Chapter 1
MOBILE AD-HOC NETWORKS

1.1 Introduction:

In the next generation of wireless communication systems, there will be a need for the rapid deployment of independent mobile users. Significant examples include establishing survivable, efficient, dynamic communication for emergency/rescue operations, disaster relief efforts, and military networks. Such network scenarios cannot rely on centralized and organized connectivity, and can be conceived as applications of Mobile Ad Hoc Networks. A MANET is an autonomous collection of mobile users that communicate over relatively bandwidth constrained wireless links. Since the nodes are mobile, the network topology may change rapidly and unpredictably over time. The network is decentralized, where all network activity including discovering the topology and delivering messages must be executed by the nodes themselves, i.e., routing functionality will be incorporated into mobile nodes.

The set of applications for MANETs is diverse, ranging from small, static networks that are constrained by power sources, to large-scale, mobile, highly dynamic networks. The design of network protocols for these networks is a complex issue. Regardless of the application, MANETs need efficient distributed algorithms to determine network organization, link scheduling, and routing. However, determining viable routing paths and delivering messages in a decentralized environment where network topology fluctuates is not a well-defined problem. While the shortest path (based on a given cost function) from a source to a destination in a static network is usually the optimal route, this idea is not easily extended to MANETs. Factors such as variable wireless link quality, propagation path loss, fading, multiuser interference, power expended, and topological changes, become relevant issues.

The network should be able to adaptively alter the routing paths to alleviate any of these effects. Moreover, in a military environment, preservation of security, latency, reliability, intentional jamming, and recovery from failure are significant concerns. Military networks are designed to maintain a low probability of intercept and/or a low probability of detection. Hence, nodes prefer to radiate as little power as necessary and transmit as infrequently as possible, thus
decreasing the probability of detection or interception. A lapse in any of these requirements may degrade the performance and dependability of the network.

An ad-hoc network is a collection of wireless mobile hosts forming a temporary network without the aid of any stand-alone infrastructure or centralized administration. Mobile

Ad-hoc networks are self-organizing and self re-configuring multi hop wireless networks where, the structure of the network changes dynamically. This is mainly due to the mobility of the nodes. Nodes in these networks utilize the same random access wireless channel, cooperating in a friendly manner to engaging themselves in multi hop forwarding. The nodes in the network not only act as hosts but also as routers that route data to/from other nodes in network.

In mobile ad-hoc networks where there is no infrastructure support as is the case with wireless networks, and since a destination node might be out of range of a source node transmitting packets; a routing procedure is always needed to find a path so as to forward the packets appropriately between the source and the destination. Within a cell, a base station can reach all mobile nodes without routing via broadcast in common wireless networks. In the case of ad-hoc networks, each node must be able to forward data for other nodes. This creates additional problems along with the problems of dynamic topology which is unpredictable connectivity changes.

MANETS rely on wireless transmission, a secured way of message transmission is important to protect the privacy of the data. An insecure ad-hoc network at the edge of an existing communication infrastructure may potentially cause the entire network to become vulnerable to security breaches. In mobile ad hoc networks, there is no central administration to take care of detection and prevention of anomalies.

Mobile devices identities or their intentions cannot be predetermined or verified. Therefore nodes have to cooperate for the integrity of the operation of the network. However, nodes may refuse to cooperate by not forwarding packets for others for selfish reasons and not want to exhaust their resources. Various other factors make the task of secure communication in ad hoc wireless networks difficult include the mobility of the nodes, a promiscuous mode of operation, limited processing power, and limited availability of resources such as battery power, bandwidth and memory.There are two sources of threats to routing protocols.

EXTERNAL ATTACKERS: The first comes from external attackers. By injecting erroneous routing information, replaying old routing information, or distorting routing information, an attacker could successfully partition a network or introduce a traffic overload by causing retransmission and inefficient routing. The second and more severe kind of threat comes from compromised nodes, which might (i) misuse routing information to other nodes or (ii) act on applicative data in order to induce service failures.

The provision of systematic approaches to evaluate the impact of such threats on particular routing protocols remains an open challenge today.

Attacks on ad hoc are classified into non disruptive passive attacks and disruptive active attacks. The active attacks are further classified into internal attacks and external attacks are carried out by nodes that do not belong to network and can be prevented by firewalls and encryption techniques. Internal attacks are from internal nodes which are actually authorized nodes and part of the network hence it is difficult to identify.

In the next generation of wireless communication systems, there will be a need for the rapid deployment of independent mobile users. Significant examples include establishing survivable, efficient, dynamic communication for emergency/rescue operations, disaster relief efforts, and military networks. Such network scenarios cannot rely on centralized and organized connectivity, and can be conceived as applications of Mobile Ad Hoc Networks. A MANET is an autonomous collection of mobile users that communicate over relatively bandwidth constrained wireless links. Since the nodes are mobile, the network topology may change rapidly and unpredictably over time. The network is decentralized, where all network activity including discovering the topology and delivering messages must be executed by the nodes themselves, i.e., routing functionality will be incorporated into mobile nodes.

The set of applications for MANETs is diverse, ranging from small, static networks that are constrained by power sources, to large-scale, mobile, highly dynamic networks. The design of network protocols for these networks is a complex issue. Regardless of the application, MANETs need efficient distributed algorithms to determine network organization, link

scheduling, and routing. However, determining viable routing paths and delivering messages in a decentralized environment where network topology fluctuates is not a well-defined problem. While the shortest path (based on a given cost function) from a source to a destination in a static network is usually the optimal route, this idea is not easily extended to MANETs. Factors such as variable wireless link quality, propagation path loss, fading, multiuser interference, power expended, and topological changes, become relevant issues. The network should be able to adaptively alter the routing paths to alleviate any of these effects. Moreover, in a military environment, preservation of security, latency, reliability, intentional jamming, and recovery from failure are significant concerns. Military networks are designed to maintain a low probability of intercept and/or a low probability of detection. Hence, nodes prefer to radiate as little power as necessary and transmit as infrequently as possible, thus decreasing the probability of detection or interception. A lapse in any of these requirements may degrade the performance and dependability of the network.

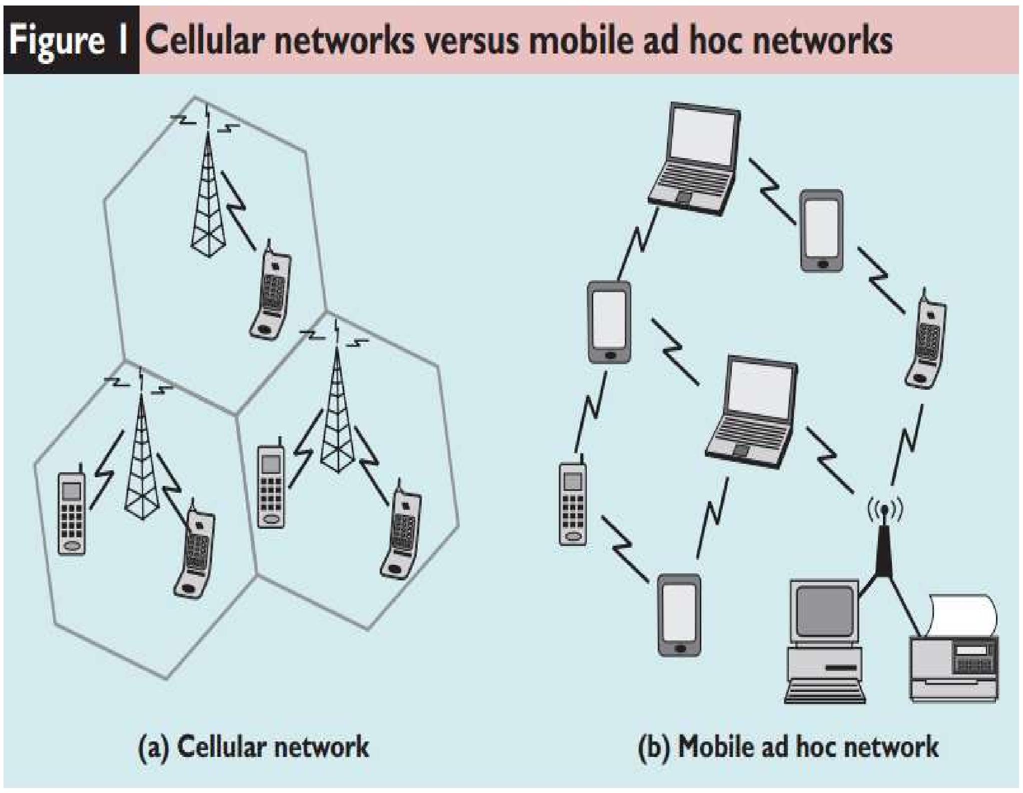


Figure 1 Cellular networks versus mobile ad hoc networks
(a) Cellular network
(b) Mobile ad hoc network

Figure 1.1 cellular networks versus mobile ad hoc networks

1.2 Characteristics of MANET

1. Dynamic topologies: nodes can move freely, network topology may change rapidly, restructuring, but also may also have symmetric and asymmetric links.
2. Bandwidth-constrained, variable capacity links Compared with the wired network environment, the capacity of the wireless link itself is relatively small, but also susceptible to external noise, interference, and signal attenuation effects.
3. Energy-constrained operation: A laptop or handheld computers are often used batteries to provide power, how to save electricity in the context of depletion of system design is also necessary to consider the point.
4. Limited physical security :Network Security With the network deeply embedded in our daily lives and the benefits have become increasingly important in the wireless network to provide security support is also an important issue.

