

WORMHOLE ATTACK DETECTION

A Project Report

Submitted in partial fulfilment of the requirements for the award of the Degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

by

P. AJAY BABU
(17FE1A05A0)

P. MALLIKARJUN REDDY
(17FE1A0597)

K. SPOORTHY
(17FE1A0563)

N. GEETHA MADHURI
(17FE1A0590)

Under the guidance of

Mr. R. PRATHAP KUMAR, M. Tech,(Ph.D.)
Assistant Professor
Department of Computer Science and Engineering



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
VIGNAN'S LARA INSTITUTE OF TECHNOLOGY AND SCIENCE
(Affiliated to Jawaharlal Nehru Technological University Kakinada, Kakinada)

(An ISO 9001-2008 Certified Institution, Approved by AICTE),

Vadlamudi, Guntur Dist., Andhra Pradesh-522213,

July- 2021.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VIGNAN'S LARA INSTITUTE OF TECHNOLOGY AND SCIENCE

(Affiliated to Jawaharlal Nehru Technological University Kakinada, Kakinada)

(An ISO 9001-2015 Certified Institution, Approved by AICTE) Vadlamudi-522213



CERTIFICATE

This is to certify that the project report entitled “**WORMHOLE ATTACK DETECTION**” is a bonafide work done by **P. AJAY BABU (17FE1A05A0)**, **P.MALLIKARJUNA REDDY (17FE1A0597)**, **K. SPHOORTHY (17FE1A0563)** and **N. GEETHA MADHURI (17FE1A0590)** under my guidance and submitted in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in **COMPUTER SCIENCE AND ENGINEERING** from **JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY KAKINADA, KAKINADA**. The work embodied in this project report is not submitted to any University or Institution for the award of any Degree or Diploma.

Project Guide

Mr. R. PRATHAP KUMAR, M.Tech (Ph.D.)

Assistant Professor

Head of the Department

Mr. T. V. VAMSI KRISHNA, M.Tech(Ph.D.)

Assistant Professor

External Examiner

DECLARATION

We hereby declare that the project report entitled “**WORMHOLE ATTACK DETECTION**” is a record of an original work done by us under the guidance of **Mr. R. PRATHAP KUMAR**, Assistant Professor, Department of Computer Science and Engineering and this project report is submitted in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science and Engineering. The results embodied in this project report are not submitted to any other University or Institute for the award of any Degree or Diploma.

	Project Members	Signature
Place: Vadlamudi	P. Ajay Babu (17FE1A05A0)	_____
Date:	P. Mallikarjuna Reddy (17FE1A0597)	_____
	K. Sphoorthy (17FE1A0563)	_____
	N. Geetha Madhuri (17FE1A0590)	_____

ACKNOWLEDGEMENT

The satisfaction that accompanies with the successful completion of any task would be incomplete without the mention of people whose ceaseless cooperation made it possible, whose constant guidance and encouragement crown all efforts with success.

We are grateful to **Mr. R. PRATHAP KUMAR**, Assistant professor, Department of Computer Science and Engineering for guiding through this project and for encouraging right from the beginning of the project till successful completion of the project. Every interaction with him was an inspiration.

We thank **Mr. T. V. VAMSI KRISHNA**, Assistant Professor & HOD, Department of Computer Science and Engineering for support and Valuable suggestions.

We also express our thanks to **Dr. K. PHANEENDRA KUMAR**, Principal, Vignan's Lara Institute of Technology & Science for providing the resources to carry out the project.

We also express our sincere thanks to our beloved **Chairman Dr. LAVU RATHAIAH** for providing support and stimulating environment for developing the project.

We also place our floral gratitude to all other teaching and lab technicians for their constant support and advice throughout the project.

Project Members

P. Ajay Babu (17FE1A05A0)

P. Mallikarjuna Reddy(15FE1A0597)

K. Sphoorthy(17FE1A0563)

N. Geetha Madhuri(17FE1A0590)

TABLE OF CONTENTS

ABSTRACT	i
LIST OF FIGURES	ii
LIST OF ABBREVIATIONS	iii-iv
CHAPTER I: INTRODUCTION	1-23
1.1 VANET	1
1.2 Architecture of the VANET	2-3
1.3 Routing Protocol in VANET	3-4
1.4 Characteristics of VANET	4-6
1.5 Routing Protocol	6-18
1.6 VANET Security	18-23
CHAPTER II: LITERATURE SURVEY	24-31
2.1 Literature Study	24-29
2.2 Existing System	30-31
2.2.1 Wormhole attack	30
2.2.2 Detection of Wormhole attack	30-31
CHAPTER III: METHODOLOGY	32-43
3.1 Introduction	32
3.2 Proposed System	32-33
3.3 Software Requirements	33-42
3.3.1 Introduction to Network Simulator	33
3.3.2 Uses of Network Simulator	34
3.3.3 Network Simulator	35
3.3.4 Basic Commands in NS2	36
3.3.5 NAM tool	36-38
3.3.6 Objects in NAM	39
3.3.7 Automatic Layout	40
3.3.8 The User Interface	40-41
3.3.9 AWK Script	41-42
3.4 System Implementation	42
3.4.1 Network Formation	42
3.4.2 Activation	42
3.4.3 Sharing Data	42

3.4.4 Communication	43
3.5 Hardware Requirements	43
CHAPTER IV: SYSTEM DESIGN	44-45
4.1 Data Flow Diagrams	44-45
CHAPTER V: TESTING	46-50
5.1 Testing in NS2	46
5.2 System Testing and Maintenance	46-48
5.3 System Study	49-50
CHAPTER VI: RESULTS	51-56
6.1 Simulation without Wormhole Nodes	51-53
6.2 Simulation with Wormhole Nodes	54-56
CHAPTER VII: CONCLUSION AND FUTURE SCOPE	57
REFERENCES	58-60
BIBLIOGRAPHY	61
URL'S	62
APPENDIX	63-73

ABSTRACT

Wormhole attack is one of the serious attack on the VANET's. A wormhole node is a malicious node that causes the packets to be incorrectly routed. A wormhole attack can be launched by one or several malicious nodes. Generally wormholes can be either external or internal but, in most of the cases internal wormholes are ignored. Here we put forward a algorithm that is capable of detecting both internal and external wormholes the algorithm is known as CREDND (Crediable Neighbour Discovery protocol). This algorithm can detect wormhole without requiring any additional hardware or high communication overheads.

LIST OF FIGURES

Figure No:	Name of the figure	Page No
Figure 1.1	Vehicular Ad-Hoc Networks	1
Figure 1.3.1	Taxonomy of Various Routing Protocols in VANET.	3
Figure 1.3.2	Three categories of VANET network architecture.	4
Figure 1.5.1	AODV route discovery.	7
Figure 1.5.2	Loops without using sequence number.	8
Figure 1.5.3	Loops with using sequence number.	9
Figure 1.5.4	Example of flooding of RREQ packets.	10-11
Figure 1.5.5	Transmission of data through packets.	13
Figure 1.5.6	The Path it had chosen for packet reply	15
Figure 1.5.7	Route from source to destination for data packet delivery.	16
Figure 1.6.2	Warm Hole Attack.	23
Figure 4.2	Flow chart for C++ and OTcl	34
Figure 4.2.2	Simplified user's view of NS	36
Figure 6.1.1	NAM Window	51
Figure 6.1.2	Sending RREQ from source to Destination	51
Figure 6.1.3	Transmission of packets from source to destination	53
Figure 6.1.4	Destination receiving packets	53
Figure 6.2.1	Sending Packets through wormhole tunnel	54
Figure 6.2.2	Destination receiving packets from wormhole tunnel	54
Figure 6.2.3	Dropping of Packets due to wormholes	55
Figure 6.2.4	Dropping and Sending Packets	55
Figure 6.3	Xgraph for output	56

LIST OF ABBREVIATIONS

AODV	Ad-hoc On Demand Distance Vector
ATM	Asynchronous Transfer Mode
AU	Application Unit
CBF	Control of Broadcast Forwarding
DDoS	Distributed Denial of Service
DoS	Denial of Service
DPRAODV	Detection, Prevention and Reactive Ad-hoc On Demand Distance Vector
DSN	Destination Sequence Number
DSR	Dynamic Source Routing
DSRC	Dedicated Short Range Communication
DTN	Delay and Disruption Tolerant Networks
FSR	Fisheye State Routing
GPS	Global Positioning System
IEEE	Institute of Electrical and Electronic Engineers
IP	Internet Protocol
IRA	Internet Routing Address
ITS	Intelligence Transportation System
LAN	Local Area Network
LAR	Location Aided Routing
MANET	Mobile Ad-hoc Networks
MOSAVODV	Modified Secured Ad-hoc On Demand Distance Vector
NS2	Network Simulator 2

NAM	Network Animator tool
OBU	On Board Unit
OSI	Open Systems Interconnections
PDR	Packet Delivery Ratio
RERR	Route Error
RREP	Route Reply
RREP_ACK	Route Reply Acknowledgment
RREQ	Route Request
RREQ_ACK	Route Request Acknowledgment
RSU	Road Side Unit
RPT	Round Trip Time
SAP	Systems, Applications, Products
TCP/IP	Transmission Control Protocol/Internet Protocol
TORA	Temporarily Ordered Routing Algorithm
UDP	User Datagram Protocol
WAN	Wide Area Networks
WAP	Wireless Application Protocol
Wi-Fi	Wireless Fidelity
WLAN	Wireless Local Area Networks
VANET	Vehicular Ad-hoc Networks
ZRP	Zone Routing Protocol

INTRODUCTION

1.1 VANET (Vehicular Ad-Hoc Network)

A Vehicular Ad-Hoc Network, or VANET, is a form of mobile ad-hoc network, to provide communications among nearby vehicles and between vehicles and nearest fixed equipment, usually described as roadside equipment. It is autonomous & self-organizing wireless communication network, where nodes in VANET involve themselves as servers and/or clients for exchanging & sharing information

The VANET used to providing safety and comfort for passenger. Having VANET inside vehicle need only small electronic device, which will provide Ad-Hoc Network connectivity for the passengers inside the vehicle. By this device operating this network does not need complicated connection and server communication. Each vehicle equipped with VANET device will be a node in the Ad-Hoc network and can receive and relay others messages through the wireless network and simple communication between vehicles and dynamic mobility.

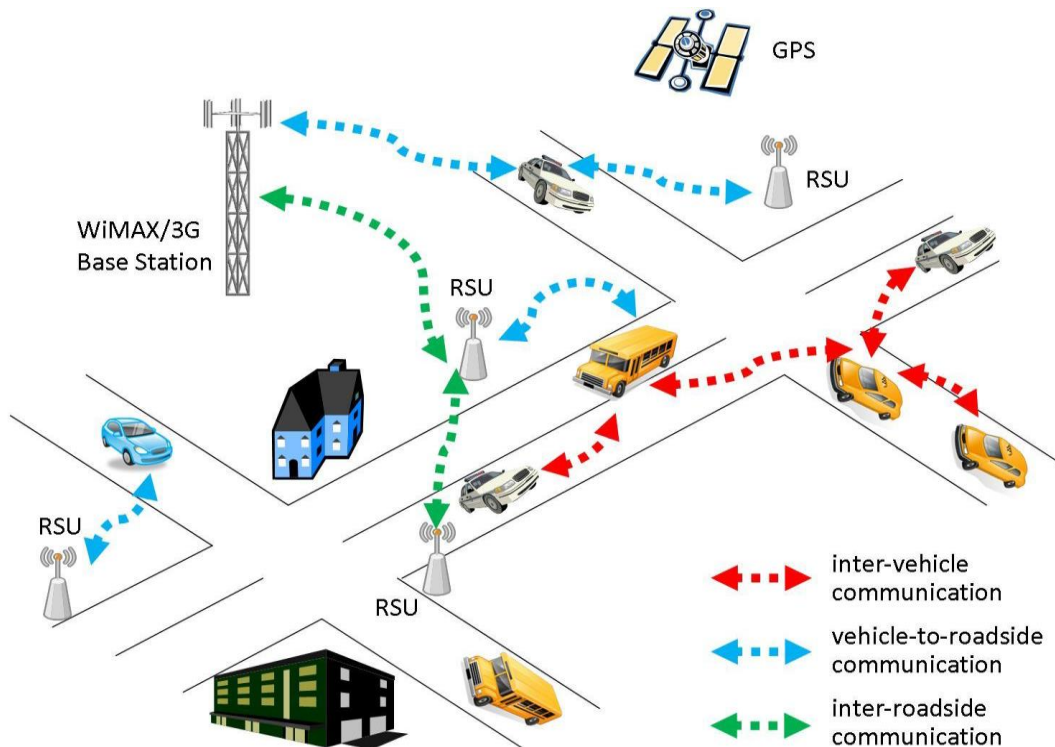


Fig 1.1: Vehicular Ad-Hoc Networks.

In vehicular Ad-Hoc network using different ad-hoc networking technologies such as Wi-Fi IEEE 802.11 b/g, WiMAX IEEE 802.16, Bluetooth, IRA, ZigBee for easy, accurate, effective.

1.2 Architecture of the VANET:

In general, protocol architecture achieves for communication among network nodes and provides the framework for implementation. When designing the communication suit for VANETs two approaches can be taken:

1.First, following the traditional approach, the overall functionality could be decomposed and organized in layers such that at the protocols fulfill small, well-defined tasks and form a protocol stack as in TCP/IP and OSI.

2.Second, one could try to build a customized solution that meets the requirements of VANETs.

The first approach—called **layered approach** and depicted attempts to retain the order of functions and protocol layers with well-defined interfaces between them. It adapts system functionalities to the needs of a VANET communication system resulting, e.g., in protocol layers for single-hop and multi-hop communication. The limitations and inflexibility of traditional network stacks when used in ad hoc networks are well known. E.g., each layer is implemented as an independent module with interfaces (SAPs) only to the above and below layers. Consequently, protocols cannot easily access state or metadata of a protocol on a different layer what makes data aggregation difficult. Moreover, some of VANET-specific functions do not fit into the traditional layered OSI model, such as those for network stability and control, and cannot be uniquely assigned to a certain layer. It is also worth noting that every layer accesses external information separately with no common interface which might lead to problems when this information influences protocol flow.

The second un-layered approach would be the result of tailoring a whole new system to the needs of VANETs' main focus, i.e., safety applications. Having accurate specifications of these applications and willing to use the probabilistic channel in the most efficient manner leads to have a highly coupled set of protocols. Therefore, all application and communication protocols are placed in one single logical block right over the physical interface and connected to the external sensors. Inside this block, all

protocol elements are modularized such that there are no restrictions for interaction, state information is arbitrary accessible. Note though, that this ‘architecture’ inherits a high design complexity due to arbitrary and complex interactions of their modules.

This makes protocol specification a complicated work and so, once designed becomes an extremely inflexible system for other types of application. Also it would be tough to systematically avoid control loop, what is rather easy in the layered approach with its clean top-down or bottom-up packet traversal. While both approaches would certainly be feasible.

1.3 Routing Protocols in Vehicular Ad Hoc Networks:

The routing protocols fall into two major categories of **topology-based** and **position-base** this project we mainly concentrate on one of the reactive(on-demand) routing protocol and it’s working.

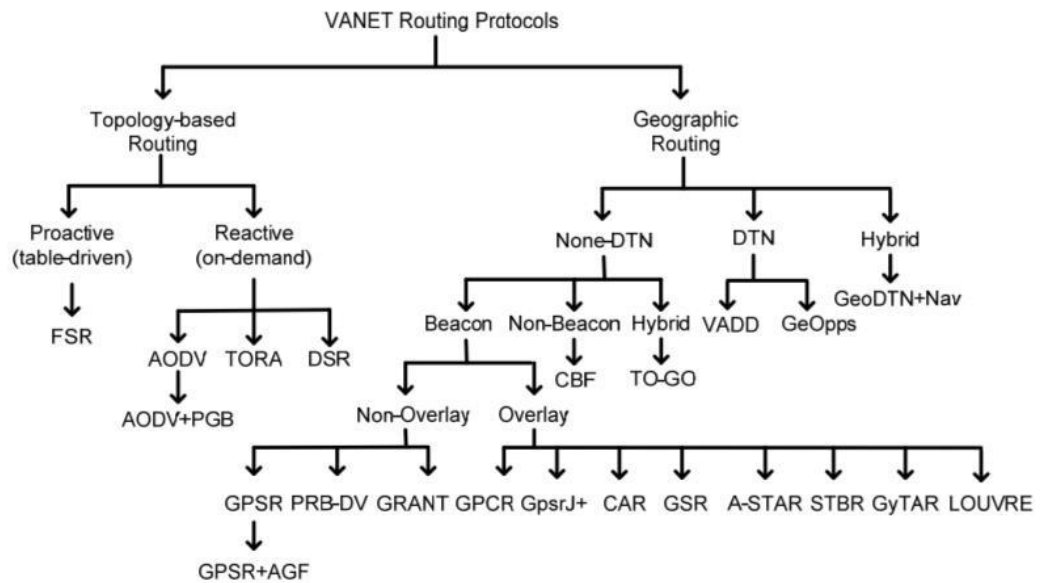


Fig1.3.1: Taxonomy of Various Routing Protocols in VANET.

In this project we mainly concentrate on one of the reactive(on-demand) routing protocol and it’s working. VANETs falls within three categories: pure cellular/WLAN, pure ad hoc, and hybrid. In pure cellular/WLAN architecture, the network uses cellular gateways and WLAN access points to connect to the Internet and facilitate vehicular applications. Vehicles communicate with the Internet by driving by either a cellular tower or a wireless access point.

