackholeAODV" function. "RecvblackholeAODV" function checks the type of the AODV management packet and based on the packet type it sends them to appropriate function with a "case" statement. For instance; RREQ packets are sent to the "recvRequest" function, RREP packets to "recvReply" function etc. case statements of "recvblackholeAODV" function is shown in Figure 4.5.



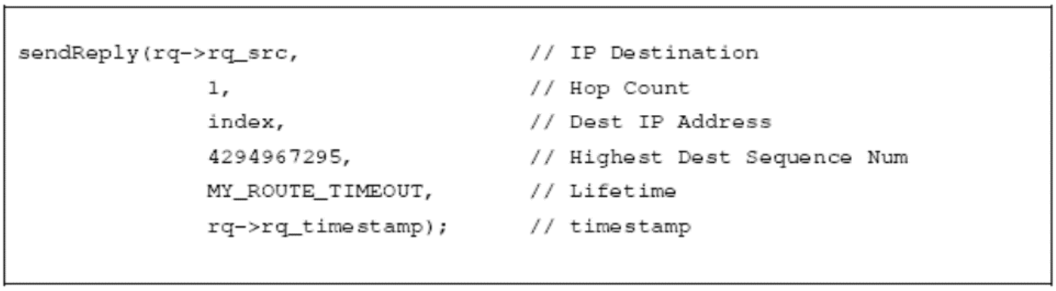
```
case AODVTYPE_RREQ:
   recvRequest(p);
   break;
case AODVTYPE_RREP:
   recvReply(p);
   break;
case AODVTYPE_RERR:
   recvError(p);
   break;
case AODVTYPE_HELLO:
   recvHello(p);
   break;

default:
   fprintf(stderr, "Invalid blackholeAODV type (%x)\n", ah>ah_type);
   exit(1);
```

Figure 4.5 Case Statement for Choosing the Aodv Control Message Types

In our case we will consider the RREQ function because Black Hole behavior is carried out as the malicious node receives an RREQ packet. When malicious node receives an RREQ packet it immediately sends RREP packet as if it has fresh enough path to the destination. Malicious node tries to deceive nodes sending such an RREP packet.

Highest sequence number of AODV protocol is 4294967295, 32 bit unsigned integer value [13]. Values of RREP packet that malicious node will send are described below. The sequence number is set to 4294967295 and hop count is set to.

The false RREP message of the Black Hole Attack is shown in Figure 4.6.



```
sendReply(rq->rq_src,                  // IP Destination
              1,                       // Hop Count
              index,                   // Dest IP Address
              4294967295,              // Highest Dest Sequence Num
              MY_ROUTE_TIMEOUT,        // Lifetime
              rq->rq_timestamp);       // timestamp
```

Figure 4.6 False RREP Message of Black Hole Attack

After all changes are finished we have recompiled all NS-2 files to create object files. Having finished compilation, we have a new test bed to simulate Black Hole Attack in AODV protocol. In the next chapter we will describe the simulations and simulation results.

4.4 Conclusions:

In the chapter 2, we have examined the conveniences offered by wireless network. Different types of networks and routing were also discussed. The security issues related to MANET summarized. In chapter 3, we have a detailed study of AODV protocol, its loopholes and attacks.

What is network simulator (NS 2.32) and how does it work, were discussed in chapter 4. Besides this, implementing a new protocol in the simulator were also examined. Finally, how black hole attack and grey whole attacks are implemented foe MANET were demonstrated in the chapter.

4.5 References:

1. F.Kargl, A.Klenk, S.Schlott andM.Weber, “Advanced Detection of Selfish or Malicious Nodes in Ad-Hoc Network”.
2. B.Xie and A.Kumar, “A Framework for Integrated Internet and Ad-Hoc Network Security”
3. I. Aad, J.P.Hubaux and E.W.Knightly, “Denial of Service Resilience in Ad-Hoc Network”.
4. I.Stamouli, P.G.Argyroudis and H.Tewari, “Real Time Intrusion Detection for Ad-Hoc Network”.
5. J.kong, H.Luo, K.Xu, D.L. GU, M.Gerla and S. Lu, “Adaptive Security for Multilevel Adhoc Networks”.
6. S. Sharma and R. Gupta, “Simulation Study of Blackhole Attack in the Mobile Ad Hoc Networks”.

7. M.Revankar, “Attacks in Ad-Hoc Networks and Modeling in NS-2”.
8. S. Dokurer, Y. M. Erten and C.E. Acar, “Performance analysis of ad-hoc networks under black hole attacks”
9. F. J. Ros and P. M. Ruiz, “Implementing a New Manet Unicast Routing Protocol in S2”.
10. The ns Manual (formerly ns Notes and Documentation).
11. Kurose, Ross “How to Misuse Aodv: A Case Study of Insider Attacks Against Ad- Hoc Routing Protocols”.
12. Shaik Noor Mohammad, “Security Attacks in MANETS (Survey Prospective)”, International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958, Volume-6 Issue-3, February 2017
13. S.Nithya, S.Prema, G.Singh, “Security Issues & Challenging Attributes in Mobile Ad-Hoc Networks '', International Research Journal of Engineering and Technology (IRJET), Volume: 03 Issue: 01, P.P 1083-1087, Jan-2016
14. Zhou L, Chao H-C, “Multimedia Traffic Security Architecture for the Internet of Things” IEEE Network 25(3):29–34. IEEE 2011.
15. John Ousterhout, Scripting: Higher-level programming for the 21st century.IEEE Computer, 31(3):23-30, March 1998.
16. Sandeep Bajaj, Lee Breslau, Deborah Estrin, Kevin Fall, Sally Floyd, Padma Haldar, Mark Handley, Ahmed Helmy, John Heidemann, Polly Huang, Satish Kumar, Steven McCanne, Reza Rejaie, Puneet Sharma, Kannan Varadhan, Ya Xu, Haobo Yu, and Daniel Zappala. Improving simulation for network research.Technical Report 99-702b, University of Southern California, March 1999. (Revised September 1999).
17. Hussain, Salim. (2015). Incorporation of Security Mechanism in AODV Routing Protocol to Eliminate The Effect of Black Hole Attack. 10.13140/RG.2.1.1250.8885

About The Book

The book uses plane and lucid language to explain the fundamentals of Mobile ad hoc networks. This book explains the standards and routings associated with Mobile ad hoc networks (MANETS). The book not only covers the security issues related to the network but also explain the philosophy of denial-of-service attacks. Various types of attacks possible in the mobile ad-hoc network have been added and discussed in chapters.

The book also so gives us an insight into Network Simulator 2.0 and the implementation of New routing protocol to simulate black hole and grey hole attack.

About The Author

Dr. Sudipta Majumder

Sudipta Majumdar, *PhD, is an assistant professor in the department of Computer Science and Engineering at Dibrugarh University Institute of Engineering and Technology (DUIET), Dibrugarh University, Assam.*

He has vast practical experience in computer networking and related fields. He is having academic experience of more than one decade in teaching and has published numerous research articles.

KRIPA DRISHTI PUBLICATIONS

Kripa-Drishti Publications
A-503 Poorva Heights, Pashan-Sus Road, Near Sai Chowk,
Pune – 411021, Maharashtra, India.
Mob: +91 8007068686
Email: editor@kdpublications.in
Web: https://www.kdpublications.in

ISBN: 978-1-6780-9552-9

9 781678 095529