1.3 Wireless ad-hoc network have many advantages:

- ***Low cost of deployment***: Ad hoc networks can be deployed on the fly; hence no expensive infrastructure such as copper wires or data cables is required.
- ***Fast deployment***: Ad hoc networks are very convenient and easy to deploy since there are no cables involved. Deployment time is shortened.
- ***Dynamic Configuration***: Ad hoc network configuration can change dynamically over time. When compared to configurability of LANs, it is very easy to change the network topology of a wireless network.

MANET has various potential applications. Some typical examples include emergency search-rescue operations, meeting events, conferences, and battlefield communication between moving vehicles and/or soldiers. With the abilities to meet the new demand of mobile computation, the MANET has a very bright future.

1.4 Current challenges:

In a mobile ad hoc network, all the nodes cooperate with each other to forward the packets in the network, and hence each node is effectively a router. Thus one of the most important issues is routing. This thesis focuses mainly on routing issues in ad hoc networks. In this section, some of the other issues in ad hoc networks are described:

- ***Distributed network***: A MANET is a distributed wireless network without any fixed infrastructure. That means no centralized server is required to maintain the state of the clients.
- ***Dynamic topology***: The nodes are mobile and hence the network is self-organizing. Because of this, the topology of the network keeps changing over time. Consequently, the routing protocols designed for such networks must also be adaptive to the topology changes.
- ***Power awareness:***Since the nodes in an ad hoc network typically run on batteries and are deployed in hostile terrains, they have stringent power requirements. This implies that the underlying protocols must be designed to conserve battery life.
- ***Addressing scheme***: The network topology keeps changing dynamically and hence the addressing scheme used is quite significant. A dynamic network topology requires a ubiquitous addressing scheme, which avoids any duplicate addresses. In wireless

WAN environments, Mobile IP is being used. Because the static home agents and foreign agents are needed, hence, this solution is not suitable for ad hoc network.

- ***Network size:***The ability to enable commercial applications such as voice transmission in conference halls, meetings, etc., is an attractive feature of ad hoc networks. However, the delay involved in the underlying protocols places a strict upper bound on the size of the network.
- ***Security:*** Security in an ad hoc network is extremely important in scenarios such battlefield. The five goals of security – availability, confidentiality, integrity authenticity and non-repudiation - are difficult to achieve in MANET, mainly because every node in the network participates equally in routing packets.

1.5 MANET Applications:

With the increase of portable devices as well as progress in wireless communication, ad-hoc networking is gaining importance with the increasing number of widespread applications. Ad-hoc networking can be applied anywhere where there is little or no communication infrastructure or the existing infrastructure is expensive or inconvenient to use. Ad hoc networking allows the devices to maintain connections to the network as well as easily adding and removing devices to and from the network. The set of applications for MANET is diverse, ranging from large-scale, mobile, highly dynamic networks, to small, static networks that are constrained by power sources. Besides the legacy applications that move from traditional infra structured environment into the ad hoc context, a great deal of new services can and will be generated for the new environment. Typical applications include

- **Military Battlefield**: Military equipment now routinely contains some sort of computer equipment. Ad- hoc networking would allow the military to take advantage of commonplace network technology to maintain an information network between the soldiers, vehicles, and military information headquarters. The basic techniques of ad hoc network came from this field.
- **Commercial Sector**: Ad hoc can be used in emergency/rescue operations for disaster relief efforts, e.g. in fire, flood, or earthquake. Emergency rescue operations must take place where non-existing or damaged communications infrastructure and rapid deployment of a communication network is needed. Information is relayed from

one rescue team member to another over a small hand held. Other commercial scenarios include e.g. ship-to-ship ad hoc mobile communication, law enforcement, etc.

- **Local Level**: Ad hoc networks can autonomously link an instant and temporary multimedia network using notebook computers or palmtop computers to spread and share information among participants at e.g. conference or classroom. Another appropriate local level application might be in home networks where devices can communicate directly to exchange information. Similarly in other civilian environments like taxicab, sports stadium, boat and small aircraft, mobile ad hoc communications will have manyapplications.
- **Personal Area Network (PAN):**Short-range MANET can simplify the intercommunication between various mobile devices (such as a PDA, a laptop, and a cellular phone). Tedious wired cables are replaced with wireless connections. Such an ad hoc network can also extend the access to the Internet or other networks by mechanisms e.g. Wireless LAN (WLAN), GPRS, and UMTS. The PAN is potentially a promising application field of MANET in the future pervasive computing context.
- **MANET-VoVoN:** A MANET enabled version of JXTA peer-to-peer, modular, open platform is used to support user location and audio streaming over the JXTA virtual overlay network. Using MANET-JXTA, a client can search asynchronously for a user and a call setup until a path is available to reach the user. The application uses a private signaling protocol based on the exchange of XML messages over MANET-JXTA communication channels

1.6 MANET VULNERABILITIES:

Vulnerability is a weakness in security system. A particular system may be vulnerable to unauthorized data manipulation because the system does not verify a user's identity before allowing data access. MANET is more vulnerable than wired network. Some of the vulnerabilities are as follows:-

- **Lack of centralized management**: MANET doesn't have a centralized monitor server. The absence of management makes the detection of attacks difficult because it is not east to monitor the traffic in a highly dynamic and large scale ad-hoc network. Lack of centralized management will impede trust management for nodes.

- **Resource availability**: Resource availability is a major issue in MANET. Providing secure communication in such changing environment as well as protection against specific threats and attacks, leads to development of various security schemes and architectures. Collaborative ad-hoc environments also allow implementation of self-organized security mechanism.
- **Scalability**: Due to mobility of nodes, scale of ad-hoc network changing all the time. So scalability is a major issue concerning security. Security mechanism should be capable of handling a large network as well as small ones.
- **Cooperativeness**: Routing algorithm for MANETs usually assumes that nodes are cooperative and non-malicious. As a result a malicious attacker can easily become an important routing agent and disrupt network operation by disobeying the protocol specifications.
- **Dynamic topology**: Dynamic topology and changeable nodes membership may disturb the trust relationship among nodes. The trust may also be disturbed if some nodes are detected as compromised. This dynamic behavior could be better protected with distributed and adaptive security mechanisms.
- **Limited power supply**: The nodes in mobile ad-hoc network need to consider restricted power supply, which will cause several problems. A node in mobile ad-hoc network may behave in a selfish manner when it is finding that there is only limited power supply.
- **Bandwidth constraint:** Variable low capacity links exists as compared to wireless network which are more susceptible to external noise, interference and signal attenuation effects.
- **Adversary inside the Network**: The mobile nodes within the MANET can freely join and leave the network. The nodes within network may also behave maliciously. This is hard to detect that the behavior of the node is malicious. Thus this attack is more dangerous than the external attack. These nodes are called compromised nodes.

- **No predefined Boundary**: In mobile ad- hoc networks we cannot precisely define a physical boundary of the network. The nodes work in a nomadic environment where they are allowed to join and leave the wireless network. As soon as an adversary comes in the radio range of a node it will be able to communicate with that node. The attacks include Eavesdropping impersonation; tempering, replay and Denial of Service (DoS) attack

1.7 CLASSIFICATION OF MANET ROUTING :

For the nature and challenges found in designing an ad hoc network routing protocol, a large amount of work has been done in the research community to find a perfect routing protocol for wireless ad hoc networks . The research has resulted to a number of routing protocols which can be classified as table-driven or proactive, on-demand or reactive and hybrid routing protocols and shown in Figure.

1.7.1 Table-Driven (or) proactive protocols:

Proactive routing protocols attempt to maintain consistent, up-to-date routing information between every pair of nodes in the network by propagating, proactively, route updates at fixed intervals. As the resulting information is usually maintained in tables, the protocols are sometimes referred to as table-driven protocols. Representative proactive protocols include: Destination-Sequenced Distance Vector (DSDV) routing, Clustered Gateway Switch Routing
,Wireless Routing Protocol (WRP) ,and Optimized Link State Routing (OLSR) .