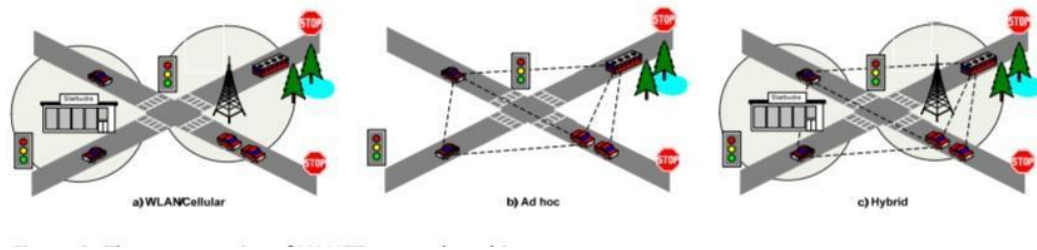


Fig 1.3.2: Three categories of VANET network architecture.

Since the infrastructure of cellular towers and wireless access points are not necessarily widely deployed due to costs or geographic limitations, nodes may only engage in communication with each other. Information collected from sensors on a vehicle can become valuable in notifying other vehicles about traffic condition and helping the police solve crimes (Lee, 2006). The infrastructure-less network architecture is in the pure ad hoc category where nodes perform vehicle-to-vehicle communication with each other. When there are roadside communication units such as a cellular tower and an access point and vehicles are equipped with wireless networking devices, vehicles can take advantage of the infrastructure in communicating with each other.

Various applications in areas of urban monitoring, safety, driving assistance, and entertainment (Lee, 2006) have used infrastructure communicating units to access dynamic and rich information outside their network context and share this information in a peer-to-peer fashion through ad hoc, infrastructure less communication. The hybrid architecture of cellular/WLAN and ad hoc approaches provides richer contents and greater flexibility in content sharing. Similar to mobile ad hoc networks (MANETs), nodes in VANETs self-organize and self-manage information in a distributed fashion without a centralized authority or a server dictating the communication. In this type of network, nodes engage themselves as servers and/or clients, thereby exchanging and sharing information like peers. Moreover, nodes are mobile, thus making data transmission less reliable and suboptimal.

1.4 Characteristics of VANET:

Highly dynamic topology:

Since vehicles are moving at high speed, the topology formed by VANETs is

always changing. On highways, vehicles are moving at the speed of 60 mph (25 m/sec). Suppose the radio range between two vehicles is 250 m. Then the link between the two vehicles lasts at most 10 sec.

Frequently disconnected network (Intermittent connectivity):

The highly dynamic topology results in frequently disconnected network since the link between two vehicles can quickly disappear while the two nodes are transmitting information. The problem is further exacerbated by heterogeneous node density where frequently travelled roads have more cars than non-frequently travelled roads. Moreover, (non) rush hours only result in disparate node density, thus disconnectivity. A robust routing protocol needs to recognize the frequent disconnectivity and provides an alternative link quickly to ensure uninterrupted communication.

Patterned Mobility:

Vehicles follow a certain mobility pattern that is a function of the underlying roads, the traffic lights, the speed limit, traffic condition, and driver's driving behaviours. Because of the particular mobility pattern, evaluation of VANET routing protocols only makes sense from traces obtained from the pattern. There are various VANET mobility trace generators developed for the very purpose of testing VANET routing protocols in simulation. Realistic mobility traces gathered from vehicles (Jetcheva, 2003) have also been gathered for the same purpose.

Propagation Model:

In VANETs, the propagation model is usually not assumed to be free space because of the presence of buildings, trees, and other vehicles. A VANET propagation model should well consider the effects of free standing objects as well as potential interference of wireless communication from other vehicles or widely deployed personal access points.

Unlimited Battery Power and Storage:

Nodes in VANETs are not subject to power and storage limitation as in sensor networks, another class of ad hoc networks where nodes are mostly static. Nodes are assumed to have ample energy and computing power. Therefore, optimizing duty cycle is not as relevant as it is in sensor networks.

On-board Sensors:

Nodes are assumed to be equipped with sensors to provide information useful

for routing purposes. Many VANET routing protocols have assumed the availability of GPS unit from on-board Navigation system. Location information from GPS unit and speed from speedometer provides good examples for plethora of information that can possibly be obtained by sensors to be utilized to enhance routing decisions.

1.5 Routing Protocol:

A routing protocol governs the way that two communication entities exchange information; it includes the procedure in establishing a route, decision in forwarding, and action in maintaining the route or recovering from routing failure. This section describes recent unicast routing protocols proposed in the literature where a single data packet is transported to the destination node without any duplication due to the overhead concern. Some of these routing protocols have been introduced in MANETs but have been used for comparison purposes or adapted to suit VANETs' unique characteristics. Because of the plethora of MANET routing protocols and surveys written on them, we will only restrict our attention to MANET routing protocols used in the VANET context. Figure 1 illustrates the taxonomy of these VANET routing protocols which can be classified as topology-based and geographic (position-based) in VANET.

Reactive (On Demand):

Reactive routing opens a route only when it is necessary for a node to communicate with another node. It maintains only the routes that are currently in use, thereby reducing the burden on the network. Reactive routings typically have a route discovery phase where query packets are flooded into the network in search of a path. The phase completes when a route is found.

DSR (Dynamic Source Routing):

In DSR uses source routing, that is, the source indicates in a data packet's the sequence of intermediate nodes on the routing path. In DSR, the query packet copies in its header the IDs of the intermediate nodes that it has traversed. The destination then retrieves the entire path from the query packet (a la source routing), and uses it to respond to the source. As a result, the source can establish a path to the destination. If we allow the destination to send multiple route replies, the source node may receive and store multiple routes from the destination. An alternative route can be used when

some link in the current route breaks. In a network with low mobility, this is advantageous over AODV since the alternative route can be tried before DSR initiates another flood for route discovery

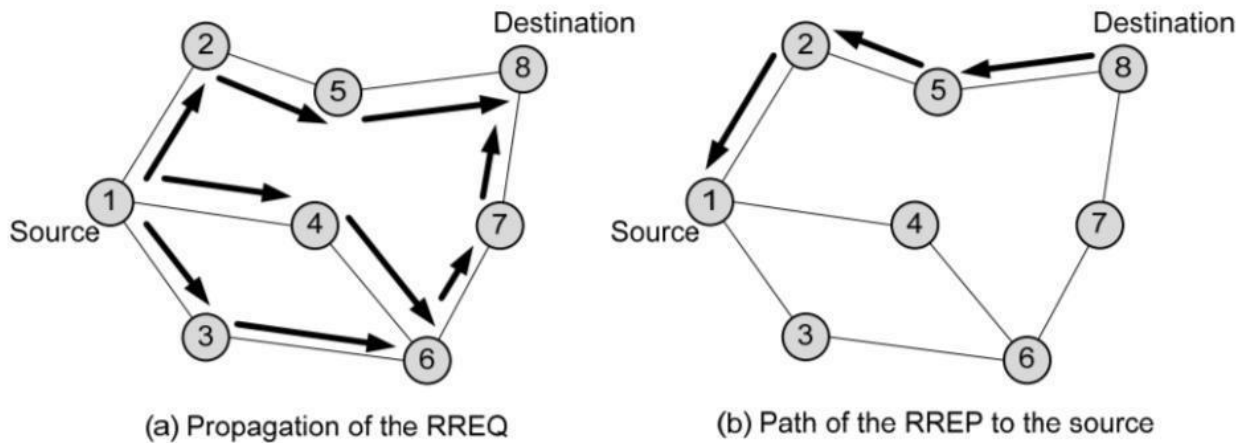


Fig 1.5.1: AODV route discovery.

AODV (Ad Hoc On Demand Distance Vector):

In AODV routing, upon receipt of a broadcast query (RREQ), nodes record the address of the node sending the query in their routing table. This procedure of recording its previous hop is called backward learning. Upon arriving at the destination, a reply packet (RREP) is then sent through the complete path obtained from backward learning to the source. At each stop of the path, the node would record its previous hop, thus establishing the forward path from the source. The flooding of query and sending of reply establish a full duplex path. After the path has been established, it is maintained as long as the source uses it. A link failure will be reported recursively to the source and will in turn trigger another query-response procedure to find a new route.

There are two major differences between AODV and DSR. The first is that in AODV data packets carry the destination address, whereas in DSR, data packets carry the full routing information. This means that DSR has potentially more routing overheads than AODV. Furthermore, as the network diameter increases, the amount of overhead in the data packet will continue to increase. The second difference is that in AODV, route reply packets carry the destination address and the sequence number, whereas, in DSR, route reply packets carry the address of each node along the route.

AODV Sequence number:

The sequence number is used as a way to deal with failed routes/links. It does not share the benefits of the approach used by DSR (I.e., including the known broken links in RREQ).

- Each node keeps a destination sequence number.
- The sequence number is only incremented.
- The larger the sequence number, the newer the route information to the destination.
 - Any routing information that has a higher sequence number both overrides the old routing information and invalidates the old route information.
 - E.g., if a node has a route to a node and then a RREQ arrives with a higher sequence number, then the old route is no longer valid.
 - Here we see the inability of AODV to precisely specify which link has broken and hence its inability to make use of topology that has been previously explored (i.e., explored with a smaller sequence number).
- Adjusting the sequence number
 - When a destination offers a new route it either increments its current sequence number or uses the sequence found in the RREQ, whichever is greater.
 - When a node receives a message with routing information that has a higher sequence number, then the node will take this new/higher value.
 - If a node learns that the path to the destination breaks, then a node will increment the sequence number (to invalidate all old top info).
 - If no sequence number for the destination is known, then it is set to unknown.

EX: -

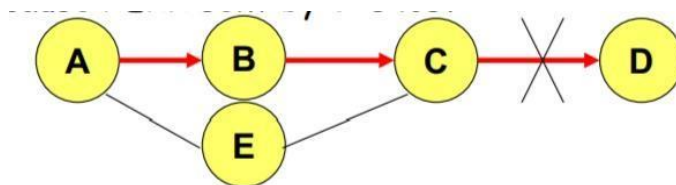


Fig 1.5.2: Loops without using sequence number.

To prevent formation of loops (without using sequence number):

1. A had a route to D initially

2. Assume that A does not know about failure of link C-D because RERR sent by C is lost.
3. Now C performs a route discovery for D. Node A receives the RREQ.
4. Node A will reply since A knows a route to D via node B.
5. Results in a loop (C-E-A-B-C).

By Using Sequence Number:

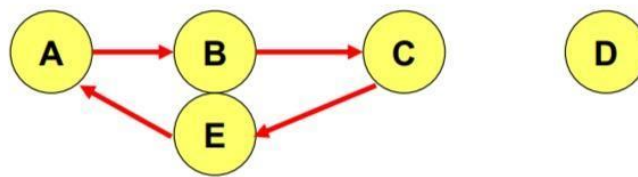


Fig 1.5.3: Loops with using sequence number.

1. Loop C-E-A-B-C.
2. But because of usage of sequence number, A will not use the route A-B-C. Because the sequence number will be lower than what A receives from A.

Routing Table

When a node receives a route message (e.g., RREQ or Reply), if no route currently exists a new entry is made

- If an entry already exists, it is updated only if
 - The message has a higher sequence number than this one is used in the entry.
 - The sequence number is the same, but the path (hops) is smaller.
 - The current sequence number is unknown (e.g., the entry was just made).
 - Entry timeout
- An entry will time out.
 - Every time a packet traverses the route, the lifetime is extended.
 - At the same time, the reverse path lifetime is also updated.
- Each entry also includes a list of precursors
 - A node is a precursor if there is an active path that goes to from the node to the destination.

- When a link fails, then the list of precursors can be notified and so on...

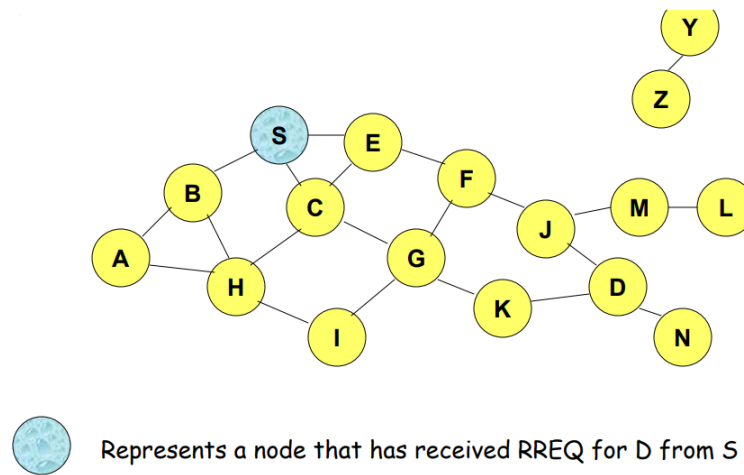
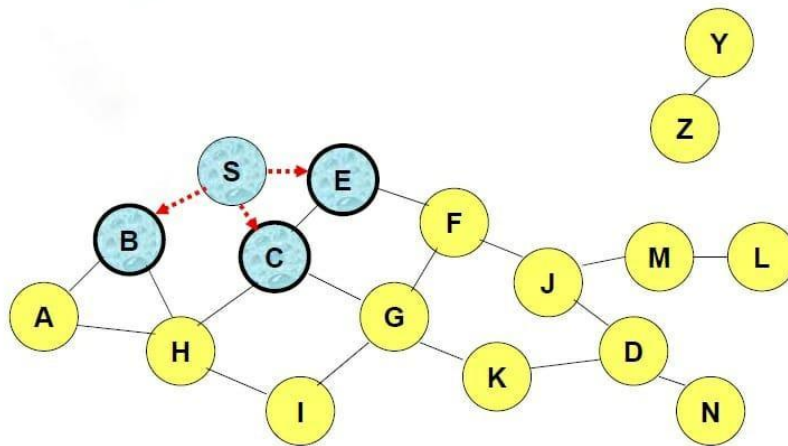


Fig 1.5.4: Flooding of RREQ packets.



-----> Represents transmission of RREQ.

Fig 1.5.4.1: Flooding of RREQ packets.

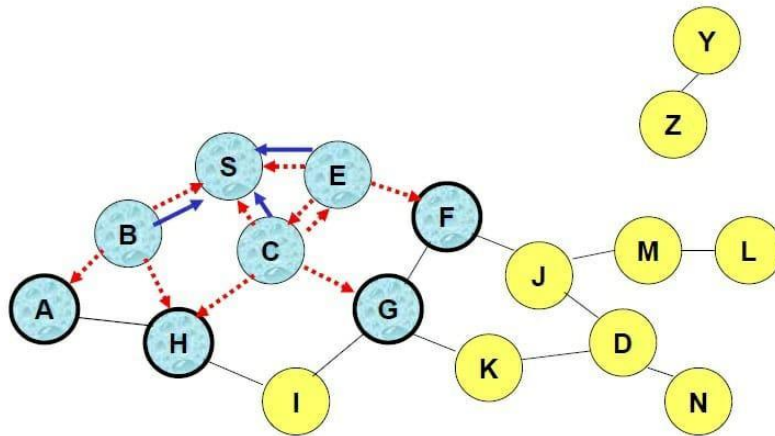
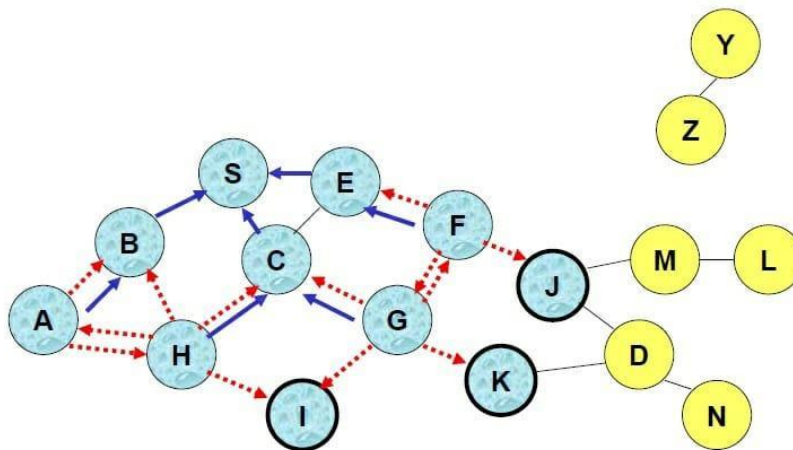


Fig 1.5.4.2: Flooding of RREQ packets.



<----- Represents links on Reverse Path.

Fig 1.5.4.3: Flooding of RREQ packets.

- Node C receives RREQ from G and H, but does not forward it again, because node C has already forwarded RREQ once.

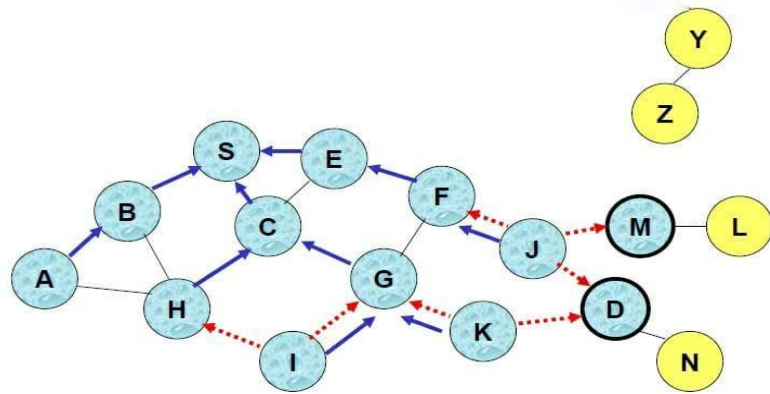


Fig 1.5.4.4: Flooding of RREQ packets.