1.7.2 On-Demand or Reactive Protocols:

A different approach from table-driven routing is reactive or on-demand routing. These protocols depart from the legacy Internet approach. Reactive protocols, unlike table-driven ones, establish a route to a destination when there is a demand for it, usually initiated by the source node through discovery process within the network. Once a route has been established, it is maintained by the node until either the destination becomes inaccessible or until the route is no longer used or has expired. Representative reactive routing protocols include: Dynamic Source Routing (DSR) , Ad hoc On Demand Distance Vector (AODV) routing, Temporally Ordered Routing Algorithm (TORA) and Associativity Based Routing (ABR) .

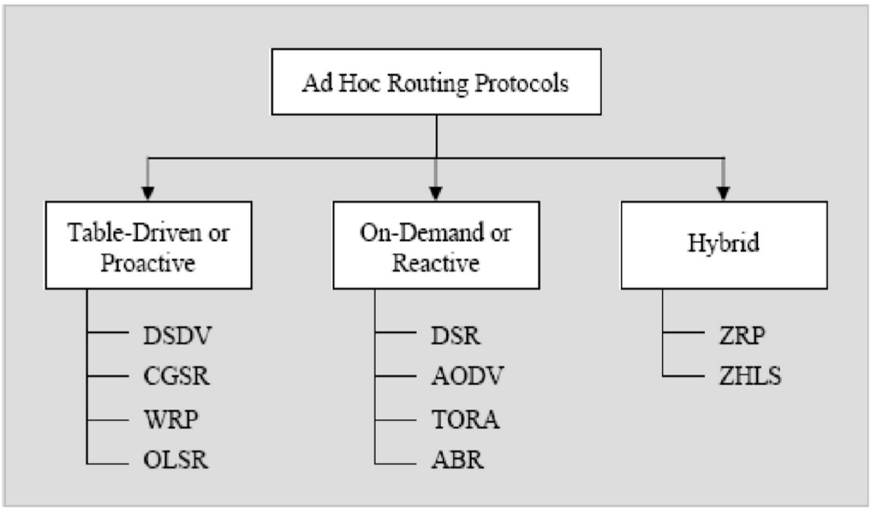


Figure 1.2: Classification of MANET Routing Protocols

1.7.3 Hybrid Routing Protocols:

Purely proactive or purely reactive protocols perform well in a limited region of network setting. However, the diverse applications of ad hoc networks across a wide range of operational conditions and network configuration pose a challenge for a single protocol to operate efficiently . For example, reactive routing protocols are well suited for networks where the call-to-mobility ratio is relatively low. Proactive routing protocols, on the other hand, are well suited for networks where this ratio is relatively high. The performance of either class of protocols degrades when the protocols are applied to regions of ad hoc networks space between the two extremes.

Researchers advocate that the issue of efficient operation over a wide range of conditions can be addressed by a *hybrid* routing approach, where the proactive and the reactive behavior is mixed in the amounts that best match these operational conditions. Representative hybrid routing protocols include: Zone Routing Protocol (ZRP) and Zone-based Hierarchal Link State routing protocol (ZHLS) .

In the following sub sections we examine three protocols, the Dynamic Source Routing (DSR) Protocol, Ad hoc On-demand Distance Vector (AODV) routing protocol and Optimized Link-State Routing (OLSR), as they were found useful for the thesis work.

Chapter 2

Ad Hoc On-Demand Distance Vector Routing (AODV)

2.1 AODV

2.1.1 Route Discovery in AODV:

- AODV discovers routes as and when necessary.
- Does not maintain routes from every node to every other.
- Routes are maintained just as long as necessary.
- Every node maintains its monotonically increasing sequence number -> increases every time the node notices change in the neighbourhood topology.
- When a node wishes to send a packet to some destination –

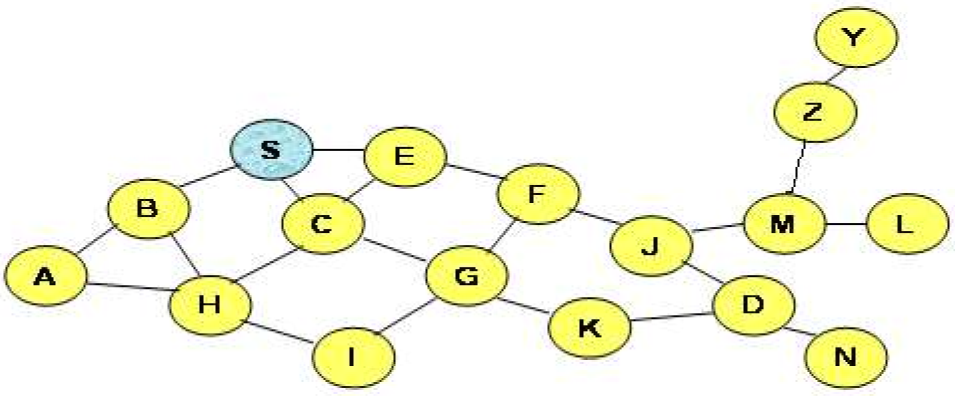
It checks its routing table to determine if it has a current route to the destination

  a. If Yes, forwards the packet to next hop node
  b. If No, it initiates a route discovery process

- Route discovery process begins with the creation of a Route Request (RREQ) packet -> source node creates it.
- The packet contains – source node's IP address, source node's current sequence number, destination IP address, destination sequence number.
- Packet also contains broadcast ID number.

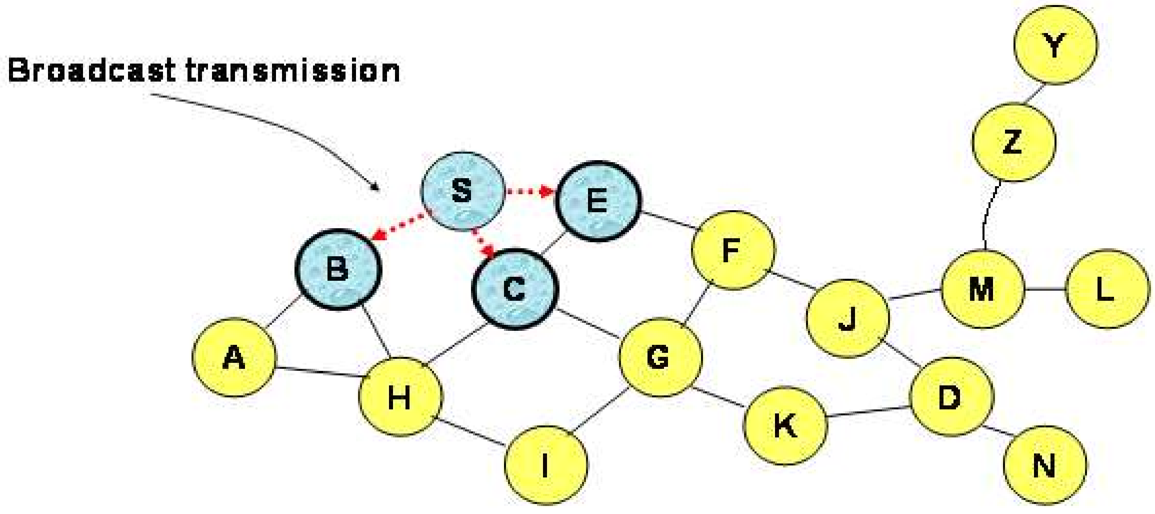
  Broadcast ID gets incremented each time a source node uses RREQ.
  - Broadcast ID and source IP address form a unique Identifier for the RREQ.
- Broadcasting is done via Flooding.

Route Discovery in AODV:



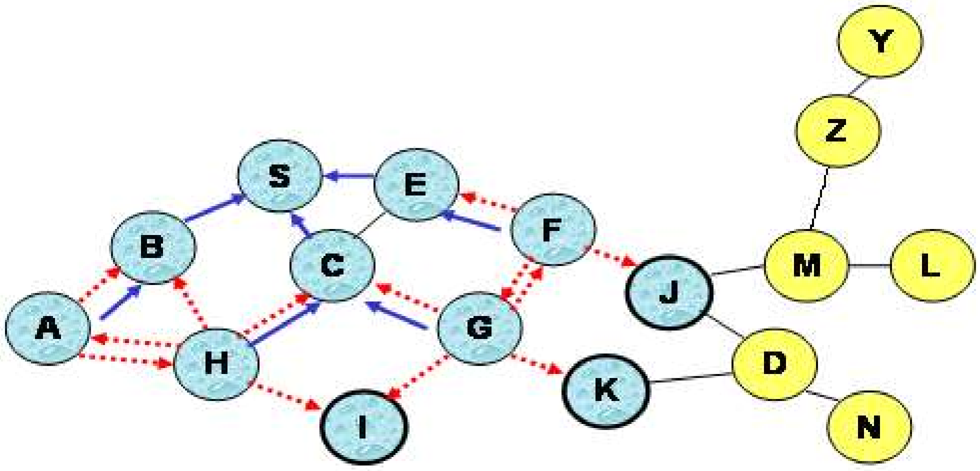
● Represents a node that has received RREQ for D from S.