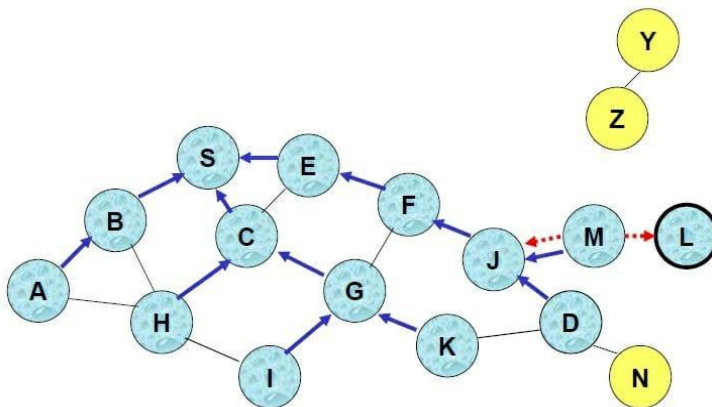
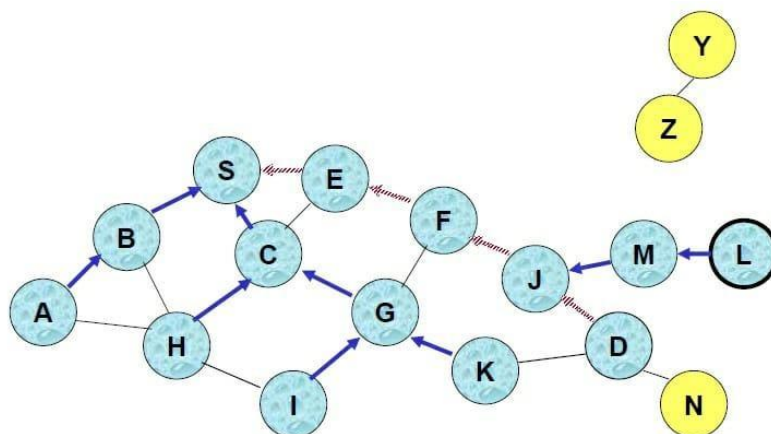


Fig 1.5.4.5: Flooding of RREQ packets.



<----- Represents link on path taken by RREP.

Fig 1.5.4.6: Flooding of RREQ packets.

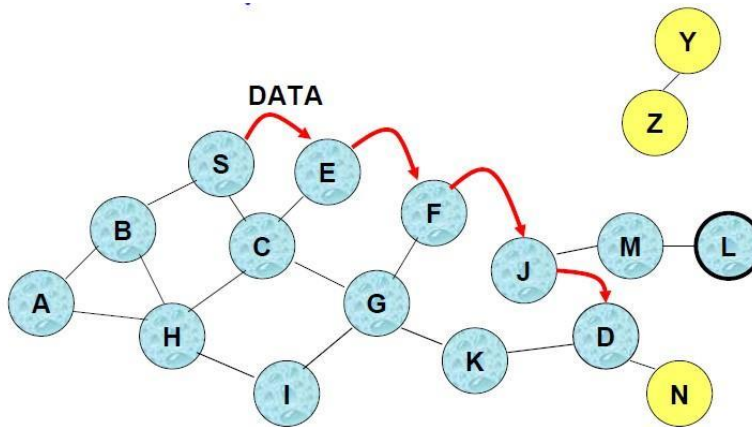


Fig 1.5.5: Transmission of data through packets.

- Forward links are setup when RREP travels along the reverse path.
- □ Red coloured arrow represents a link on the forward path.

Generating Route Request

- If a route is needed, e.g., if the current one timed out or the path has been found to have broken
- The last known sequence number is put into the RREQ.
- If none is known, then the destination sequence number is marked as invalid.
- Note, the destination sequence number is incremented when a link failure has been detected or timeouts.
- The sender/originating node includes its own sequence number.
- The number is incremented just before being put into the RREQ.
- This sequence number is then used by all neighbour nodes to update their cost...
 - This also acts to update the current path lifetime.
 - However, if the routing follows a funny path, this may result in increasing the hop count to the destination and perhaps could break some paths.

Processing Route Request

The route to the previous hop is updated/created.

- Next, it is checked if this RREQ has been previously observed, if so, the RREQ is dropped
 - This means that not even the return path is updated, so if a return path includes this node, it too has been updated.

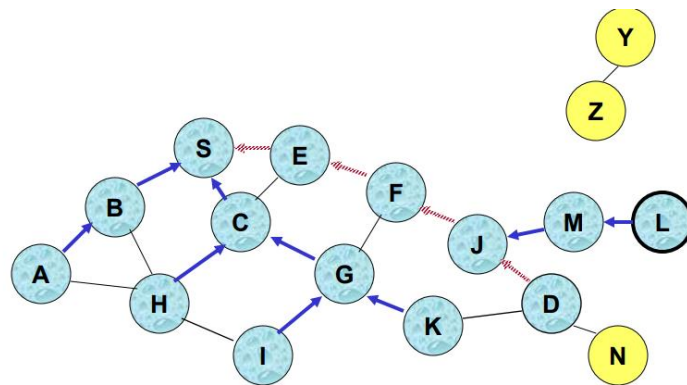
- However, downstream nodes (toward the destination) are not aware of this better path. Hence, not all paths are explored.
- On the other hand, shorter/better routes should be traversed sooner. But if a better route means one that will last longer, or has good power properties, then this approach will not work. It only makes sense if the metric is delay.
- Compare this to real distance vector and bellman.
- When a RREQ is received the return/reverse path entry is saved and perhaps updated specifically.
 - The sequence number is updated (if it is larger).
 - The sequence number is set to valid.
 - The next hop is set to the sender of the RREQ (previous hop).
 - The hop count is set.
 - The lifetime is set to be the max of the existing lifetime and $\text{currenttime} + 2 * \text{net_traversal_time} - 2 * \text{hop Count} * \text{Link_traversal_time}$.
 - If the TTL is zero or if a RREP is generated, the RREQ is dropped.
- Note that if a RREP is generated, the destination may never receive a RREQ and hence never build a return route to the source. In this case the destination will have to perform route discovery.
 - But the destination could observe the source of the packet and form a route accordingly. But this is not done. It is rather odd that no such route forming is done. For example, a table entry could also be formed by overhearing packets.
 - To solve the problem with the destination, the Gratuitous flag is set. In this case, if a RREP is generated, then a gratuitous RREP packet is sent over the known path to the destination.

Generating route reply (RREP)

If the node is the destination or if the node has an active route with sequence number that is greater or equal to the sequence number in the RREQ.

- If the node is the destination.
 - If the sequence number is the same as the sequence number in the RREQ, then the node increments its sequence number and puts that into the RREP, otherwise, it puts the current sequence number in the RREP
- If the node is an intermediate node
 - It leaves the sequence number unchanged

- The previous hop is placed in the precursor list for the destination
- The route to the source is updated by placing the next hop toward the destination as a precursor for the route to the source.
- On both cases, the hop count is set.




 Represents links on path taken by RREP

Fig1.5.6: The Path it had chosen for packet reply

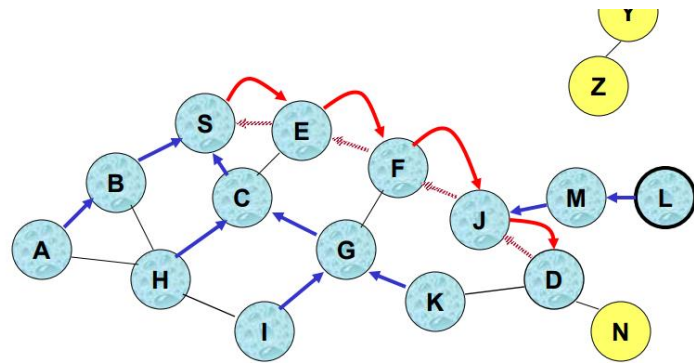
Processing a RREP

Upon receiving a RREP, a table entry is made/updated to the previous hop (the sender of the RREP), but the sequence number marked as invalid if the entry is created.

- The hop count in the RREP is incremented
- The table entry to the destination is updated only if
 - The sequence number in the table is marked as invalid.
 - The sequence number in the RREP is valid and is greater than the entry's.
 - The sequence numbers are the same but the route is marked as inactive.
 - The sequence numbers are the same but the hop count is smaller.
- If the entry is updated/created
 - The route is marked as active.
 - The sequence number is marked as valid.
 - The next hop is set to the sender of the RREP.
 - The hop count is set.
 - The expiry time is set from the RREP.
 - The destination sequence number is the same as in the RREP.

- The RREP is forwarded upstream and the precursor lists are updated.

Fig 1.5.7: Route from source to destination for data packet delivery



Route Error, Route Expiry and Route Deletion

- The routes that use the link are invalidated
- The destination that are impacted are determined.
 - The affected upstream neighbours are determined.
 - The upstream neighbours are informed of the breakage via a RERR (either via unicast or b-cast).
 - This continues until the sources are reached.
- Hence, a RERR is generated when
 - A transmission fails and hence a link break is detected and further that the route repair fails.
 - If a data packet arrives for a destination for which the node does not have a route and a route repair is not ongoing (I.e., the route had already broken or had already timeout) (note: there is a limit to the rate at which RERR packets can be generated – if a burst of packets arrives, then a burst of route error messages might not be generated).
 - If a RERR message arrives from a downstream node for an active route.
- Before a RERR is transmitted
 - The sequence number for the destination in the table entry is incremented, unless the RERR comes from a downstream node, in which case the sequence number in the RERR packet is used.
 - The entry is marked as invalid.
 - The lifetime of the entry is set to the DELETE_PERIOD. The entry is not deleted before this time.

- The idea of delete period is that you don't want a data packet to arrive with a destination that is not in the routing table.
- So the delete period is at least as large as the `active_route_timeout`. Thus, when the route is finally deleted, the upstream nodes will have marked the route as invalid.
- Note that if a data packet arrives for an invalid route, then the lifetime is reset to the `DELETE_PERIOD`.

Local Repair

- When a link break is detected, the upstream node may try to repair the link if the destination is less than `MAX_REPAIR_TTL` hops away.
- The TTL of the route search is
 - $\max(\text{hops to destination}, 0.5 * \text{Number hops to the source}) + \text{Local_ADD_TTL}$.
 - The idea is that the route search will try not to reach the source.
- During route search, packets are buffered.
- If no route is found, then a RERR is generated and any buffered packets are dropped.
- If a new route is found, then the buffered packets are sent over this new route.
- If the new route has a hop count that is longer than the old, now broken route.
 - A RERR is generated with the N bit set.
 - A node receiving a RERR with the N bit set does not delete the entry but merely forwards the RERR packet upstream.
 - When a source receives a RERR with the N bit set, it MAY begin a new route search.

Link Break Cont.

When a link breaks, routes are marked as invalid.

- These route may be repaired when a packet arrives for these routes. To support this, these routes are marked as repairable.
- If a route is marked as repairable, it remains so until it times out.

Flooding of Packets

Flooding is the static routing algorithm. In this algorithm, every incoming packet is sent on all outgoing lines except the line on which it has arrived. One major problem of this algorithm is that it generates a large number of duplicate packets on the network.

Several measures are takes to stop the duplication of packets. These are:

1. One solution is to include a hop counter in the header of each packet. This counter is decremented at each hop along the path. When this counter reaches zero the packet is discarded. Ideally, the hop counter should become zero at the destination hop, indicating that there are no more intermediate hops and destination is reached. This requires the knowledge of exact number of hops from a source to destination.
2. Another technique is to keep the track of the packets that have been flooded, to avoid sending them a second time. For this, the source router puts a sequence number in each packet it receives from its hosts. Each router then needs a list per source router telling which sequence numbers originating at that source have already been seen. If an incoming packet is on the list, it is not flooded.
3. Another solution is to use selective flooding. In selective flooding the routers do not send every incoming packet out on every output line. Instead packet is sent only on those lines which are approximately going in the right direction.

1.6 VANET SECURITY

VANET suffer from various attacks; these attacks are discussed in the following subsections:

A. ATTACKERS

- a. Greedy Drivers: Selfish drivers trying to maximize their gain by making believe a congested path to their destinations, and consequently suppress traffic by attacking the routing mechanisms.
- b. Snoops: Drivers attempting to profile drivers and extract their identifying information. Malicious Snoops can even track vehicle locations and determine the identities of drivers by corresponding them to the house or work sites.
- c. Pranksters: Drivers trying to disable applications or prevent information from reaching others vehicles. Such attacks are denoted by Denial of service attacks (DoS).
- d. Malicious Attackers: Drivers deliberately attempting to make harm via the available applications within the network. Several attacks focus on damaging exchanged data between vehicles such as message fabrication, suppression or alteration. Sybil attack (Masquerade) belongs also to this category.

- e. Industrial Insiders: If vehicle manufacturers are responsible for securing communications within VANETs, employees can reveal confidential data to malicious entities.

B. ATTACKER'S Properties:

Attacker create problem in the network by getting full access of communication medium DSRC. Here we are discussing some properties and capability of the attackers which has been mentioned in studies.

- a. Coverage Area: Attacker could cover the main area of road, and it depends on the nature of the attacks. Basic level attacker has controlled one DSRC channels and covers the range of at most 1000 meters but the extended level attackers are more organized and cover more area using of hundred DSRC channels.
- b. Technical Expertise: Technical expertise of the attacker makes them stronger for creating attacks in the network. Attacker having ability to extracts the program code and secret keys of the computing platform of OBU and RSU by launching physical attacks.
- c. Resources: Budget, manpower and tools are the three main key resources and attackers depend on it to achieve their goals.

Security attacks in VANET:

The connection between nodes is mostly established wirelessly in VANET. This increases the chance of security attacks as every node has the access to other nodes in a network. There are various types of attack that could be incurred in a network. If the security attack would be successful, then the connection between the nodes could be disrupted (in many cases) or the nodes would be falsified. It could lead to severe accidents. The types of attacks in VANET are as follows:

1. Sybil attack

In Sybil attack, the attacker node has more than one fake identity which results in getting access to the classified information and disrupting the communication between nodes. In case of VANET, it could lead to fake traffic jams which would cause other nodes to take alternative route.

2. Impersonation Attack

The word “impersonation” means the act to do something by being someone else basically to gain information or to do fraudulent activities. In VANET, node impersonation attack means the attacker node sends the wrong information to another node by changing its identity which can cause accidents and the attacker node can escape away without getting identified.

3. DOS attack

If the DOS attack is implemented in any server then the authentic user cannot access the contents of the targeted system. In VANET, the attacker jams all the medium of communication which results in no availability of network to authentic users.

4. DDOS attack

The aim of the attacker which use Distributed denial of service (DDOS) attack is same as that of DOS attack but the approach is different. In DDOS attack, the attacker attempts to attack from different locations in a distributed manner and the attacker uses different time to send the message.

5. Black hole attack

Black hole attack is also known as packet drop attack. As the name suggests, the attacker node attracts other nodes to send the packets through it but it drops the packet instead of forwarding it. Black hole attack has been demonstrated in this project and has been discussed again in the malicious node.

6. Wormhole attack

In wormhole attack, the attacker node captures the packets in a network at a particular location and then tunnels the packets to different location and retransmits. In case of VANET, the shortest route cannot be found out using protocol if wormhole attack is performed in a network.

Security requirements in Vanet:

- 1. Message Authentication:** All vehicles and road side units should be properly authenticated for sending notifications to other vehicles
- 2. Traceability:** Ability to obtain vehicle real identity, so that it can be traced and charged for navigation services.

- 3. Identity preservation:** Real identity of vehicle should be kept private from other vehicles and roadside equipment.
- 4. Confidentiality:** Information and queries of other vehicles should be kept private; such as navigation results.
- 5. Non-Repudiation:** In this scenario, a roadside vehicle cannot deny that it has transmitted notification, information must be crucial in it.
- 6. Congestion and collision control:** In this case, due to lack of poor communication between vehicles and RSUs (Road side units), a traffic scenario will be created and resulting in accidents, collisions and congestion problem.

Security Challenges in VANET:

- 1. Entity Authentication:** The receiver isn't solely ensured that the sender generated a message, however additionally has evidence of the liveness of the sender.
- 2. Access Control:** Access to specific services provided by the infrastructure nodes, or different nodes, is decided locally by police. As a part of access management, authorization establishes what every node is allowed to try and do in VANET.
- 3. Availability:** The network and applications ought to stay operational even within the presence of faults or malicious conditions. This means not solely secure however additionally fault-tolerant styles, resilience to resource depletion attacks, further as survivable protocols, that resume their traditional operations when the removal of the faulty participants.
- 4. Privacy and Anonymity:** Conditional privacy should be achieved within the sense that the user connected info, as well as the driver's name, the license plate, speed, position, and traveling routes at the side of their relationships, has got to be Protected: whereas the authorities ought to be ready to reveal the identities of message senders within the case of a dispute like a crime/car accident scene investigation, which may be accustomed hunt for witnesses.
- 5. Liability Identification:** Users of vehicles are liable for their deliberate or accidental actions that disrupt the operation of other nodes, or the transportation system. Several attacks are known which will be classified depending on the layer

the attacker uses. At the physical layer and link layers the attacker will disturb the system either by jamming or overloading the channel with messages. Flooding false messages or rebroadcasting a recent message is also an attainable attack. Message fabrication, alteration, and replay can all be used towards impersonation.

- 6. Mobility:** The basic idea from Ad Hoc Networks is that each node in the network is mobile, and can move from one place to another within the coverage area, but still the mobility is limited, in Vehicular Ad Hoc Networks nodes moving in high mobility, vehicles make connection throw their way with another vehicle that maybe never faced before, and this connection lasts for only few seconds as each vehicle goes in its direction, and these two vehicles may never meet again. So securing mobility challenge is hard problem.
- 7. Volatility:** The connectivity among nodes can be highly ephemeral, and maybe will not happen again, vehicles travelling throw coverage area and making connection with other vehicles, these connections will be lost as each car has a high mobility, and maybe will travel in opposite direction. Vehicular networks lack the relatively long life context, so personal contact of user's device to a hot spot will require long life password and this will be impractical for securing VC.

Wormhole Attack

In wormhole attack, the attacker node captures the packets in a network at a particular location and then tunnels the packets to different location and retransmits. In case of VANET, the shortest route cannot be found out using protocol if wormhole attack is performed in a network. In this attack, an adversary receives packets at one point in the network, tunnels them to another point in the network, and then replays them into the network from that point. This tunnel between two adversaries are called wormhole. It can be established through a single long-range wireless link or a wired link between the two adversaries. Hence it is simple for the adversary to make the tunnelled packet arrive sooner than other packets transmitted over a normal multi-hop route.

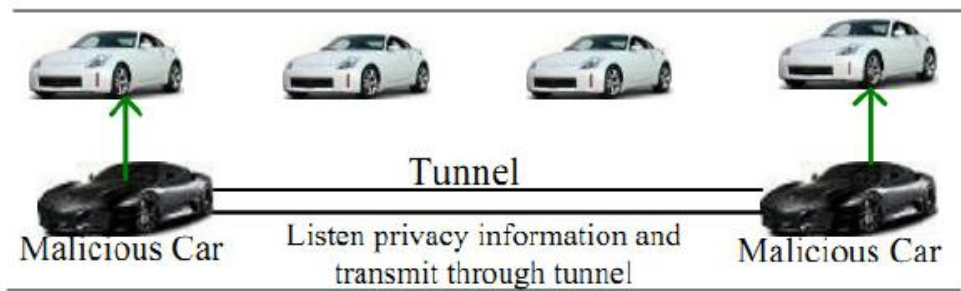


Fig 1.6.2: Wormhole Attack.

CHAPTER-II

LITERATURE SURVEY

2.1 LITERATURE STUDY

The research community in VANET has proposed many solutions to overcome the security challenges of routing protocols caused by various attacks. Some of them are described as follows:

The author Yi et al. presented “a Security-aware ad hoc routing for wireless networks” and examined unauthorized and malicious nodes and represented flaws in the security aspects of ad-hoc network communication.

In Authenticated Routing protocol for ad hoc Networks (ARAN) Sanzgiri et al. presented “A secure routing protocol for Ad hoc networks” and proposed a method to resolve the security issues based on cryptographic public-key certificates. ARAN is efficient in maintaining and discovering the route but used larger packets resulting to overall higher routing overhead.

Secure Efficient Ad hoc Distance Vector Routing Protocol (SEAD) was proposed by Hu et al. presented “SEAD: secure efficient distance vector routing for mobile wireless Ad Hoc networks”. It was based on hash chain sequences to authenticate hop counts between nodes. The security features in Distance Sequence Distance Vector (DSDV) protocol was enhanced by the sequence numbers. The packet delivery ratio of SEAD is better than DSDV. But due to increase in number of routing advertisement network overhead is also increased.

“A secure on- demand routing protocol for Ad Hoc networks” was proposed by Ariadne Perrig. The algorithm was based on Dynamic Source Routing (DSR). It shared the secret key between two nodes. Though these distributed and independent developments have provided an analysis of network security features, still there is a lack in protocol standards.

A novel mechanism was proposed by Shurman et al. proposed “Black hole attack in mobile Ad Hoc networks” in which the source node has computational capabilities to verify the authenticity of the node initiating the RREP messages. The possible paths to the destination could be identified by the node and compute the safest route to the destinations. The routing delays are increased in this method. Performance

analysis of ad-hoc networks under black hole attacks

Arya et al. presented “Detecting and avoiding of worm hole attack and collaborative black hole attack on manet using trusted aodv routing algorithm” look at a reputation-based approach which adds a trust model on top of the AODV routing protocol. Their trust function estimates the reliability of nodes during route discovery and assigns a trust-status of either ‘Unreliable’, ‘Reliable’, or ‘Most Reliable’ to each node. The trust-status has bearing on the decision of which route is selected; instead of the normal procedure of automatically selecting the shortest route. Their simulation results show greater throughput and Packet Delivery Ratio (PDR) with Trusted AODV than AODV during worm hole and black hole attacks.

Dave and Dave presented “an effective black hole attack detection mechanism using permutation based acknowledgement in manet” and proposed an acknowledgement-based detection mechanism for black hole attacks in MANETs using Permutation based Acknowledgement. Their solution improves on previous acknowledgement-based solutions (namely the Adaptive Acknowledgement and TWO-ACK) with a reduced overhead. Routing in this solution uses a secure multipath variant of AODV. While their detection method is effective, the overhead associated with this solution when there are few paths between source and destination nodes would likely make it unsuitable for use in VANETs. The routing overhead further increases as the node speed increases. Overhead issues such as these typically make acknowledgement-based solutions unsuitable for the vehicular wireless network domain.

Zapata published a secure, cryptographic-based routing protocol, which he named “Secure Ad Hoc On-Demand Distance Vector (SAODV) in Secure ad hoc on-demand distance vector routing”. Seen as an extension to the AODV routing protocol in Ad hoc on-demand distance vector (aodv) routing, it was designed with tackling the challenge of securing MANETs in mind. The routing protocol offers authentication, integration and non-repudiation – features black hole attack countermeasures are often designed around. SAODV has the benefit of a malicious node only being able to lie about itself so if advertising a route, for example, the route must exist (due to hash chaining) and the advertised number of hops can be at most one less than the actual number of hops (due to individual incrementation and authentication of the hop counter by intermediate nodes).

In a cooperative black hole attack with multiple malicious nodes, the advertised

number of hops could be less however. The drawback of SAODV though is that it does nothing to prevent the Denial-of-Service aspect of a black hole attack in Secure Ad hoc On-Demand Distance Vector (SAODV) Routing and as such it does not present a viable solution by itself. This was re-iterated by Von Mulert et al. in “Security threats and solutions in manets: A case study using aodv and saodv” in 2012.

Roshan et al. have presented a routing strategy to detect and prevent malicious nodes in Detection of malicious node and development of routing strategy in VANET, the idea of the proposed strategy is based on double acknowledgement packet which means every intermediate node has to inform the source node that it has sent the packet forward. This process ends when the destination is reached. This method adds heavy overhead in the network and extra delay.

In Detection of Intelligent Malicious and Selfish Nodes in VANET using Threshold Adaptive Control, Chaker et al. proposed a mechanism for the detection of selfish and intelligent malicious nodes using threshold adaptive control. However, direct and indirect trust are computed based on the number of malicious and legal actions. Direct trust is calculated between a specific node and its neighbour. In the other hand, indirect trust is calculated based on the recommendation from one hop neighbours about other vehicles. But, in the presence of collaborative Black Hole attack this mechanism fails.

P.S. Hiremath et al. proposed “an adaptive system of fuzzy inference to detect and prevent the Black Hole attack”. In Adaptive Fuzzy Inference System for Detection and Prevention of Cooperative Black Hole Attack in MANETs, four inputs used for the Fuzzy Inference System (FIS): data, trust, data rate, data loss, and energy (characterize the quality of next hop neighbourhood). This information is sent periodically by each node to update neighbour information. The system of fuzzy inference is used in the step of selecting of the next hop neighbour. This strategy is compared to an adaptive method to Adaptive Method for Detection and Prevention of Cooperative Black Hole Attack in MANETs. The new proposed strategy shows a better performance in the simulation results.

Sagar R Deshmukh et al. proposed “an AODV-based secured routing to detect and prevent single and cooperative black hole attacks in AODV-Based Secure Routing Against Black Hole Attack in MANET”, the idea of the authors is keeping the basic mechanism of AODV unchanged and just attach a validity value to the RREP. The

simulation results show a good performance against the Black Hole attack with negligible overheads compared to the fundamental AODV. However, in the presence of an intelligent adaptive Black Hole in the network, this strategy falls flat, hence, an intelligent malicious node could easily set the validity value in the same way in which it claims that it has the freshest and the shortest route to a target node.

Otero et al. implemented two procedures which are based on range-free localization methods for the detection of wormhole. In the proposed work, first procedure performed the detection process simultaneously by using the localization procedure to find the node's position and the second procedure followed the post-localization detection approach to validate the estimate node position and detected the malicious nodes.

Fatehpuria et al. presented "wormhole attack prevention by verification of digital signatures". The proposed work has identified two types of wormhole attacks: hidden and exposed attacks by using a mechanism delay per hop indication that detects the pinpoint location of wormhole and prevent it. It has followed two phases to identify the attacks. In the first phase, verification of digital signature is done to depict the presence of wormhole or not and in the second phase, delay/hop is analyzed using RTT mechanism.

Harikishan et al. proposed "a unique approach called IDS with use of Fuzzy inference system to encounter to the intrusion behaviour within the network". Using Sugeno Fuzzy Inference approach and ANFIS editor, an accurate attack was detected. In proposed work, MATLAB and ANFIS editor is used for experimentation and KDD CUP dataset is used for detecting the anomaly based intrusion with the use of fuzzy inference approach.

Upadhyay described "WPAODV technology for identify and prevent any wormhole". The WPAODV is depending on the hybrid model that encapsulates the proper area, neighbour node and hop count approach. In the planned scheme, WPAODV extends the AODV by adding one extra feature in it i.e. after detection; WPAODV will bypass the path that is having a wormhole. To detect whether the route is having a wormhole or not, the WPAODV used divide and conquer mechanism over the route recommended by AODV. The main objective of the planned scheme is to detect the wormhole in the route recommended by AODV.

Shamaei et al. presented "a two-phase detection mechanism to detect and

prevent the wormhole attacks”. In the first phase, it checks whether tunnel exists or not on the selected path by calculating the average delay per hop. If the average delay per hop is higher than the predefined threshold then the wormhole existence can be imagined and in the second phase it depicts the existence of wormhole attacks and discovers the malicious node. The proposed scheme has the capability to detect in band and out band attacks without need of any addition H/W. Biswas implemented WADP i.e. Wormhole attack detects and prevents approach by altering the AODV routing protocol. The modification in AODV is done by doing addition of two extra fields in the route reply packet that is the IP of intermediate node and the unique number assigned to it. Assumption is made in the proposed work i.e. only authentic nodes know the information. If the node does not able to specify the right IP and number, then it will be considered as malicious node. WADP is an enhancement of WAP algorithm. False positives are removed in the WADP algorithm by using the node authentication process and it also helps to know the exact location of wormhole node. Thus this process is considered as a double verification for the detection of wormhole attack.

Aggarwal et al. proposed “a beacon node mechanism with neighbour node discovery for the identify and counter of wormhole attack”. Defined technique detects any wormhole by the use of deviation in routing information among neighbours and it does not require any location information and additional hardware. The main goal of the defined method is to detect the wormhole in the route recommended for the AODV with the use of divide and conquer technique that will make all possible combinations of nodes. If any combination comes out to be suspicious, it will be considered as malicious.

In “a static-node assisted adaptive routing protocol in vehicular networks” presented by Savas Konur and Michael Fisher, Vehicular ad hoc networks (VANETs), which are a class of Vehicular ad hoc networks, have recently created a standard method for correspondence among moving vehicles. Since VANETs are imperative to the wellbeing and safety of the vehicles, the people and the infrastructure, a profound analysis of their potential behaviour is obviously required. In this paper they give this analysis using formal verification. In particular, they formally examine a particular congestion control protocol for VANETs utilizing a probabilistic model checking procedure, and researching its effectiveness and adequacy.

In “Proceedings of the Fourth ACM international Workshop on Vehicular Ad

Hoc Networks VANET” presented by Trishita Ghosh and Sulata Mitra, the remote access in vehicular environment system is created for upgrading the driving security and comfort of car users. In any case, such framework endures from quality of service degradation for security applications brought on by the direct congestion in scenarios with high vehicle density. The work is a congestion control technique in which vehicular networks are safe from congestion. It supports the correspondence of protected and unprotected messages among vehicles and infrastructure.

Several approaches have been developed about detecting and avoiding wormhole attacks in wireless ad hoc network. “In Detecting and Avoiding Wormhole Attacks in Wireless Ad Hoc Networks”, Naït-Abdesselam and his co-workers devised an efficient method to detect and avoid wormhole attacks in the OLSR protocol. This method first attempts to pinpoint links that may potentially be part of a wormhole tunnel. Then a proper wormhole detection mechanism is applied to suspicious links by means of an exchange of encrypted probing packets between the two supposed neighbours (endpoints of the wormhole). The proposed solution exhibits several advantages, among which are its non-reliance on any time synchronization or location information, and its high detection rate under various scenarios.

Also in related work of “Detecting and Avoiding Wormhole Attacks in Wireless Ad Hoc Networks” other methods which have been presented previously are Explained summarily. In Using Directional Antennas to Prevent Wormhole Attacks proposed by L. Hu and D. Evans, directional antennas are used to prevent against wormhole attacks. Each node in the network shares a secret key with every other node and broadcasts HELLO messages to discover its neighbours using directional antennas in each direction.

“In Detecting and Locating Wormhole Attacks in Wireless Ad Hoc Networks through Statistical Analysis of Multipath” by L. Qian, N. Song proposed a statistical approach based on multipath routing is proposed. This approach uses the relative frequency of each link when discovering routes within the network. The main idea beneath this approach resides in the fact that the relative frequency of a link which is part of a wormhole tunnel is much higher than other normal links.

“In DelPHI: Wormhole Detection Mechanism for Ad Hoc Wireless Networks” presented by H.S. Chiu and K.S. Lui, the proposed Delphi protocol allows a sender to observe the delays associated with the different paths to a receiver. Therefore, a sender

can check whether there are any malicious nodes sitting along its paths to a receiver trying to launch wormhole attacks. The obtained delays and hop count information of some disjoint paths are used to decide whether a certain path among these disjoint paths is under a wormhole attack.

2.2 Existing system

2.2.1 Wormhole attack:

Wormhole is a severe attack in VANETs and other ad hoc networks that could be considered as a variation of Black Hole attack. In this attack, two or more malicious nodes create a tunnel to transmit data packets from one end to the malicious node at the other end and these packets are broadcasted to the network. Owing to the nature of wireless transmission, a malicious node is capable creating a wormhole even for packets not addressed to it, simply by overhearing them in wireless environment and then tunnelling them to the colluding node at the other end of the wormhole. The wormhole allows the attacker getting a very dominant role in comparison to other nodes, and it can exploit this position in a variety of ways, for example, to gain unauthorized access, disrupt routing, or perform a Denial of Service (DoS) attack, thus, threaten the security of transmitting data packets.

Wormhole attacks disrupt the multicast and broadcast operations for transmitting messages in VANETS, particularly in on-demand routing protocols such as AODV (Ad hoc On- demand authentication and protection mechanisms for routing packets and thus, is affected by wormhole attacks. The malicious nodes or wormholes can gain unauthorized access to perform Denial of Service (DoS) attacks.

2.2.2 Detection of wormhole attack:

In our routing algorithm we need to identify wormhole attackers by detecting the link lifetime of two nodes, route reliability and route redundancy from source to destination. Here we identify the link lifetime because nodes in the VANET moves at high speed their communication gets broken if a node moves out of coverage. Initially sender nodes discover routes to find destination by broadcasting RREQ packets to its 1-hop neighbours. If a destination is identified, it sends RREP to sender node on specific path based on first received RREQ packet. Here when source node broadcasts packet, it starts to calculate Round Trip Time (RTT). Destination node sends RREP to Sender node within some given time. RTT is calculated for every possible route from

source to destination.

After getting possible routes, we need to calculate link lifetime between two node. Source starts to count RTT is calculated for each route when RREQ is send through next 1-hop members. RREP is received to source at specific time; source stamps calculated RTT of its routes or route. Finally, all routes with link lifetime are listed with number of hops and RTT.

CHAPTER III

METHODOLOGY

3.1 INTRODUCTION

Wormhole attack is a grave attack in which two attackers locate themselves strategically in the network. Then the attackers keep on listening to the network, and record the wireless information. A Wormhole attack can easily be launched by the attacker without having knowledge of the network.

Detection of Wormhole attacks can be possible by using CREDND algorithm and it stands for Credible Neighbor Discovery. This algorithm can detect not only external wormholes through the hop difference between the own exclusive neighbors, but also internal wormholes through enabling the common neighbor nodes as witnesses to monitor whether the authentication packets are forwarded by malicious nodes.

3.2 PROPOSED SYSTEM

In view of the shortcomings of current approaches, we propose a novel Credible Neighbor Discovery algorithm named CREDND for wormhole detection, which can not only detect wormholes without additional hardware, but also detect wormholes whatever type the wormhole is. CREDND achieves better performance and less energy consumption. The main contributions of this paper are summarized as follows.