Figure 2.1 : route discovery in AODV



⇢ Represents Transmission of RREQ.

← Represents links on reverse path.



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

2.1.2 Reverse Path Set-Up in AODV:

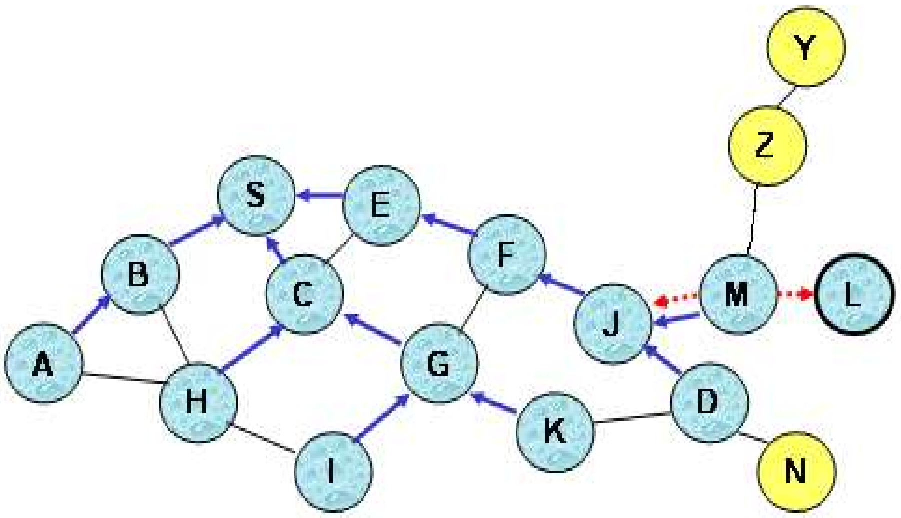


Figure 2.2: Reverse path set up in AODV

Node D does not forward RREQ, because Node D is the intended target of RREQ.

2.1.3 Forward path set-up in AODV:

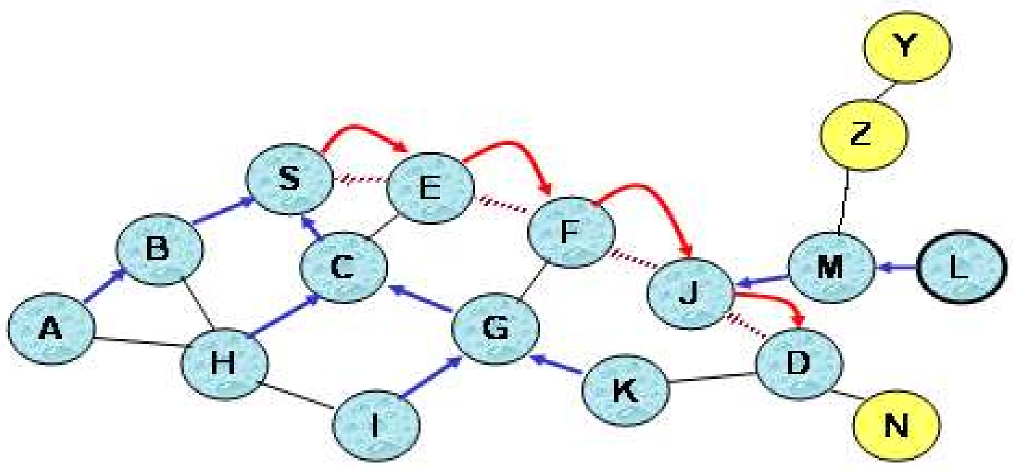


Figure 2.3: Forward path set up in AODV

Forward links are Set-Up when RREP travels along the reverse path.

➢ ⤻ Represents a link broken on the forward path.

2.2 PERFORMANCE ANALYSIS OF AODV:

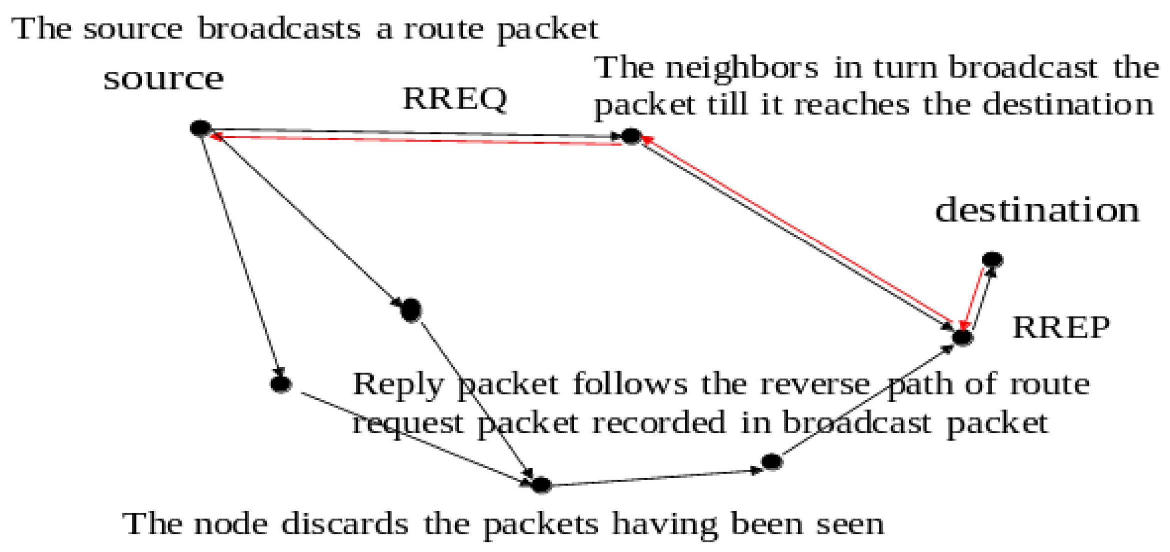
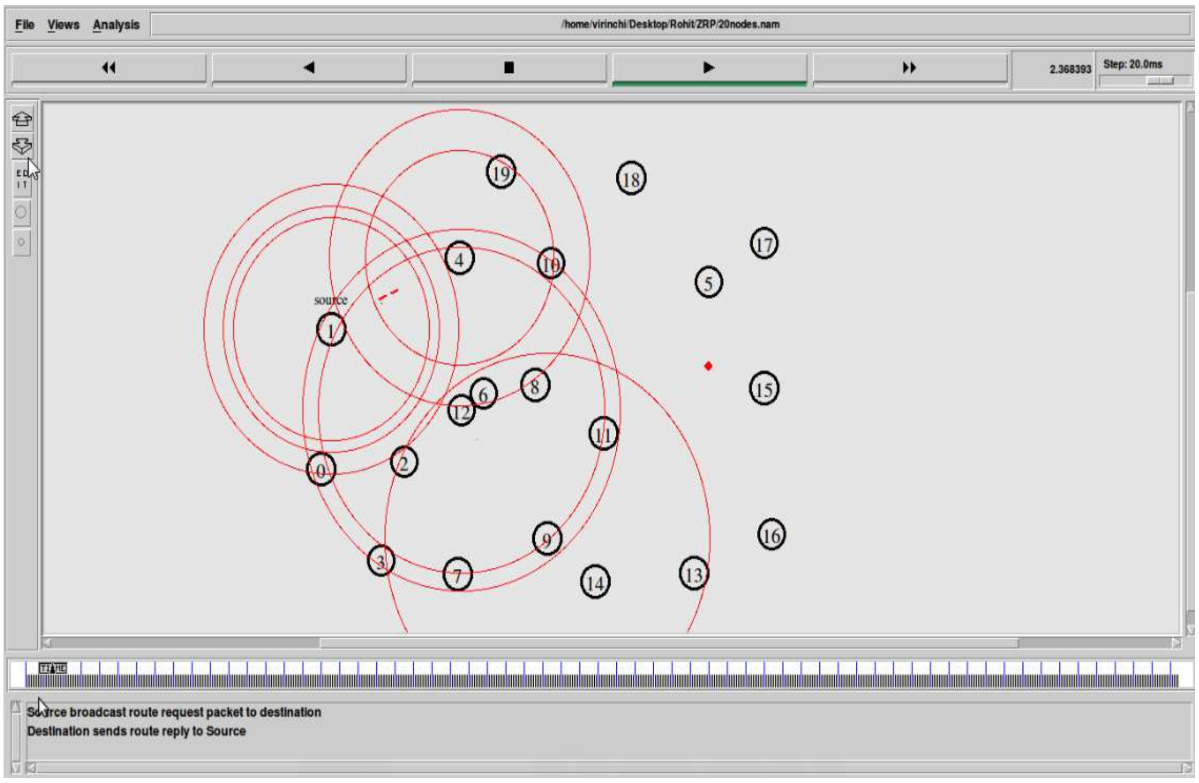

The source broadcasts a route packet
source
RREQ
The neighbors in turn broadcast the packet till it reaches the destination
destination
RREP
Reply packet follows the reverse path of route request packet recorded in broadcast packet
The node discards the packets having been seen

Figure 2.4: Performance analysis of AODV Protocol

The AODV Routing protocol uses an on-demand approach for finding routes, that is, a route is established only when it is required by a source node for transmitting data packets. It employs destination sequence numbers to identify the most recent path. The major difference between AODV and Dynamic Source Routing (DSR) stems out from the fact that DSR uses source routing in which a data packet carries the complete path to be traversed. However, in AODV, the source node and the intermediate nodes store the next-hop information corresponding to each flow for data packet transmission. In an on-demand routing protocol, the source node floods the Route Request packet in the network when a route is not available for the desired destination. It may obtain multiple routes to different destinations from a single Route Request. The major difference between AODV and other on-demand routing protocols is that it uses a destination sequence number (DestSeqNum) to determine an up-to-date path to the destination. A node updates its path information only if the DestSeqNum of the current packet received is greater than the last DestSeqNum stored at the node.