- We propose CREDND, a novel secure neighbor discovery algorithm for wormhole detection, using hop difference and local monitoring. CREDND is able to detect both external wormholes and internal wormholes. It improves the ability of wormhole attack defense, and saves node energy at the same time.
- We also propose the concept of neighbor ratio threshold, which avoids to perform wormhole detections for all nodes in WSN, contributing to improving the accuracy of wormhole detection and saving energy.
- We test the accuracy of CREDND by simulating and comparing it with two other wormhole detection algorithms, SECUND [45] and SEDINE [46], who also use hop difference and local monitoring. The results demonstrate that our approach outperforms existing approaches

Algorithm

Input: The entire network W with nodes O and their neighbor set N , and Neighbor Ratio Threshold r

Output: The node pairs who need to be detected.

for each node $o(i)$ in O and its neighbor set N_i in N do

Let $n_i = |N_i|$ (the neighbor number of o_i);

for each node $o_j \in N_i$ do

$n_j = |N_j|$ (the neighbor number of o_j);

Initialize $s = 0$;

$s = s + n_j$;

Calculate the average neighbor number of o_i 's

neighbors $\bar{n}_i = s/n_i$;

Calculate o_i 's neighbor ratio $r_i = n_i/\bar{n}_i$;

if $r_i > r$ then

Add $o(i)$ to suspected nodes set S ; for each nodes $s_i \in S$ do

for each nodes $s_j \in S$ do

if s_i and s_j are neighbor then

Perform external wormhole detection

3.3 Software Requirements:

Operating System: Ubuntu

Tools : NS2, NAM

3.3.1 Introduction to the NETWORK SIMULATOR:

A network simulator is a software program that imitates the working of a computer network. In simulators, the computer network is typically modelled with devices, traffic etc. and the performance is analyzed. Typically, users can then customize the simulator to fulfill their specific analysis needs. Simulators typically come with support for the most popular protocols in the use today, such as Wireless LAN, Wi-Max, UDP, and TCP. A network simulator is a piece of software or hardware that predicts the behaviour of a network, without an actual network being present. NS

is an object oriented simulator, written in C++, with an OTcl interpreter as a frontend.

The simulator supports a class hierarchy in C++ and a similar class hierarchy within the OTcl interpreter. The two hierarchies are closely related to each other; from the user's perspective, there is one-to-one correspondence between a class in the interpreted hierarchy and one in the compiled hierarchy. The root of this hierarchy is the class Tcl object.

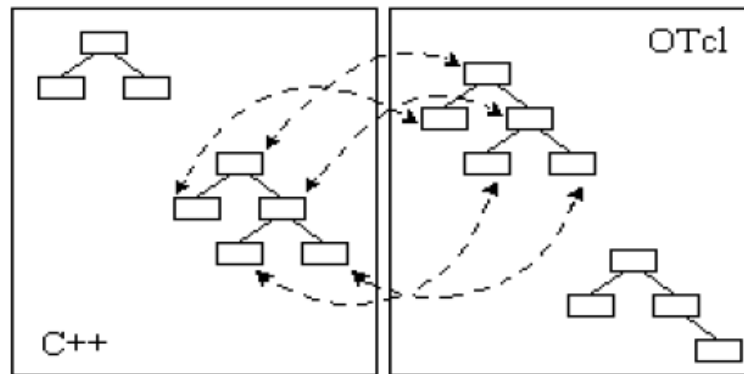


Figure 4.2. Flow chart for C++ and OTcl

Users create a new simulator objects through the interpreter; these objects are instantiated within the hierarchy. The interpreted class hierarchy is automatically established through methods defined in the class Tcl object. There are other hierarchies in the C++ code and OTcl scripts; these other hierarchies are not mirrored in the manner of Tcl object.

3.3.2 USES OF NETWORK SIMULATORS

Network simulators serve a variety of needs. Compared to the cost and time involved in setting up an entire test bed containing multiple networked computers, routers and data links, network simulators are relatively fast and inexpensive. They allow engineers to test scenarios that might be particularly difficult or expensive to emulate using real hardware- for instance, simulating the effects of sudden bursts in the traffic or a Dos attack on a network service. Networking simulators are particularly useful in allowing designers to test new networking protocols or changed to existing protocols in a controlled and reproducible environment. Network simulators simulate and then analyse the effect of various parameters on the network performance.

Typical network simulators encompass a wide range of networking technologies and help the users to build complex networks from basic building blocks

like variety of nodes and links. With the help of simulators one can design hierarchical networks using various types of nodes like computers, hubs, bridges, routers, optical cross-connects, multicast routers, mobile units, etc. various types of Wide Area Network (WAN) technologies like TCP, ATM, IP etc. and Local Area Network (LAN) technologies like Ethernet, token rings etc. all can be simulated with the typical simulator and the user can test, analyse various routing etc. There are a wide variety of network simulators, ranging from the very simple to very complex. Minimally a network simulator must a user to represent a network topology, specifying the nodes of the network, the links between the nodes and the traffic between the nodes.

More complicated systems may allow the user to specify everything about the protocols used to handle network traffic. Graphical applications allow users to easily visualize the working of their simulated environment. Text based applications may provide a less intuitive interface, but may permit more advanced forms of customization. Others, such as GTNets, are programming- oriented, providing a programming framework that the user then customizes to create an application that simulates the networking environment to be tested.

3.3.3 NETWORK SIMULATOR 2 (NS2)

NS2 is an open- source simulation tool that runs on Linux. It is a discreet event simulator targeted at networking research and provides substantial support for simulation of routing, multicast protocols and IP protocols, such as UDP, TCP over wired and wireless (local and satellite) networks. It has many advantages that make it useful tool, such as support for multiple protocols and the capability of graphically detailing network traffic. Additionally, NS2 supports several algorithms in routing and queuing. Queuing algorithms include fair queuing, deficit round-robin and FIFO. REAL is a network simulator originally intended for studying the dynamic behavior of flow and congestion control schemes in packet switched data network. NS2 is available on several platforms such as FreeBSD, Linux, Sim OS and Solaris. NS2 also builds and runs under Windows.

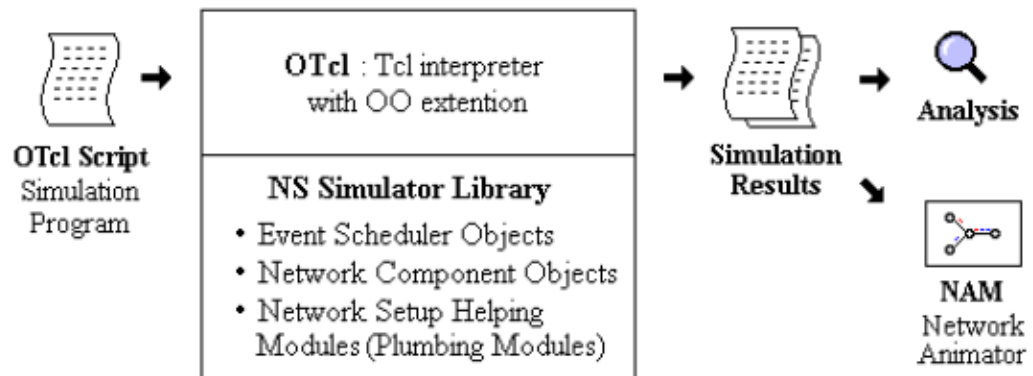


Figure 4.2.2 Simplified user's view of NS

3.3.4 Basic commands in NS2

Node Basics

The basic primitive for creating a node is

```
set ns[new Simulator]
```

```
$ns node
```

The instance procedure node constructs a node out of simpler classifier objects. All nodes contain the following components:

1. An address or id_, monotonically increasing by 1 (from initial value 0) across the simulation namespace as nodes are created
2. A list of neighbours (neighbour_)
3. A list of agents (agent_)
4. A node type identifier (node type_)

Node Methods: Configuring the Node

Procedures to configure an individual node can be classified into:

1. Control functions
2. Address and Port number management, unicast routing functions
3. Agent management

Creating a Tcl Scenario

To define trace files with the data that needs to be collected from the simulation, we have to create these files using the command open:

```
#open the trace file
```

```

set traceFile [open out.tr w]
$ns trace-all $traceFile

#open the Nam trace file
set namFile [open out.nam w]
$ns namtrace-all $namFile
#define the TCP agent
Set tcp [new Agent/TCP]
$ns attach-agent $n(0) $tcp
Set sink [new Agent/TCPSink]
$ns attach-agent $n(1) $sink
$ns connect $tcp $sink

```

Node Configuration

Node configuration essentially consists of defining the different node characteristics before creating them. They may consist of the type of addressing structure used in the simulation, defining the network components for mobile nodes, turning on or off the trace options at Agent/Router/MAC levels, selecting the type of AdHoc routing protocol for wireless nodes or defining their energy model. The node-configure command would look like the following:

```

$ns_ node-configure -addressType hierarchical\
-adhocRouting AODV\ 23
-11TypeLL\
-macTypeMac/802_11\
-ifqType Queue/DropTail/PriQueue\
-ifqLen50\
-antType Antenna/OmniAntenna\
-propType propagation/TwoRayGround\
-phyTypePhy/WirelessPhy\

```

Simulation Procedure

Run the script by typing at the Konsole `Nsfilename.tcl`
 On completion of the run, Trace output file “filename-out.tr” and nam output file

“filename-out.nam” are created. Running filename-out.nam, the mobile nodes moving in the nam window can be seen. The active senders start informing the network about its presence and begin sending data according to the random progress method.

The finish procedure is given as

```
procfinish{} {  
  $ns flush-trace  
  close $r  
  close $nf  
  exec nam -r filename. nam &  
  exit 0  
}
```

In the finish procedure, the trace file buffer is cleared and the graphs are generated in the terminal in a pipelined manner. \$ns is used to close the trace field. Now the animator field is generated using command

```
exec namfilename.nam
```

To run the file **\$ns run** command is used and the tcl script is executed.

To execute the graph **exec ns graph.tcl command is used.**

3.3.5 NAM Tool

Nam is a Tcl/TK based animation tool for viewing network simulation traces and real world packet trace data.

The first step to use nam is to produce the trace file. The trace file should contain topology information, e.g., nodes links, as well as packet traces. The detailed format is described in the TRACE FILE section. Usually, the trace file is generated by ns. During an ns simulation, user can produce topology configurations, layout information, and packet traces using tracing events in ns.

When the trace file is generated, it is ready to be animated by nam. Upon start up, nam will read the trace file, create topology, pop up a window, do layout if necessary, then pause at the time of the first packet in the trace file. Through its user interface, nam provides control over many aspects of animation. These functionalities will be described in detail in the USER INTERFACE section.

3.3.6 OBJECTS IN NAM

Nam does animation using the following building blocks node, link, queue, packet, agent, monitor. They are defined below:

- **node**

Nodes are created from 'n' trace event in trace file. It represents a source/host/router, etc. nam will terminate if there are duplicate definition for the same node. Node may have many shapes, (circle, square, and hexagon), but once created it cannot change its shape. Node may also have many colors; it can change its color during animation.

- **link**

Links are created between nodes to form a network topology. nam links are internally simplex, but it is invisible to the users. The trace event 'l' creates two simplex links and other necessary setups, hence it looks to users identical to a duplex link. Link may have many colors; it can change its color during animation.

- **queue**

Queue needs to be constructed in nam between two nodes. Unlike link, nam queue is associated to a simplex link. The trace event 'q' only creates a queue for a simplex link. In nam, queues are visualized as stacked packets. Packets are stacked along a line, the angle between the line and the horizontal line can be specified in the trace event 'q'.

- **packet**

Packet is visualized as a block with an arrow. The direction of the arrow shows the flow direction of the packet. Queued packets are shown as little squares. A packet may be dropped from a queue or a link. Dropped packets are shown as rotating squares, and disappear at the end of the screen. Dropped packets are not visible during backward animation.

- **agent**

Agents are used to separate protocol states from nodes. They are always associated with nodes. An agent has name, which is a unique identifier of the agent. It is shown as a square with its name inside, and a line link the square to its associated node.

3.3.7 AUTOMATIC LAYOUT

In nam, a topology is specified by alternating node objects with edge objects. But to display the topology in a comprehensible way, a layout mechanism is needed. Currently nam provides two layout methods.

First, user may specify edges orientations. An edge orientation is the angle between the edge and the horizontal line, in the interval $[0, 2\pi)$. During layout, nam will honour the given edge orientations. Generally, it will first choose a reference node, then place other nodes using edge orientation and edge length, which is determined by line delay. This works well for small and manually generated topologies.

Second, when we are dealing with randomly generated topologies, be it small or large, we may want to do layout automatically. An automatic graph layout algorithm is adapted and implemented. The basic idea of the algorithm is to model the graph as balls (nodes) connected by spring (edges). Balls will repulse each other, while springs pull them together. This system will (hopefully) converge after some iterations. In practice, after a small number of iterations (tens or hundreds), most graphs will converge to a visually comprehensible structure.

3.3.8 THE USER INTERFACE

The top of the nam window is a menu bar. Two pulldown menus are on the left of the menu bar. The 'File' menu currently only contains a 'Quit' button. It has a 'Open...' button as well, but that is not implemented yet. The 'View' menu has 4 buttons:

- **New view button:** Creates a new view of the same animation. User can scroll and zoom on the new view. All views will be animated synchronously.
- **Show monitors checkbox:** If checked, will show a panel at the lower half of window, where monitors will be displayed.
- **Show auto layout checkbox:** If checked, will show a panel at the lower half of window, which contains input boxes and a button for automatic layout adjusts. This box may not always be enabled. When a trace file has its own layout specifications, this box will be disabled. If and only if the trace file does not have complete layout specification (i.e., each link has orientation specified in the traces), will this box be enabled.
- **Show annotation checkbox:** If checked, will show a list-box at the lower half of window, which will be used to list annotations in the ascending order of time.

Acceleration Keys

ALT+'f' will pull down the 'File' menu.

ALT+'v' will pull down the 'Open...' menu.

ESC will abort a menu selection in progress.

Below the menu bar, there is a control bar containing 6 buttons, a label, and a small scrollbar (scale). They can be clicked in any order. We will explain them from left to right.

Button 1 (<<): Rewind. When clicked, animation time will go back at the rate of 25 times the current screen update rate.

Button 2 (<): Backward play. When clicked, animation will be played backward in time.

Button 3 (square): Stop. When clicked, animation will pause.

Button 4 (>): Forward play. When clicked, animation will be played in time ascending order.

Button 5 (>>): Fast Forward. When clicked, animation time will go forward at the rate of 25 times the current screen update rate.

Button 6 (Chevron logo): Quit.

Time label: Show the current animation time (i.e., simulation as in the trace file).

Rate slider: Controls the screen update rate (animation granularity). The current rate is displayed in the label above the slider.

3.3.9 AWK Script

To evaluate the trace file generated as simulation results having the above describe trace format we use AWK scripts to find delay, throughput.

AWK is a programming language that is designed for processing text based data, either in files or data streams, and was created at Bell Labs in the 1970s. The name AWK is derived from the family names of its authors - Alfred Aho, Peter Weinberger and Brian Kernighan.

AWK is a language for processing files of text. A file is treated as a sequence of records, and by default each line is a record. Each line is broken up into a sequence of fields, so we can think of the first word in a line as the first field, the second

word as the second field, and so on. An AWK program is of a sequence of pattern-action statements. AWK reads the input a line at a time. A line is scanned for each pattern in the program, and for each pattern that matches, the associated action is executed. Wireless traces begin with one of four characters followed by one of two different trace formats, depending on whether the trace logs the X and Y coordinates of the mobile node.

3.4 SYSTEM IMPLEMENTATION

Modules Name

- Network formation
- Activation
- Sharing data
- Communication

Module Description

3.4.1 Network formation

The network contains the ‘n’ number of vehicle nodes, one cloud server, one work station and some Road Side Units (RSU). All the actions in the network are maintained by the cloud server and work station. Hence successful deployment of vehicular ad-hoc networks will require incentives for vehicle manufacturers, consumers and the government is a challenge to implement security in VANET.

3.4.2 Activation

At first all the vehicles in the network send a hello message to its neighbour vehicle to inform its position, direction and the moving speed. And the information is monitored by the road side units. VANET is time critical where safety related message should be delivered with 100ms transmission delay. So message and entity authentication must be done in time

3.4.3 Sharing data

Vehicles of VANET can provide several safety applications and information to other vehicle drivers concerning collision notification and avoidance, crash prevention and safety, traffic incident management, and so on. These safety applications could provide information about a larger vehicular environment compared to information provided by traditional VANETs or by the traditional cloud. Security is the major issue to implement the VANET. We study the security requirements and challenges to implement the security measure in the VANET. Different types of attacks and their solutions.

3.4.4 Communication

The source first discovers the path through existing route discovery protocol and determines the initial Active Path Sets (APS) for communication. After completion of

this a source has a set of APS. The source disperses each outgoing messages into a number of pieces and encodes and transmits across different routes. Each dispersed piece carries a MAC (Message Authentication Code) which is used to check the integrity and authentication of its origin. Based on the packet received or failed on different APS, the source node rates the APS path. The destination validates and sends a feedback acknowledgement to the source. And the communication done by two or more vehicles even they are in different sides of the road.

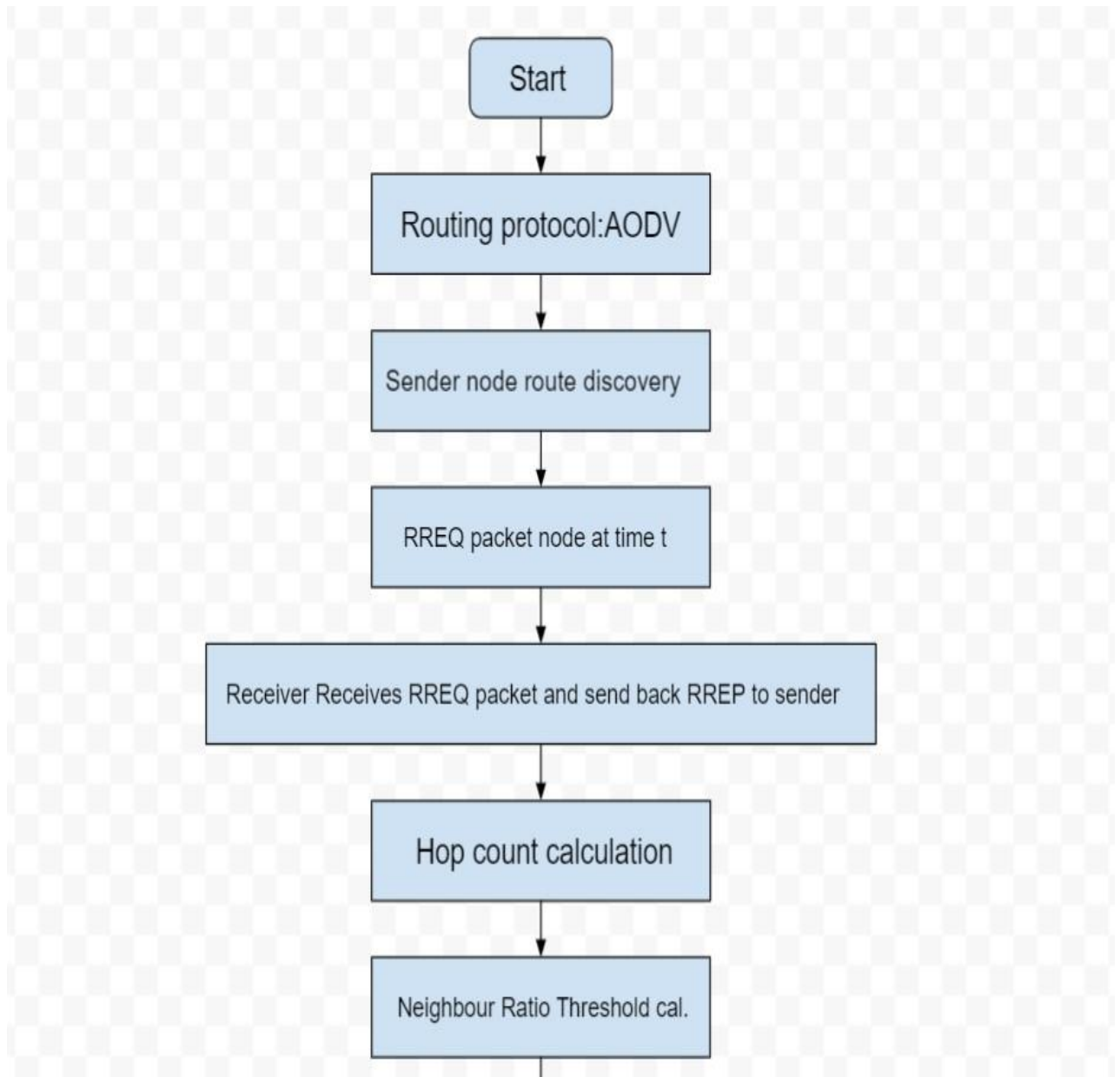
3.5 Hardware Requirements:

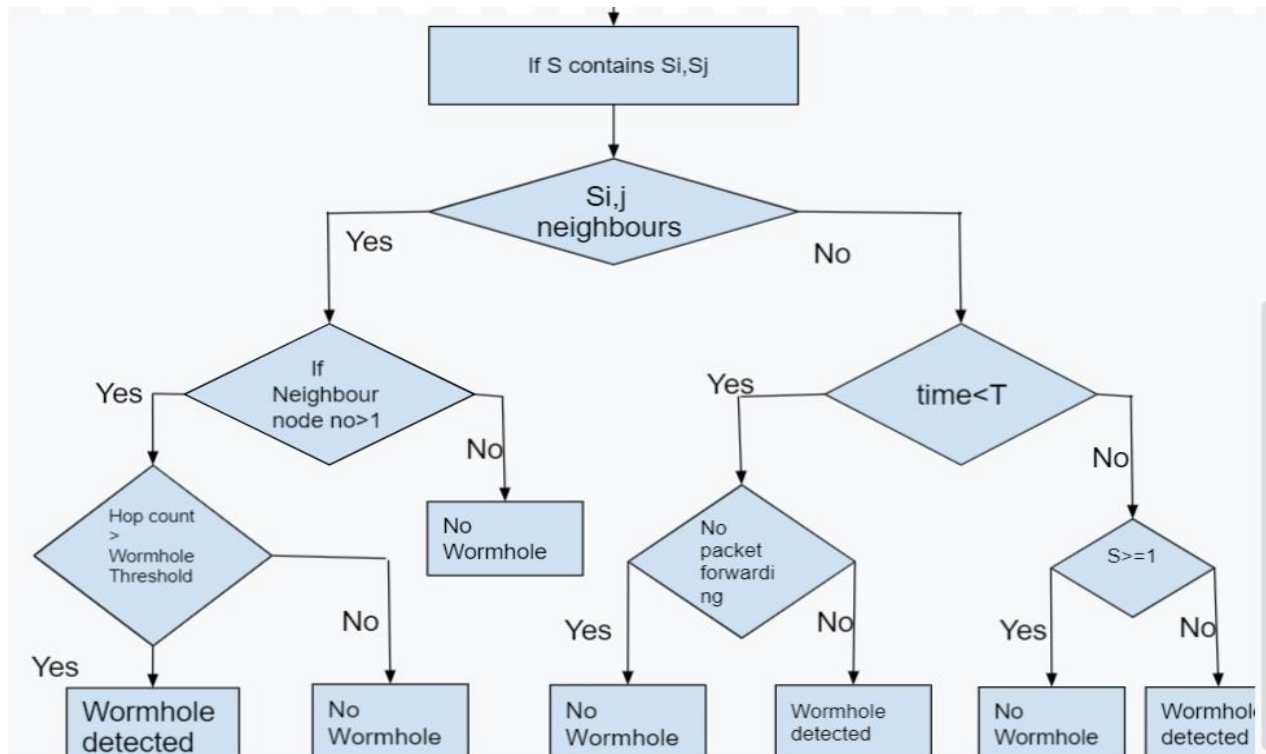
Hard Disk	:	40 GB
Processor	:	Any processor above 500 MHz
RAM	:	Above 512 MB
Input device	:	Standard Keyboard and Mouse.
Output device	:	VGA and High Resolution Monitor.

CHAPTER IV

SYSTEM DESIGN

4.1 DATA FLOW DIAGRAM





AODV Routing protocol is used here to route the packets, the sender node then initiates the route discovery process. A route request is sent to the new destination node if it is not present in the table and the time t is stored. When the receiver node receives the route request (RREQ) it replies the sender with a Route reply packet(RREP). The number of hops it took to reach the destination is calculated, known as hop count. After hop count calculation suspected nodes are identified based on the following formulas : $n\bar{1} = s/n_i$; $r_i = n_i/n\bar{1}$; $r_i > r$. Then all the suspected nodes are added into a set, if the set contains any of the node pairs taken as input then few conditions help classify the nodes as internal or external wormhole.

Classification conditions for wormhole :

If the node pair are neighbors and the neighbor node number is greater than 1, hop count is also greater than 1 then external wormhole is detected wormhole. Else no internal wormhole is suspected and time taken for the packet to be sent is compared with the time t that has been previously saved if T is less than time and if packet forwarding is done then a external wormhole is detected else no external wormhole is suspected.

CHAPTER V

TESTING

5.1 TESTING IN NS2

Testing your code is an integral part of developing quality software. To guide software development and monitor for regressions in code functionality, you can write unit tests for your programs. To measure the time, it takes for your code (or your tests) to run, you can write performance tests.

Script-Based Unit Tests

Function-Based Unit Tests

Class-Based Unit Tests

Extend Unit Testing Framework

Performance Testing Framework

5.2 SYSTEM TESTING AND MAINTENANCE

Testing is a set activity that can be planned and conducted systematically. Testing begins at the module level and work towards the integration of entire computers based system. Nothing is complete without testing, as it is vital success of the system.

- Testing Objectives:

There are several rules that can serve as testing objectives, they are

- Testing is a process of executing a program with the intent of finding an error
- A good test case is one that has high probability of finding an undiscovered error.
- A successful test is one that uncovers an undiscovered error.

If testing is conducted successfully according to the objectives as stated above, it would uncover errors in the software. Also testing demonstrates that software functions appear to be working according to the specification, that performance requirements appear to have been met.

There are three ways to test a program

- For Correctness
- For Implementation efficiency

- For Computational Complexity.

Tests for correctness are supposed to verify that a program does exactly what it was designed to do. This is much more difficult than it may at first appear, especially for large programs.

Tests for implementation efficiency attempt to find ways to make a correct program faster or use less storage. It is a code-refining process, which re-examines the implementation phase of algorithm development.

Tests for computational complexity amount to an experimental analysis of the complexity of an algorithm or an experimental comparison of two or more algorithms, which solve the same problem.

- Testing Correctness

The following ideas should be a part of any testing plan:

- Preventive Measures
- Spot checks
- Testing all parts of the program
- Test Data
- Looking for trouble
- Time for testing
- Re Testing

The data is entered in all forms separately and whenever an error occurred, it is corrected immediately. A quality team deputed by the management verified all the necessary documents and tested the Software while entering the data at all levels. The entire testing process can be divided into 3 phases

- Unit Testing
- Integrated Testing
- Validation Testing
- Maintenance

UNIT TESTING:

As this system was partially GUI based WINDOWS application, the following were tested in this phase

- Tab Order
- Reverse Tab Order
- Field length
- Front end validations

In our system, Unit testing has been successfully handled. The test data was given to each and every module in all respects and got the desired output. Each module has been tested found working properly.

INTEGRATION TESTING:

Test data should be prepared carefully since the data only determines the efficiency and accuracy of the system. Artificial data are prepared solely for testing. Every program validates the input data.

VALIDATION TESTING:

In this, all the Code Modules were tested individually one after the other. The following were tested in all the modules

- Loop testing
- Boundary Value analysis
- Equivalence Partitioning Testing

In our case all the modules were combined and given the test data. The combined module works successfully without any side effect on other programs. Everything was found fine working.

MAINTENANCE:

The Objectives of this maintenance work are to make sure that the system gets into the work all time without any bug. Provision must be for environmental changes which may affect the computer or a software system. This is called the maintenance of the system. Nowadays there is a rapid change in the software world. Due to this rapid change, the system should be capable of adapting these changes. In our project the process can be added with affecting other parts of the system.

Maintenance plays a vital role. The system liable to accept any modification after its implementation. This system has been designed to all new changes. Doing this will not affect the system's performance or its accuracy.

5.3 SYSTEM SYUDY

FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- **ECONOMICAL FEASIBILITY**
- **TECHNICAL FEASIBILITY**
- **SOCIAL FEASIBILITY**

ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement; as only minimal or null changes are required for implementing this system.

SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, in which he is welcomed, as he is the final user of the system.

CHAPTER VI

RESULTS

6.1 SIMULATION WITHOUT WORMHOLE NODES

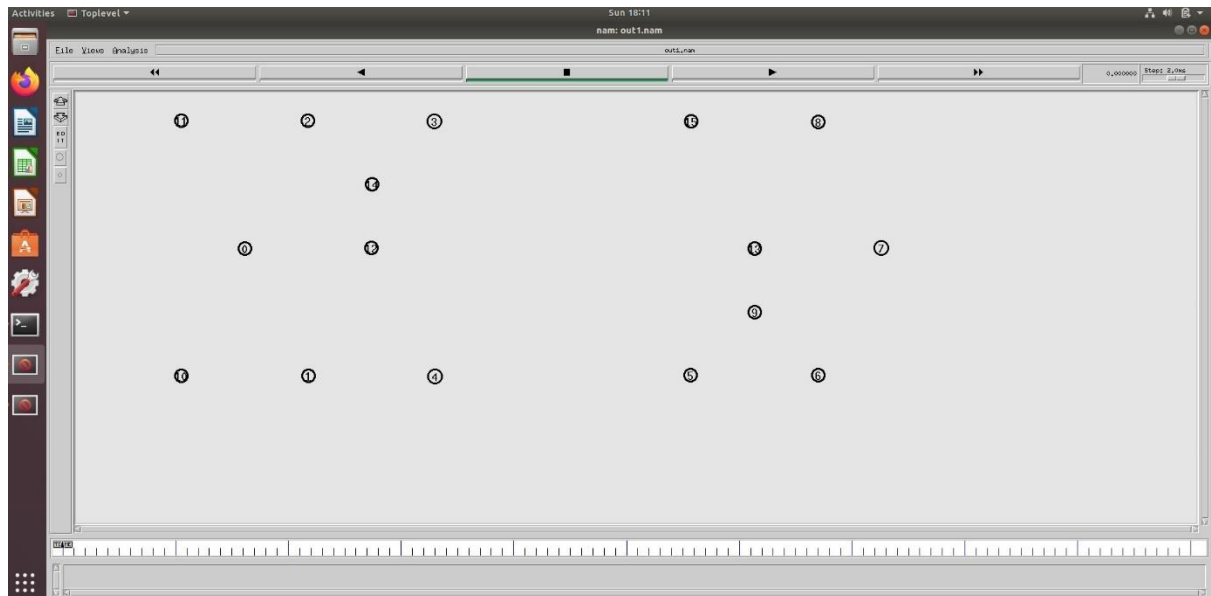


FIG 6.1.1 : NAM WINDOW

The above figure shows the NAM Window representing the node simulation.

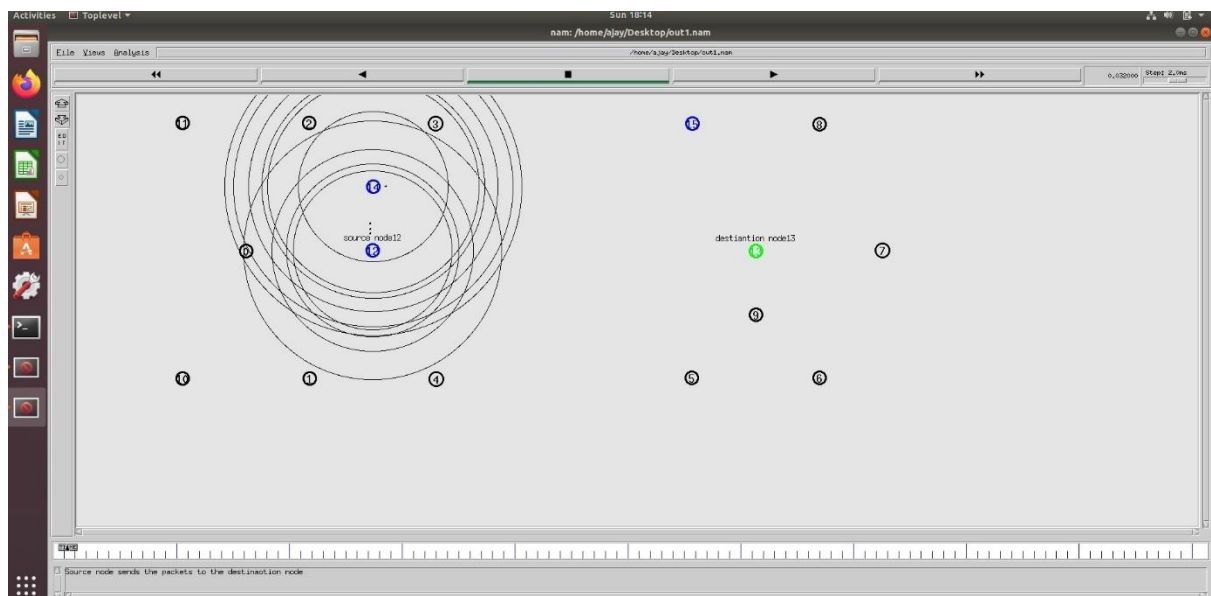


FIG 6.1.2 : Sending RREQ from source to destination

The above figure shows the simulation of the source node sending route request packets to a

new node in the network

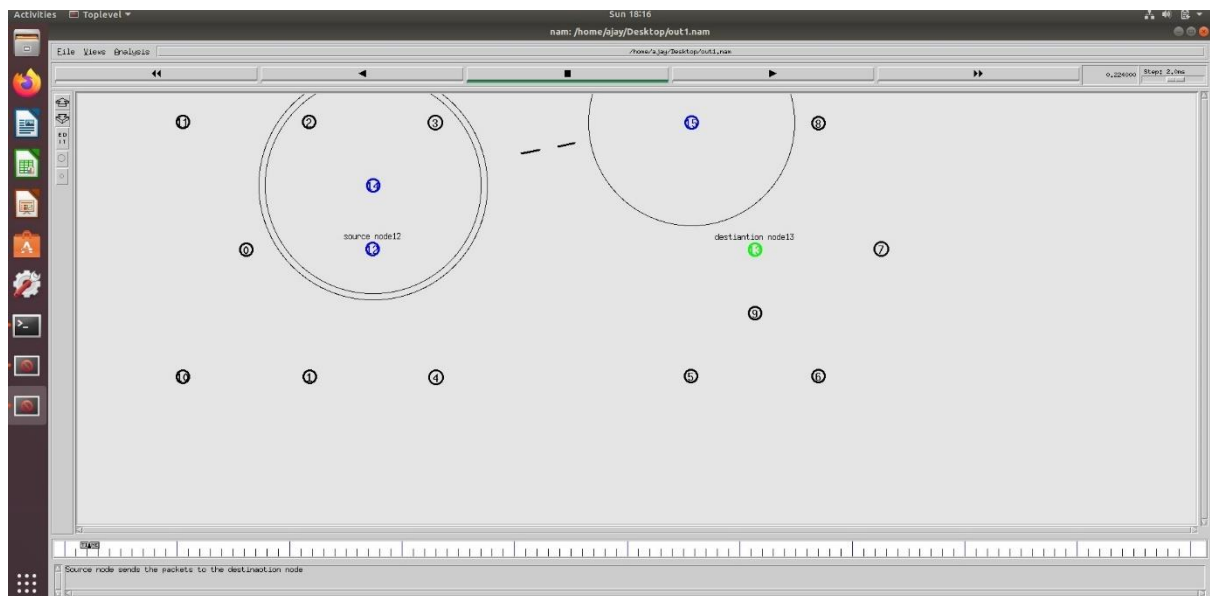


Fig 6.1.3 : Transmission of packets from source to destination

The simulation shows the transfer of packets as per the AODV protocol from the source node to the destination node.