A Route Request carries the source identifier (SrcID), the destination identifier (DestID), the source sequence number (SrcSeqNum), the destination sequence number (DestSeqNum), the broadcast identifier (BcastID), and the time to live (TTL) field. DestSeqNum indicates the freshness of the route that is accepted by the source. When an intermediate node receives a Route Request, it either forwards it or prepares a Route Reply if it has a valid route to the destination. The validity of a route at the intermediate node is determined by comparing the sequence number at the intermediate node with the destination sequence number in the Route Request packet. If a Route Request is received multiple times, which is indicated by the BcastID-SrcID pair, the duplicate copies are discarded. All intermediate nodes having valid routes to the destination, or the destination node itself, are allowed to send Route Reply packets to the source. Every intermediate node, while forwarding a Route Request, enters the previous node address and it's BcastID. A timer is used to delete this entry in case a Route Reply is not received before the timer expires. This helps in storing an active path at the intermediate node as AODV does not employ source routing of data packets. When a node receives a Route Reply packet, information about the previous node from which the packet was received is also stored in order to forward the data packet to this next node as the next hop toward the destination.

2.2.1 SIMULATION OF AODV-20 NODES:



Source broadcast route request packet to destination
Destination sends route reply to Source

Fig 2.5

Chapter 3
BLACK HOLE ATTACK IN AODV PROTOCOL

MANETs face different securities threats i.e. attack that are carried out against them to disrupt the normal performance of the networks. These attacks are categorized in previous chapter "security issues in MANET" on the basis of their nature. In these attacks, black hole attack is that kind of attack which occurs in Mobile Ad-Hoc networks (MANET). This chapter describes Black Hole attack and other attacks that are carried out against MANETs.

3.1 Black Hole Attack:

In black hole attack, a malicious node uses its routing protocol in order to advertise itself for having the shortest path to the destination node or to the packet it wants to intercept. This hostile node advertises its availability of fresh routes irrespective of checking its routing table. In this way attacker node will always have the availability in replying to the route request and thus intercept the data packet and retain it. In protocol based on flooding, the malicious node reply will be received by the requesting node before the reception of reply from actual node; hence a malicious and forged route is created. When this route is establish, now it's up to the node whether to drop all the packets or forward it to the unknown address.

The method how malicious node fits in the data routes varies. Fig. shows how black hole problem arises, here node "A" want to send data packets to node "D" and initiate the route discovery process. So if node "C" is a malicious node then it will claim that it has active route to the specified destination as soon as it receives RREQ packets. It will then send the response to node "A" before any other node. In this way node "A" will think that this is the active route and thus active route discovery is complete. Node "A" will ignore all other replies and will start seeding data packets to node "C". In this way all the data packet will be lost consumed or lost.

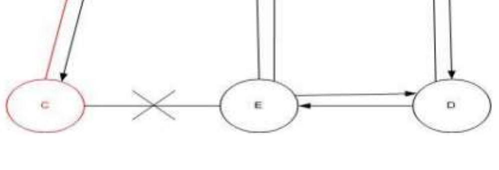


Figure 3.1: Black hole attack in MANET's

3.2 Black hole attack in AODV:

Two types of black hole attack can be described in AODV in order to distinguish the kind of black hole attack.

Internal Black hole attack:

This type of black hole attack has an internal malicious node which fits in between the routes of given source and destination. As soon as it gets the chance this malicious node make itself. an active data route element. At this stage it is now capable of conducting attack with the start of data transmission. This is an internal attack because node itself belongs to the data route. Internal attack is more vulnerable to defend against because of difficulty in detecting the internal misbehaving node.

External Black hole attack

External attacks physically stay outside of the network and deny access to network traffic or creating congestion in network or by disrupting the entire network. External attack can become a kind of internal attack when it take control of internal malicious node and control it to attack other nodes in MANET. External black hole attack can be summarized in following points

1. Malicious node detects the active route and notes the destination address.
2. Malicious node sends a route reply packet (RREP) including the destination address field spoofed to an unknown destination address. Hop count value is set to lowest values and the sequence number is set to the highest value.
3. Malicious node send RREP to the nearest available node which belongs to the active route. This can also be send directly to the data source node if route is available.
4. The RREP received by the nearest available node to the malicious node will relayed via the established inverse route to the data of source node.
5. The new information received in the route reply will allow the source node to update its routing table.
6. New route selected by source node for selecting data.
7. The malicious node will drop now all the data to which it belong in the route.

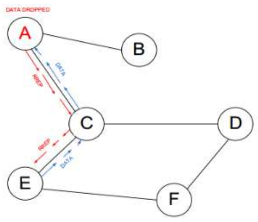
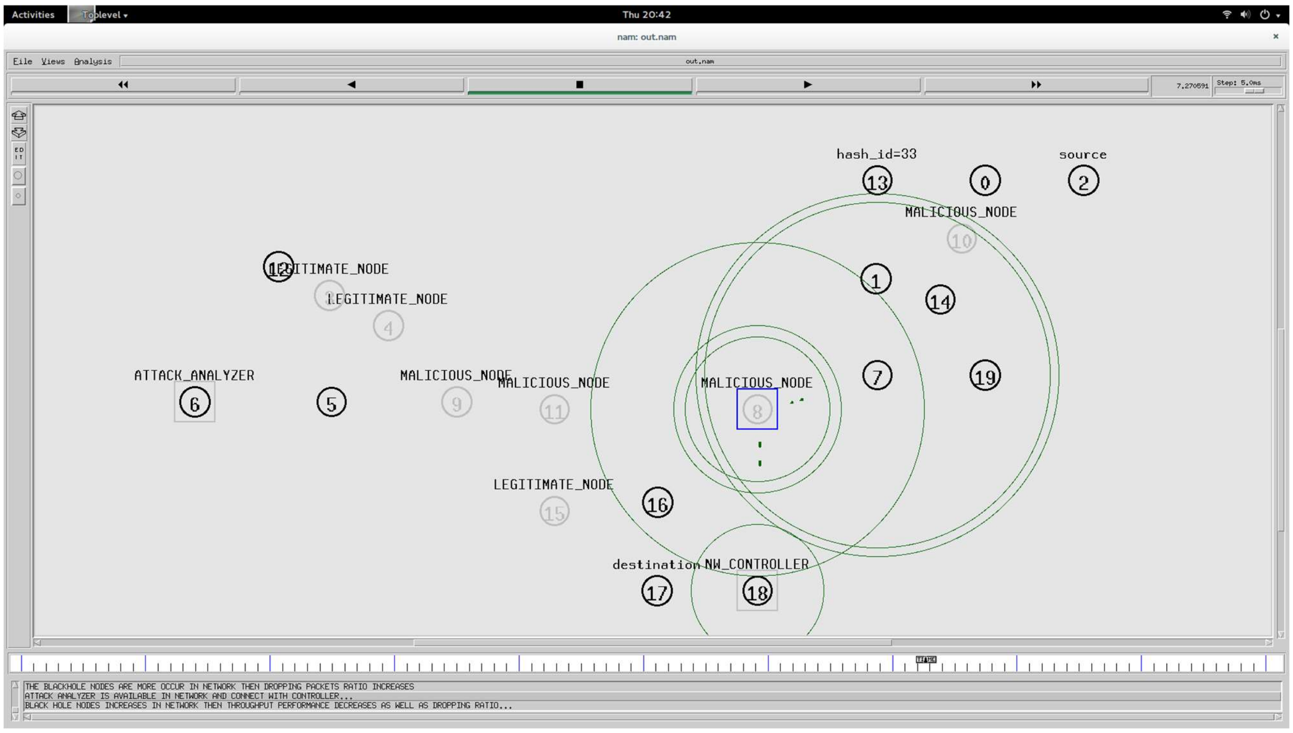


Figure 3.2: Black hole attack in AODV protocol

In AODV black hole attack the malicious node “A” first detect the active route in between the sender “E” and destination node “D”. The malicious node “A” then send the RREP which contains the spoofed destination address including small hop count and large sequence number than normal to node “C”. This node “C” forwards this RREP to the sender node “E”. Now this route is used by the sender to send the data and in this way data will arrive at the 21 malicious node. These data will then be dropped. In this way sender and destination node will be in no position any more to communicate in state of black hole attack.

3.3 SIMULATION OF AODV-20 NODES IMPLEMENTING BLACK HOLE ATTACK:



```
THE BLACKHOLE NODES ARE MORE OCCUR IN NETWORK THEN DROPPING PACKETS RATIO INCREASES
ATTACK ANALYZER IS AVAILABLE IN NETWORK AND CONNECT WITH CONTROLLER...
BLACK HOLE NODES INCREASES IN NETWORK THEN THROUGHPUT PERFORMANCE DECREASES AS WELL AS DROPPING RATIO...
```

FIG 3.3

Chapter 4

NETWORK SIMULATOR 2

4.1 INTRODUCTION:

A network simulator is a piece of software that predicts the behavior of a network, without an actual network being present. It is a software program that imitates the working of a computer network. Network simulators, as the name suggests are used by researchers, developers and engineers to design various kinds of networks, simulate and then analyze the effect of various parameters on the network performance. In simulators, the computer network is typically modeled with devices, traffic etc. and the performance is analyzed. Users can then customize the simulator to fulfill their specific analysis needs. It implements network protocols such as TCP and UPD, traffic source behavior such as FTP, Telnet, Web, CBR and VBR, router queue management mechanisms such as FIFO, RED and CBQ, routing algorithms such as Dijkstra, and more. NS also implements multicasting and some of the MAC layer protocols for LAN simulation.

There are various real hardware tools and emulators are available in the market for measuring the performance of a network wired or wireless. But one cannot invest much in terms of money and time. Hence there should be some tools that help the researchers to do their research to measure the performance and find the results. The network simulators in these circumstances save a lot of money and time in accomplishing this task. Network simulators are also particularly useful in allowing the network designers to test new networking protocols or to change the existing protocols in a controlled and reproducible manner.
There are a wide variety of network simulators, ranging from the very simple to the very complex. Minimally, a network simulator must enable a user to represent a network topology, specifying the nodes on the network, the links between those nodes and the traffic between the nodes. More complicated systems may allow the user to specify everything about the protocols used to handle traffic in a network. Graphical applications allow users to easily visualize the workings of their simulated environment.

4.1.1 Features of NS2:

- NS-2 is a discrete event driven and object oriented simulator.
- Simulation of wired as well as wireless networks and protocols (e.g. routing algorithms, TCP, UDP) can be done using NS2.
- It is a packet level simulation.
- Simulates almost any type of networks which are in great demand by the industry.
- Back end is C++ event scheduler especially for protocols mostly
- Front end is OTCL

  -Creating scenarios, extensions to C++ protocols.
  -Objects created in OTCL have a corresponding object in C++.

- Transport protocols: Unicast-TCP, Multicast-UDP, etc.
- Queuing protocols: RED, Drop Tail, DRR, FQ, SFQ ,etc.
- Traffic /Application models: CBR, FTP, TELNET
- Physical Media: Wired (point to point LANs), Wireless (Ad hoc, Mobile, WLAN, Bluetooth, Satellite).
- Measurement of Statistics:

i) Throughput, Delay, Jitter, etc.
ii) Queue monitoring, Drops at queues.
iii) All parameters that characterize the working of various protocols.

- Complex scenarios can be easily tested.
- It is cheap- Doesn’t require costly equipment.

Programming in NS2:

NS simulator is based on two languages:
1) An object oriented simulator, written in C++.
2) Tcl/OTcl(Object Tool Command Language-an object extension of Tcl) interpreter, used to execute user's command scripts.

From the users prospective, NS2 is an OTcl interpreter that takes an OTcl script as input and produces a trace file as output.

4.1.2 User's view of NS:

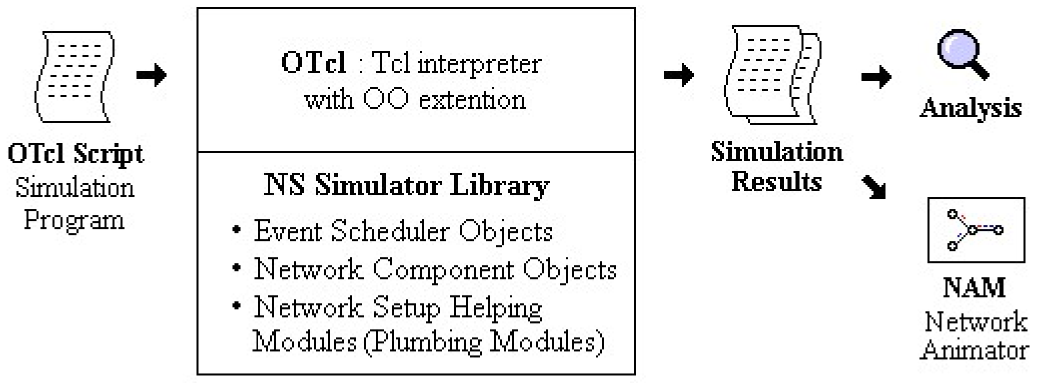

OTcl Script
Simulation Program
OTcl : Tcl interpreter with OO extention
NS Simulator Library
• Event Scheduler Objects
• Network Component Objects
• Network Setup Helping Modules (Plumbing Modules)
Simulation Results
Analysis
NAM
Network Animator

Figure 4.1 Simplified view of Network Simulator

NS-2 uses two languages because simulator has two different kinds of things it needs to do.

- Detailed simulation of protocols requires a systems programming language which can efficiently manipulate bytes, packet headers, and implement algorithms that run over large data sets. For these tasks run-time speed is important and turn-around time (run simulation, find bug, fix bug, recompile, re-run) is less important.

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

- NS-2 meets both of these needs with two languages, C++ and OTcl.

- **C++:**Detailed protocol simulations require systems programming language

1. Byte manipulation, packet processing, algorithm implementation
2. Run time speed is important
3. Turn around time (run simulation, find bug, fix bug, recompile, re-run) is slower

C++ is fast to run but slower to change, making it suitable for detailed

protocol implementation.

- **OTcl:** Simulation of slightly varying parameters or configurations

    1. Quickly exploring a number of scenarios
    2. Iteration time (change the model and re-run) is more important .

OTcl runs much slower but can be changed very quickly (and interactively), making it ideal for simulation configuration.

C++:

In order to reduce packet and event processing time (not simulation time), the event scheduler and the basic network component objects in the data path are written and compiled using C++. These compiled objects are made available to the OTcl interpreter through an OTcl linkage that creates a matching OTcl object for each of the C++ objects and makes the control functions and the configurable variables specified by the C++ object act as member functions and member variables of the corresponding OTcl object. In this way, the controls of the C++ objects are given to OTcl. It is also possible to add member functions and variables to a C++ linked OTcl object. The objects in C++ that do not need to be controlled in a simulation or internally used by another object do not need to be linked to OTcl. Likewise, an object (not in the data path) can be entirely implemented in OTcl. The compiled C++ hierarchy allows us to achieve efficiency in the simulation and faster execution times.

OTcl:

NS is Object-oriented Tcl (OTcl) script interpreter that has a simulation event scheduler and network component object libraries. To setup and run a simulation network, a user should write an OTcl script that initiates an event scheduler, sets up the network topology using the network objects and tells traffic sources when to start and stop transmitting packets through the event scheduler. .The OTcl script can make use of the objects compiled in C++ through an OTcl linkage that creates a matching of OTcl object for each of the C++.