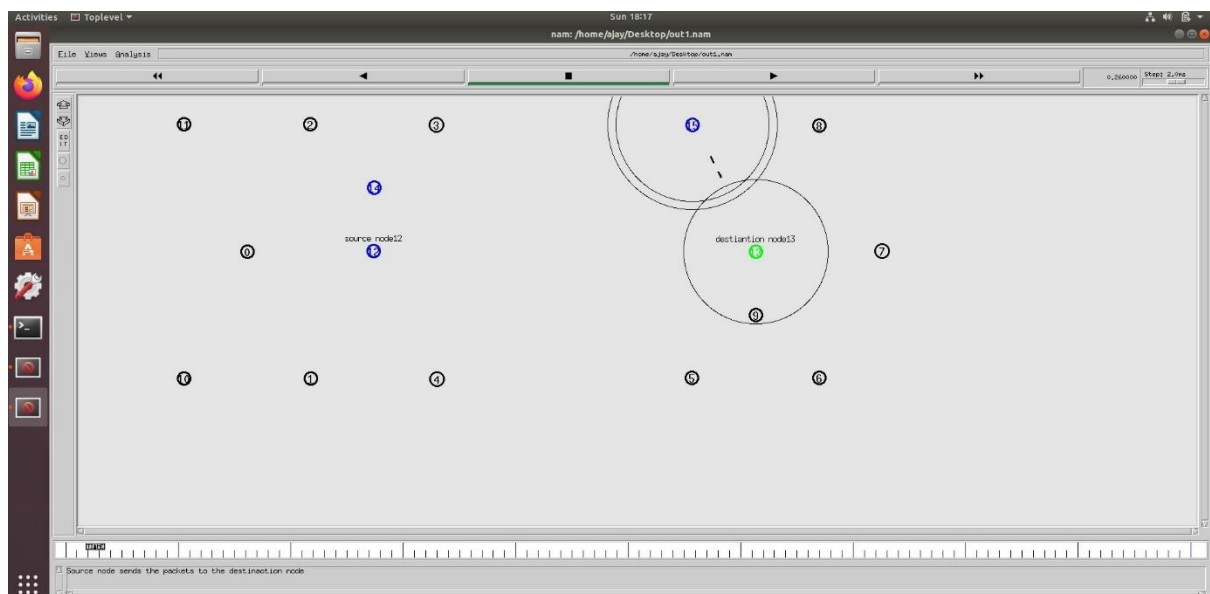


Fig 6.1.4 : Destination receiving packets

The simulation shows the packets being sent to the destination node as per the AODV protocol.

6.2 SIMULATION WITH WORMHOLE NODES

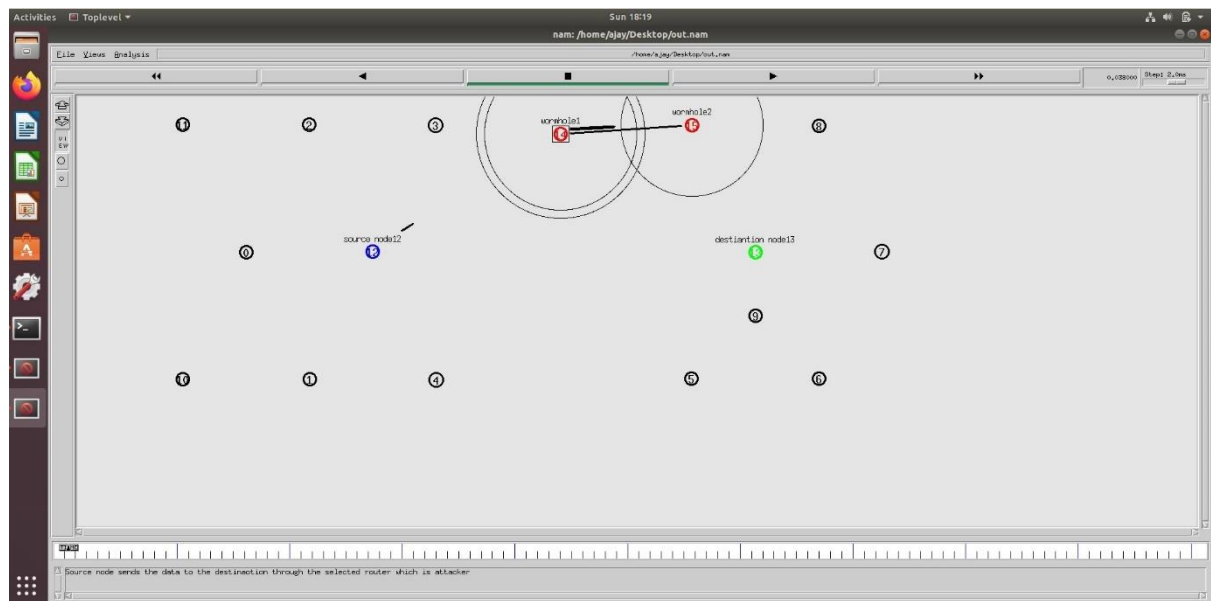


Fig 6.2.1: Sending Packets through Wormhole tunnel

The above simulation shows the presence of wormhole that has formed a tunnel between the two malicious nodes. And the packets being tunneled to the wormhole nodes instead of following the actual AODV routing path.

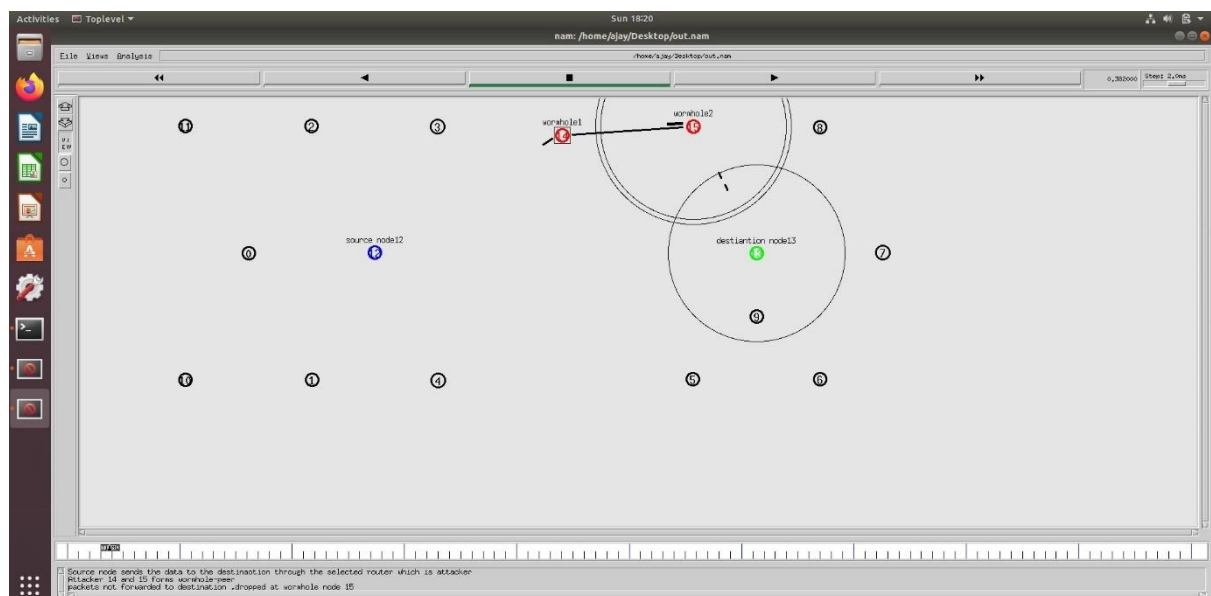


Fig 6.2.2 : Destination receiving packets from wormhole tunnel

The simulation shows the corrupt packets being received from the wormhole node.

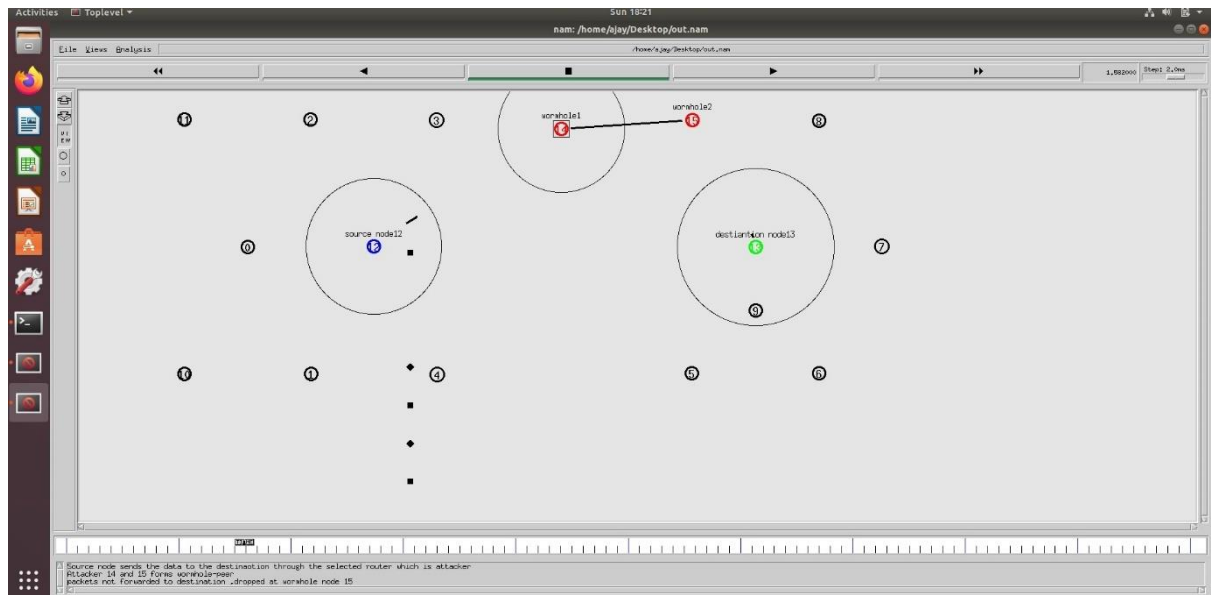


Fig 6.2.3: Dropping of packets due to Wormholes

The simulation above depicts how the packets are dropped when a wormhole node disrupts the network.

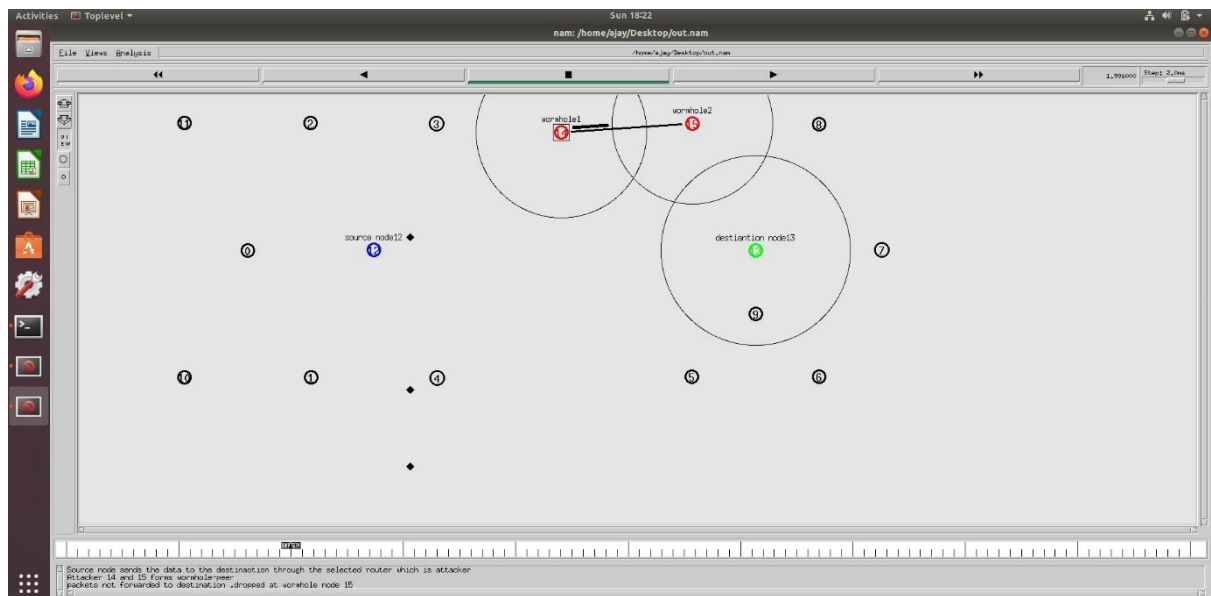


Fig 6.2.4 : Dropping and Sending Packets

The simulation shows the simultaneous dropping and sending of packets.

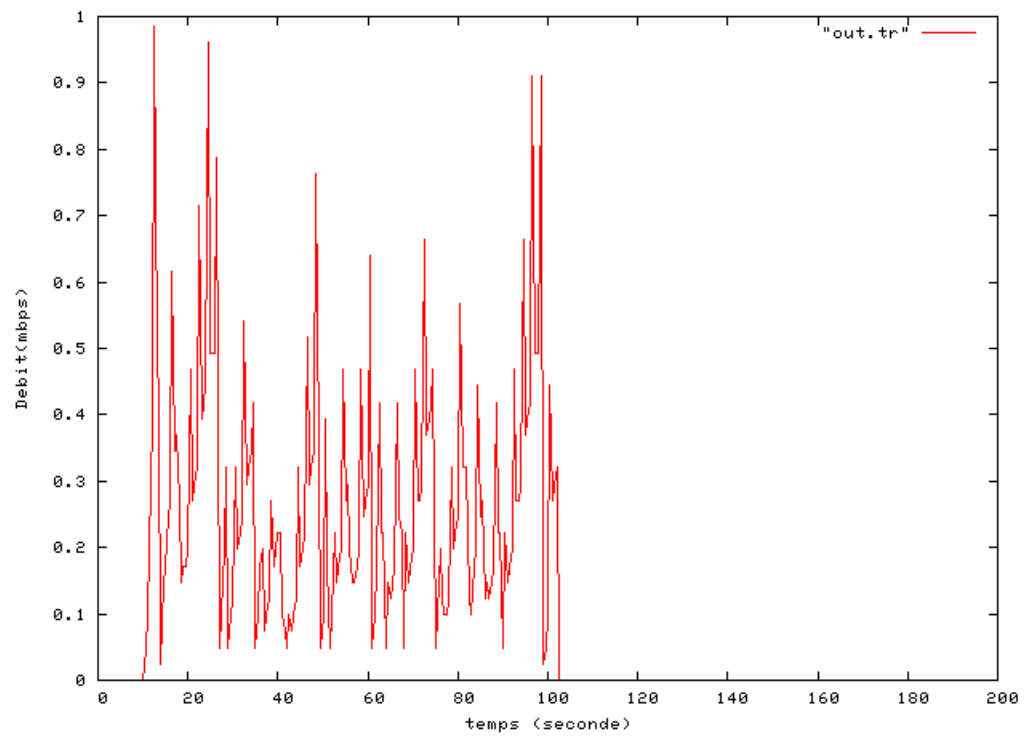


Fig6.3: Xgraph for the output

56
CHAPTER VII

CONCLUSION AND FUTURE SCOPE

In this paper, we propose a Credible Neighbor Discovery (CREDND) protocol against wormhole in VANET, based on hop difference and local monitoring. CREDND can detect not only external wormholes, but also internal wormholes. It improves the ability of wormhole attack defense in ND without additional hardware, and saves node energy at the same time. We also propose the concept of the neighbor ratio threshold, which contribute to improving the accuracy rate and energy efficiency of wormhole detection. Through the simulation experiment, we can conclude that CREDND has better performance in the wormhole detection than other same types of algorithms. However, there are still some shortcomings that we need to overcome in the future, such as CREDND cannot work well in the condition that all nodes in VANET have different communication range and conform to other distribution. It is of great significance to study wormhole detection under existing conditions (without any additional hardware devices), so we will seek the more effective solution and try to apply it to more networks.