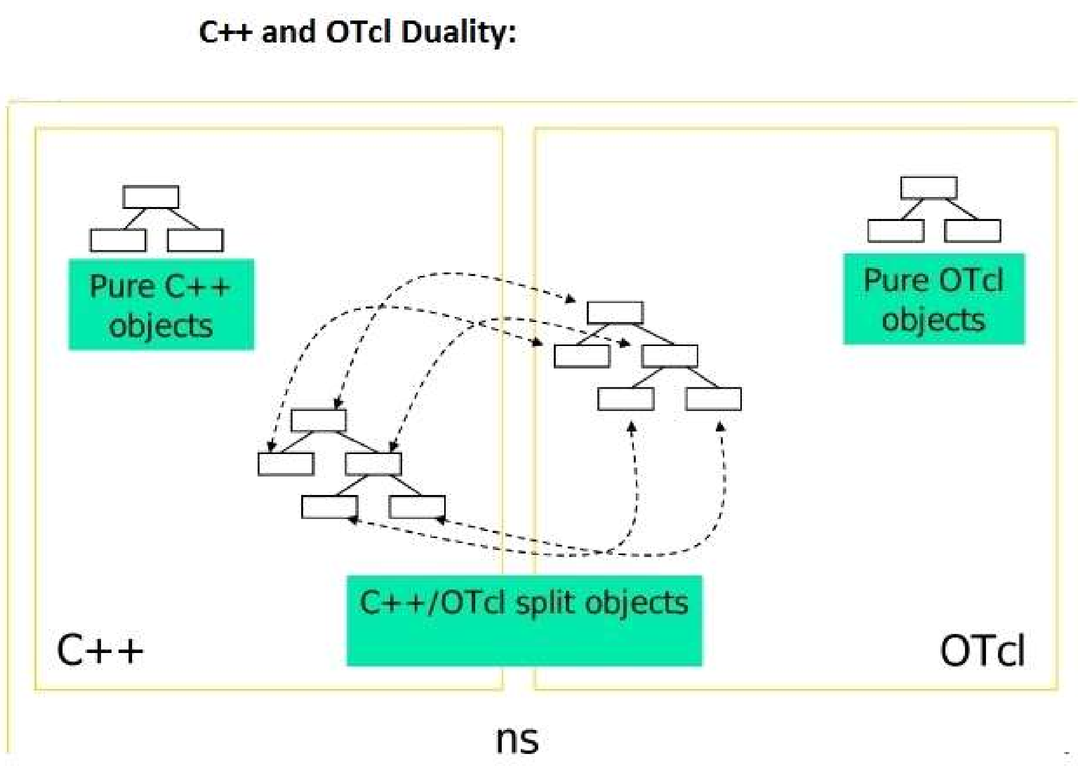


Figure 4.2: Duality between C++ and OTcl

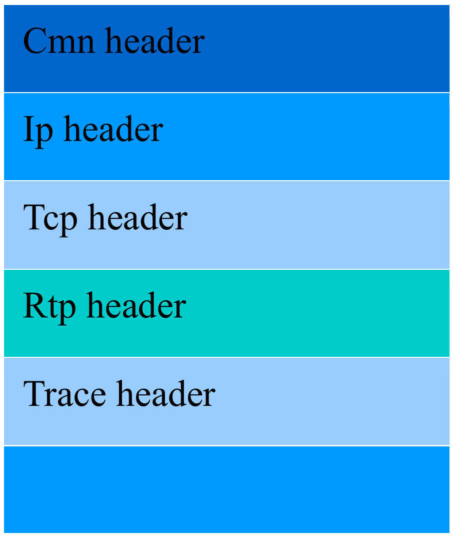
4.1.3 Scheduler:

NS is a discrete event simulator, where the advance of time depends on timing of events which are maintained by a scheduler. An event in NS is a packet ID that is unique for a packet with scheduled time and the pointer to an object that handles the event. In NS, an event scheduler keeps track of simulation time and fires all the events in the event queue scheduled for the current time by invoking appropriate network components, which usually are the ones who issued the events, and let them do the appropriate action associated with packet pointed by the event. Network components communicate with one another passing packet. All the network components that need to spend some simulation time handling a packet (i.e. need a delay) use the event scheduler by issuing an event for the packet and waiting for the event to be fired to itself before doing further action handling the packet.

Another use of an event scheduler is timer. For example, TCP needs a timer to keep track of a packet transmission time out for retransmission (transmission of a packet with the same TCP packet number but different NS packet ID). Timers use event schedulers in a similar manner that delay does. The only difference is that timer measures a time value associated with a packet and does an appropriate action related to that packet after a certain time goes by, and does not simulate a delay.

4.1.4 PACKETS:

It is the collection of data, whether header is called or not all header files where present in the stack registers.



4.1.5 Architecture of NS:

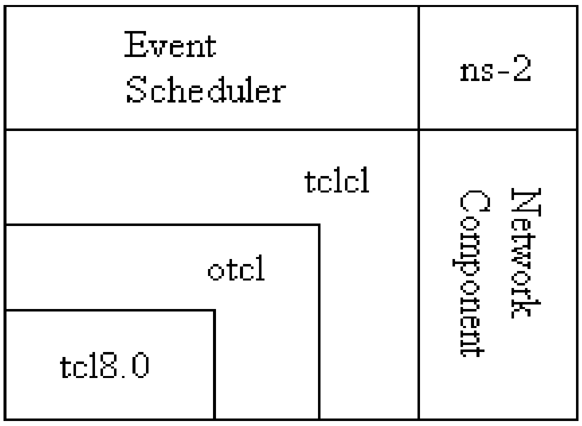


Figure 4.3 Architecture of Network Simulator

In this figure a general user can be thought of standing at the left bottom corner, designing and running simulations in Tcl using the simulator objects in the OTcl library. The event schedulers and most of the network components are implemented in C++ and available to OTcl through an OTcl linkage that is implemented using **tclcl**. The whole thing together makes NS, which is aextended Tcl interpreter with network simulator libraries.

4.1.6 NS2 Directory Structure:

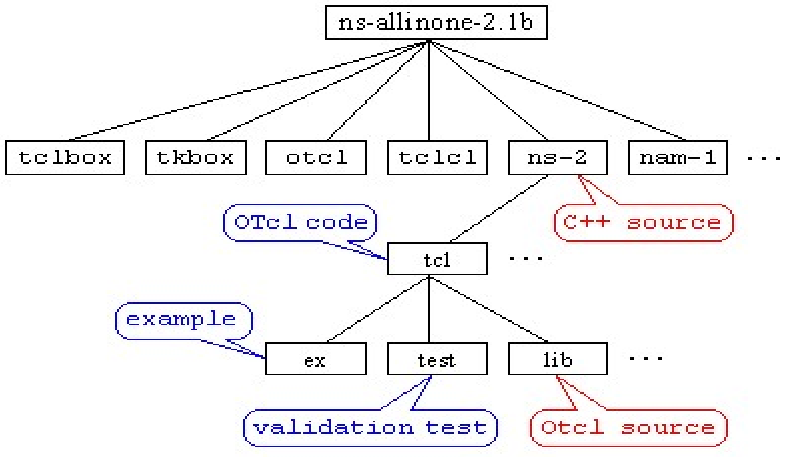


Figure 4.4 a) Directory Structure of Network Simulator

Among the sub-directories of ns-all-in-one, ns-2 is the place that has all of the simulator implementations (either in C++ or in OTcl), validation test OTcl scripts and example OTcl scripts. Within this directory, all OTcl codes and test/example scripts are located under a sub-directory called tcl and most of C++ code, which implements event scheduler and basic network component object classes, except the WWW related ones, are located in the main level. For example, if you want to see the implementation of the UDP agent, you should go to "ns-allinone-2.1b/ns-2" directory, and open up "udp.h", "udp.cc" and the files that contain the implementation of ancestor classes of UDP as needed.

- The tcl directory has sub-directories, among which the lib directory that contains OTcl source codes for the most basic and essential parts of the NS implementation (agent, node, link, packet, address, routing, and etc.)

4.2 Procedure for Simulation And Creating Topologies In Ns2:

Basically NS 2 simulation contains the following steps:

1. Create the simulator object

2. Turn on tracing

3. Create topologies:

a) Set node and link configurations

b) Create transport connection-Agents

c) Create traffic-Applications

4. Set packet size and transmission intervals

5. Define events and scheduling times.
6. Run the simulator
7. Monitoring:
   -Visualization using nam
8. Result analysis:
   -Script trace analysis: often in awk, perl or Tcl.

Create a simulator object:

An NS simulation starts with the command:

set ns [new Simulator]

Which is the first line in the tcl script.

Creating nodes:

Figure 4.4 b) : Creating Nodes in Network Simulator

The way to define a node in NS is

set n0 [$ns node]

set n1 [$ns node]

By this definition, nodes are created that are pointed by variables **n0** and **n1.**

Creating link between nodes:

Once nodes are defined in the network, we can define the links that connect them. An example of defining a link is

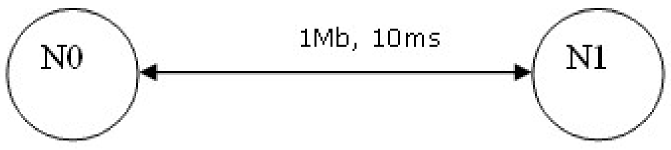


Figure 4.4 c) Connecting two nodes in Network Simulator

$ns duplex-link $n0 $n1 1Mb 10ms Drop Tail

This line tells the simulator object to connect the nodes **n0** and **n1** with a duplex link(indicated by double sided arrow) with the bandwidth 1 Megabit, a delay of 10ms and a 'Drop Tail' queue. To define a uni directional link instead of a bi-directional one, we should replace 'duplex link' by 'simplex link'.