REFERENCES

1. G. Wu, X. Chen, L. Yao, Y. Lee, and K. Yim, “An efficient wormhole attack detection method in wireless sensor networks,” *Computer Science and Information Systems*, vol. 11, no. 3, pp. 1127–1141, 2014.
2. Y.-C. Hu, A. Perrig, and D. B. Johnson, “Packet leashes: a defense against wormhole attacks in wireless networks,” in *Proceedings of the 22nd Annual Joint Conference on the IEEE Computer and Communications Societies (INFOCOM '03)*, vol. 3, pp. 1976–1986, April 2003.
3. L. Hu and D. Evans, “Using directional antennas to prevent wormhole attacks,” in *Proceedings of the Network and Distributed System Security Symposium*, pp. 1–11, San Diego, Calif, USA, February 2004.
4. H. Vu, A. Kulkarni, K. Sarac, and N. Mittal, “Wormeros: a new framework for defending against wormhole attacks on wireless ad hoc networks,” in *Proceedings of the International Conference on Wireless Algorithms, Systems, and Applications*, pp. 491–502, Springer, 2008.
5. M. R. Alam and K. S. Chan, “RTT-TC: a topological comparison based method to detect wormhole attacks in MANET,” in *Proceedings of the 12th IEEE International Conference on Communication Technology (ICCT '10)*, pp. 991–994, IEEE, Nanjing, China, November 2010.
6. X. Ban, R. Sarkar, and J. Gao, “Local connectivity tests to identify wormholes in wireless networks,” in *Proceedings of the 12th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc '11)*, ACM, Paris, France, May 2011.
7. X. Lu, D. Dong, and X. Liao, “MDS-based wormhole detection using local topology in wireless sensor networks,” *International Journal of Distributed Sensor Networks*, vol. 2012, Article ID 145702, 9 pages, 2012.
8. R. Maheshwari, J. Gao, and S. R. Das, “Detecting wormhole attacks in wireless networks using connectivity information,” in *Proceedings of the IEEE 26th IEEE International Conference on Computer Communications (INFOCOM '07)*, pp. 107–115, May 2007.
9. W. Wang and A. Lu, “Interactive wormhole detection in large scale wireless networks,” in *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology (VAST '06)*, pp. 99–106, November 2006.

10. S. Čapkun, L. Buttyán, and J.-P. Hubaux, “SECTOR: secure tracking of node encounters in multi-hop wireless networks,” in *Proceedings of the 1st ACM Workshop on Security of Ad Hoc and Sensor Networks*, pp. 21–32, ACM, Washington, DC, USA, October 2003.
11. H. S. Chiu and K.-S. Lui, “DePHI: wormhole detection mechanism for ad hoc wireless networks,” in *Proceedings of the 2006 1st International Symposium on Wireless Pervasive Computing*, pp. 1–6, IEEE, Spa Phuket, Thailand, January 2006.
12. Z. Tun and A. H. Maw, “Wormhole attack detection in wireless sensor networks,” *International Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering*, vol. 2, no. 10, pp. 2184–2189, 2008.
13. I. Khalil, S. Bagchi, and N. B. Shroff, “LITE WOPR: a lightweight countermeasure for the wormhole attack in multihop wireless networks,” in *Proceedings of the International Conference on Dependable Systems and Networks (DSN '05)*, pp. 612–621, Yokohama, Japan, July 2005.
14. I. Khalil, S. Bagchi, and N. B. Shroff, “MOBIWOPR: mitigation of the wormhole attack in mobile multihop wireless networks,” *Ad Hoc Networks*, vol. 6, no. 3, pp. 344–362, 2008.
15. S. Choi, D.-Y. Kim, D.-H. Lee, and J.-I. Jung, “WAP: wormhole attack prevention algorithm in mobile ad hoc networks,” in *Proceedings of the IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC '08)*, pp. 343–348, Taichung, Taiwan, June 2008.
16. R. Poovendran and L. Lazos, “A graph theoretic framework for preventing the wormhole attack in wireless ad hoc networks,” *Wireless Networks*, vol. 13, no. 1, pp. 27–59, 2007.
17. Z. Zhao, B. Wei, X. Dong, L. Yao, and F. Gao, “Detecting wormhole attacks in wireless sensor networks with statistical analysis,” in *Proceedings of the WASE International Conference on Information Engineering (ICIE '10)*, vol. 1, pp. 251–254, August 2010.
18. L. Lu, M. J. Hussain, G. Luo, and Z. Han, “Pworm: passive and real-time wormhole detection scheme for WSNs,” *International Journal of Distributed Sensor Networks*, vol. 2015, Article ID 356382, 16 pages, 2015.
19. W. Wang and B. Bhargava, “Visualization of wormholes in sensor networks,” in *Proceedings of the 3rd ACM Workshop on Wireless Security*, pp. 51–60, ACM, October 2004.
20. D. Dong, M. Li, Y. Liu, X.-Y. Li, and X. Liao, “Topological detection on wormholes in wireless ad hoc and sensor networks,” *IEEE/ACM Transactions on Networking*, vol. 19, no. 6, pp. 1787–1796, 2011.

21. H. Chen, W. Lou, Z. Wang, J. Wu, Z. Wang, and A. Xi, "Securing DV-Hop localization against wormhole attacks in wireless sensor networks," *Pervasive and Mobile Computing*, vol. 16, pp. 22–35, 2015.
22. H. Chen, W. Chen, Z. Wang, Z. Wang, and Y. Li, "Mobile beacon based wormhole attackers detection and positioning in wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 10, no. 3, Article ID 910242, 2014.
23. M. C. Van Wezel and W. A. Kusters, "Nonmetric multidimensional scaling: neural networks versus traditional techniques," *Intelligent Data Analysis*, vol. 8, no. 6, pp. 601–613, 2004.
24. V. D. M. Nhat, D. Vo, S. Challa, and S. Lee, "Nonmetric MDS for sensor localization," in *Proceedings of the 3rd International Symposium on Wireless Pervasive Computing (ISWPC '08)*, pp. 396–400, IEEE, Santorini, Greece, May 2008.
25. C. Miao, G. Dai, K. Mao, Y. Li, and Q. Chen, "RIMDS: multidimensional scaling iterative localization algorithm using RSSI in wireless sensor networks," in *Proceedings of the China Conference on Wireless Sensor Networks*, pp. 164–175, Springer, 2014.
26. B. Li, W. Cui, and B. Wang, "A robust wireless sensor network localization algorithm in mixed LOS/NLOS scenario," *Sensors*, vol. 15, no. 9, pp. 23536–23553, 2015.
27. V. K. Chaurasiya, N. Jain, and G. C. Nandi, "A novel distance estimation approach for 3D localization in wireless sensor network using multi dimensional scaling," *Information Fusion*, vol. 15, no. 1, pp. 5–18, 2014.
28. X. Zhang, Y. Wu, and X. Wei, "Localization algorithms in wireless sensor networks using nonmetric multidimensional scaling with RSSI for precision agriculture," in *Proceedings of the 2nd IEEE International Conference on Computer and Automation Engineering (ICCAE '10)*, vol. 5, pp. 556–559, Singapore, February 2010.
29. F. Groenen, J. Patrick, and M. Velden, *Multidimensional Scaling*, Wiley Online Library, 2005.
30. W. Härdle and L. Simar, *Applied Multivariate Statistical Analysis*, vol. 22007, Springer, Berlin, Germany, 2007.
31. D. Eppstein, "Arboricity and bipartite subgraph listing algorithms," *Information Processing Letters*, vol. 51, no. 4, pp. 207–211, 1994.

BIBLIOGRAPHY

- EFFICIENT MECHANISM FOR DETECTION OF WORMHOLE ATTACKS--(Bahekar Sushant Sudhakar)
- INTRODUCTION TO NETWORK SIMULATOR NS2--(Teerawat Issariyakul,Ekram Hossain)
- VEHICULAR ADHOC NETWORKS--(Claudia Campolo)

URL'S

- <https://ieeexplore.ieee.org/document/5340317>
- <https://www.google.com/url?sa=t&source=web&rct=j&url=https://www.cs.rice.edu/~dbj/pubs/jsac-wormhole.pdf&ved=2ahUKEwjA34Kmh-XxAhWUFLcAHZ13D6IQFjAWegQIGRAC&usg=AOvVaw2LFrHumgJPEa3d9qCHsrOU>
- <https://www.sciencedirect.com/topics/computer-science/vehicular-ad-hoc-network>
- https://en.m.wikipedia.org/wiki/Vehicular_ad_hoc_network
- https://en.m.wikipedia.org/wiki/Network_simulation
- <https://hal.archives-ouvertes.fr/>

APPENDIX

Network of 2 wormholes and 14 nodes.

#set n0 [\$ns node]

#\$n0 random-motion 0

#\$ns initial_node_pos \$n0 30

Phy/WirelessPhy set CPTthresh_ 10.0

Phy/WirelessPhy set CSTthresh_ 9.21756e-11 ;#550m

Phy/WirelessPhy set RXThresh_ 4.4613e-10 ;#250m

Phy/WirelessPhy set bandwidth_ 512kb

Phy/WirelessPhy set Pt_ 8000.2818

Phy/WirelessPhy set freq_ 2.4e+9

Phy/WirelessPhy set L_ 1.0

Antenna/OmniAntenna set X_ 0

Antenna/OmniAntenna set Y_ 0

Antenna/OmniAntenna set Z_ 0.25

Antenna/OmniAntenna set Gt_ 1

Antenna/OmniAntenna set Gr_ 1

set val(chan) Channel/WirelessChannel ;# **channel type**

set val(prop) Propagation/TwoRayGround ;# **radio-propagation model**

set val(netif) Phy/WirelessPhy ;# **network interface type**

set val(mac) Mac/802_11 ;# **MAC type**

set val(ifq) Queue/DropTail/PriQueue ;# **interface queue type**


```

set val(ll) LL ;# link layer type

set val(ant) Antenna/OmniAntenna ;# antenna model

set val(ifqlen) 50 ;# max packet in ifq

set val(nn) 16 ;# number of mobilenodes

set val(rp) AODV ;# routing protocol

set val(x) 1440 ;# X dimension of topography

set val(y) 100 ;# Y dimension of topography

set val(stop) 10.0 ;# time of simulation end

#set val(wormholes) 1;

#=====

# Initialization

#=====

#Create a ns simulator

set ns [new Simulator]


#Setup topography object

set topo [new Topography]

$topo load_flatgrid $val(x) $val(y)

create-god $val(nn)


#Open the NS trace file

set tracefile [open worm.tr w]

$ns trace-all $tracefile

```

#Open the NAM trace file

```
set namfile [open out.nam w]
```

```
$ns namtrace-all $namfile
```

```
$ns namtrace-all-wireless $namfile $val(x) $val(y)
```

```
set chan [new $val(chan)];#Create wireless channel
```

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

Mobile node parameter setup

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

```
$ns node-config -adhocRouting $val(rp) \
```

```
    -llType $val(ll) \
```

```
    -macType $val(mac) \
```

```
    -ifqType $val(ifq) \
```

```
    -ifqLen $val(ifqlen) \
```

```
    -antType $val(ant) \
```

```
    -propType $val(prop) \
```

```
    -phyType $val(netif) \
```

```
    -channel $chan \
```

```
    -topoInstance $topo \
```

```
    -agentTrace ON \
```

```
    -routerTrace ON \
```

```
    -macTrace ON \
```

```
    -movementTrace ON
```

Nodes Definition

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

```
#Create 14 nodes
```

```
set n0 [$ns node]
```

```
$n0 set X_ 100
```

```
$n0 set Y_ 399
```

```
$n0 set Z_ 0.0
```

```
$ns initial_node_pos $n0 20
```

```
set n1 [$ns node]
```

```
$n1 set X_ 200
```

```
$n1 set Y_ 200
```

```
$n1 set Z_ 0.0
```

```
$ns initial_node_pos $n1 20
```

```
set n2 [$ns node]
```

```
$n2 set X_ 199
```

```
$n2 set Y_ 600
```

```
$n2 set Z_ 0.0
```

```
$ns initial_node_pos $n2 20
```

```
set n3 [$ns node]
```

```
$n3 set X_ 398
```

\$n3 set Y_ 599

\$n3 set Z_ 0.0

\$ns initial_node_pos \$n3 20

set n4 [\$ns node]

\$n4 set X_ 399

\$n4 set Y_ 199

\$n4 set Z_ 0.0

\$ns initial_node_pos \$n4 20

#\$ns at 0.01 "\$n4 label \"wormhole2\""

#\$ns at 0.0 "[\$n4 set ragent_] malicious"

set n5 [\$ns node]

\$n5 set X_ 799

\$n5 set Y_ 201

\$n5 set Z_ 0.0

\$ns initial_node_pos \$n5 20

set n6 [\$ns node]

\$n6 set X_ 1000

\$n6 set Y_ 201

\$n6 set Z_ 0.0

\$ns initial_node_pos \$n6 20

set n7 [\$ns node]

\$n7 set X_ 1099

\$n7 set Y_ 400

\$n7 set Z_ 0.0

\$ns initial_node_pos \$n7 20

set n8 [\$ns node]

\$n8 set X_ 1000

\$n8 set Y_ 598

\$n8 set Z_ 0.0

\$ns initial_node_pos \$n8 20

set n9 [\$ns node]

\$n9 set X_ 900

\$n9 set Y_ 300

\$n9 set Z_ 0.0

\$ns initial_node_pos \$n9 20

set n10 [\$ns node]

\$n10 set X_ 1

\$n10 set Y_ 200

\$n10 set Z_ 0.0

\$ns initial_node_pos \$n10 20

```
set n11 [$ns node]
```

```
$n11 set X_ 1
```

```
$n11 set Y_ 600
```

```
$n11 set Z_ 0.0
```

```
$ns initial_node_pos $n11 20
```

```
#$ns at 0.01 "$n11 label \"wormhole1\""
```

```
set n12 [$ns node]
```

```
$n12 set X_ 299
```

```
$n12 set Y_ 400
```

```
$n12 set Z_ 0.0
```

```
$ns initial_node_pos $n12 20
```

```
$ns at 0.01 "$n12 label \"source node12\""
```

```
set n13 [$ns node]
```

```
$n13 set X_ 900
```

```
$n13 set Y_ 399
```

```
$n13 set Z_ 0.0
```

```
$ns initial_node_pos $n13 20
```

```
$ns at 0.01 "$n13 label \"destiantion node13\""
```

```
# configure Wormholes
```

```
#puts "Making first wormhole"
```

```
set n14 [$ns node]
```

```
$n14 set X_ 300.0
```

```
$n14 set Y_ 500.0
```

```
$n14 set Z_ 0.0
```

```
$ns initial_node_pos $n14 20
```

```
$ns at 0.01 "$n14 label \"wormhole1\""
```

```
set n15 [$ns node]
```

```
$n15 set X_ 800.0
```

```
$n15 set Y_ 599.0
```

```
$n15 set Z_ 0.0
```

```
$ns initial_node_pos $n15 20
```

```
$ns at 0.01 "$n15 label \"wormhole2\""
```

```
#$ns at 0.0 "[$n15 set ragent_] malicious"
```

```
#[ $n14 set ll_(0) ] wormhole-peer [ $n15 set ll_(0) ]
```

```
[ $n15 set ll_(0) ] wormhole-peer [ $n14 set ll_(0) ]
```

```
$ns duplex-link $n14 $n15 2Mb 10ms DropTail
```

\$n12 color blue

\$ns at 0.0 "\$n12 color blue"

\$n13 color blue

\$ns at 0.0 "\$n13 color green"

\$n14 color blue

\$ns at 0.0 "\$n14 color red"

\$n15 color blue

\$ns at 0.0 "\$n15 color red"

#=====

Agents Definition

#=====

#Setup a UDP connection

set udp0 [new Agent/UDP]

\$ns attach-agent \$n12 \$udp0

set null1 [new Agent/Null]

\$ns attach-agent \$n13 \$null1

\$ns connect \$udp0 \$null1

\$udp0 set packetSize_ 1500

#=====

Applications Definition

#=====

71

#Setup a CBR Application over UDP connection

```
set cbr0 [new Application/Traffic/CBR]
```

```
$cbr0 attach-agent $udp0
```

```
$cbr0 set packetSize_ 750
```

```
$cbr0 set rate_ 0.5Mb
```

```
$cbr0 set random_ null
```

```
$ns at 0.0 "$cbr0 start"
```

```
$ns at 10.0 "$cbr0 stop"
```

```
$ns at 0.0 "$ns trace-annotate \"Source node sends the data to the destinaotion through the  
selected router which is attacker\""
```

```
$ns at 0.1 "$ns trace-annotate \"Attacker 14 and 15 forms wormhole-peer\""
```

```
$ns at 0.2 "$ns trace-annotate \"packets not forwarded to destination ,dropped at wormhole  
node 15\""
```

#=====

Termination

#=====

```
#Define a 'finish' procedure
```

```
proc finish {} {
```

```
    global ns tracefile namfile
```

```
$ns flush-trace
```

```
close $tracefile
```

72

```
close $namfile
```

```
exec nam out.nam &
```

```
exit 0
```

```
}
```

```
for {set i 0} {$i < $val(nn)} {incr i} {
```

```
    $ns at $val(stop) "\n$i address?"
```

```
    $ns at $val(stop) "\n$i reset"
```

```
}
```

```
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
```

```
$ns at $val(stop) "finish"
```

```
$ns at $val(stop) "puts \"done\" ; $ns halt"
```

```
$ns run
```