In NS, an output queue of a node is implemented as a part of each link whose input is that node. The definition of the link then includes the way to handle overflow at that queue.

The Buffer capacity of the queue can be defined by:

```
$ns queue-limit $n0 $n1 10
```

This defines the queue capacity between node n0 and n1 as 10. Different links can be created by using different combinations of bandwidth, delay and type of queue.

A link in NS:

When a user creates a link using a duplex-link member function of a Simulator object, two simplex links in both directions.

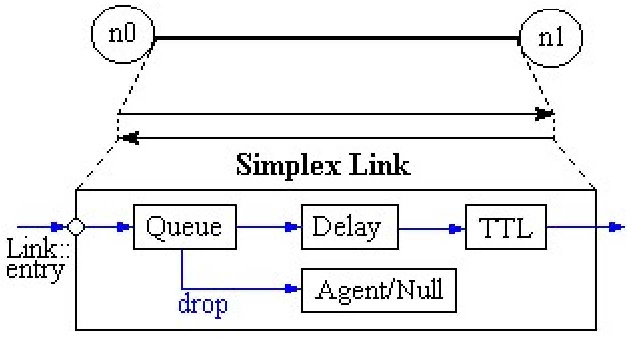


Figure 4.5 A Link in Network Simulator

One thing to note is that an output queue of a node is actually implemented as a part of simplex link object. Packets dequeued from a queue are passed to the Delay object that simulates the link delay, and packets dropped at a queue are sent to a Null Agent and are freed there. Finally, the TTL object calculates Time To Live parameters for each packet received and updates the TTL field of the packet.

Creating Transport agents:

Once the topology is defined, we should allow traffic to flow through them. For that we need to define routing between source and destination and the agents (protocols) between them. Examples of agents are TCP, UDP, TELNET, CBR, etc.

Transfer Control Protocol: TCP is a dynamic reliable congestion protocol which is used to provide reliable transport of packets from one host to another host by sending acknowledgements on proper transfer or loss of packets. Thus TCP requires bi-directional links in order for acknowledgements to return to the source.

Now the set up TCP connection between two nodes done as follows:

```
                              #setting a tcp connection
settcp [new Agent/TCP]
                                     $ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
                                     $ns attach-agent $n4 $sink
                                     $ns connect $tcp $sink
                                    $tcp set fid_1
                                    $tcp set packetSize_552
```

The command 'set tcp [new Agent/TCP]' gives a pointer called 'tcp' which indicates the tcp agent which is a object of ns. Then the command '$ns attach-agent $n0 $tcp' defines the source node of tcp connection. Next the command 'set sink [new Agent/TCPSink]' defines the destination of tcp by a pointer called sink. The next command '$ns attach-agent $n4 $sink' defines the destination node as n4.Next, the command '$ns connect $tcp $sink' makes the TCP connection between the source and the destination. i.e n0 and n4.When we have several flows such as TCP, UDP etc in a network. So, to identify these flows we mark these flows by using the command '$tcp set fid_1'. In the last line we set the packet size of tcp as 552 while the default packet size of tcp is 1000.

File Transfer Protocol (FTP over TCP):

File Transfer Protocol (FTP) is a standard mechanism provided by the Internet for transferring files from one host to another. Well this is the most common task expected from a networking or a inter networking. FTP differs from other client server applications in that it establishes between the client and the server. One connection is used for data transfer and other one is used for providing control information. FTP uses the services of the TCP. It needs two connections. The well Known port 21 is used for control connections and the other port 20 is used for data transfer.

Run a FTP connection over a TCP:

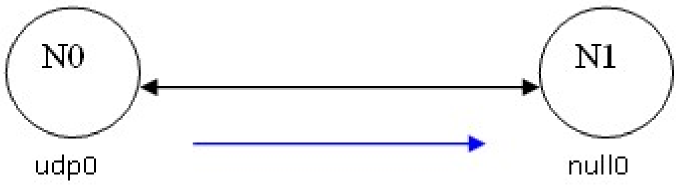
```
#Initiating FTP over TCP
set ftp [new Application/FTP]
                              $ftp attach-agent $tcp
```

In above, the command 'set ftp [new Application/FTP]' gives a pointer called 'ftp' which indicates the FTP application. Next, we attach the ftp application with tcp agent as FTP uses the services of TCP.

USER DATAGRAM PROTOCOL (UDP):

The User datagram Protocol is one of the main protocols of the Internet protocol suite. UDP helps the host to send send messages in the form of datagrams to another host which is present in a Internet protocol network without any kind of requirement for channel transmission setup. UDP provides a unreliable service and the datagrams may arrive out of order, appear duplicated, or go missing without notice. UDP assumes that error checking and correction is either not necessary or performed in the application, avoiding the overhead of such processing at the network interface level. Time-sensitive applications often use UDP because dropping packets is preferable to waiting for delayed packets, which may not be an option in a real-time system.

i) Communication using a UDP agent:



In UDP(User Datagram Protocol)communication, data is flows from UDP agent to Null agent.

#Create a UDP agent and attach it to node n0

```
set udp0 [new Agent/UDP]
            $ns attach-agent $n0 $udp0
```

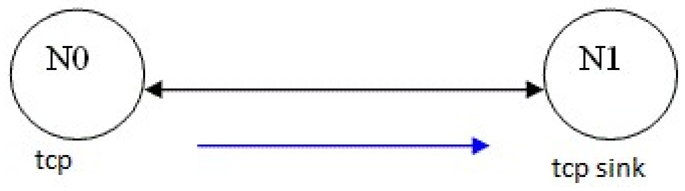
# create a null agent which act as traffic sink and attach it to node n1

```
set null0 [new Agent/Null]
           $ns attach-agent $n1 $null0
```

# connect two agents with each other

```
           $ns connect $udp0 $null0
```

ii) Communication using a TCP agent:



In TCP (Transfer Control Protocol) communication, data flows from TCP agent to TCP sink agent.

# create Tcp agent and attach it to node n0

```
set tcp0 [new Agent/TCP]
        $ns attach-agent $n0 $tcp0
```

# create a TCPsink agent which act as traffic sink and attach it to node n1

```
set tcpsink0 [new Agent/TCPSink]
        $ns attach-agent $n1 $tcpsink0
```

# connect two agents with each other

```
          $ns connect $tcp0 $tcpsink0
```

Creating Application Traffic:

For actual data to flow, we need traffic generators. They simulate some application traffic. In general CBR(Constant Bit Rate) agent is used over UDP connection. A simple example using CBR:

Constant Bit Rate (CBR):

Constant Bit Rate (CBR) is a term used in telecommunications, relating to the quality of service. When referring to codec's, constant bit rate encoding means that the rate at which a codec's output data should be consumed is constant. CBR is useful for streaming multimedia content on limited capacity channels since it is the maximum bit rate that matters, not the average, so CBR would be used to take advantage of all of the capacity. CBR would not be the optimal choice for storage as it would not allocate enough data for complex sections (resulting in degraded quality) while wasting data on simple sections.

# creating CBR agent

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

# Attach the CBR agent to some udp/tcp agent

```
        $cbr0 attach-agent $udp0 $cbr0
settype_CBR $cbr;   (used by Nam)
```

#Defining the packet size and rate:

```
set packet_size_1000 $cbr
set rate_1mb $cbr
```

Similarly, an FTP(File Transfer Protocol) agent is used over TCP connection:

```
set ftp [new Application/FTP]
$ftp attach-agent $tcp
```

The difference CBR and FTP is that CBR traffic generator will produce constant bit rate whereas FTP traffic generator produces maximum available bit rate.

Scheduling the events:

The Tcl script defines when event should occur. The initialising command set ns [new Simulator] creates an event scheduler, and events are then scheduled using the format:

```
$ns at <time><event>
```

Example:

```
$ns at 1.0 “$cbr0 start”
$ns at 5.0 “finish"
```

cbr0 will start at a time of 1.0 ms and whole process will stops at 5.0ms.we can also stop each and traffic generator. for example

```
$ns at 4.0 ”$cbr0 stop”
```

Traffic generator cbr0 will stops at 4.0

Run the simulation:

The scheduler is started using the command

```
$ns run
```

The Trace File:

Using Trace files:

There are two types of trace files:

1. generic trace: for use with xgraph, and other things.
2. nam trace: for use with visualization.

In order to have output files with data on the simulation (trace fles) or files used for visualisation (nam files), we need to create the files using the "open" command.

# Open trace file (generic)

```
settracefile [open out.tr w]
        $ns trace-all $tracefile
```

#Open the nam trace file (nam)

```
setnf [open out.nam w]
       $ns namtrace-all $nf
```

The above creates a data trace file called "out.tr" and a namvisualisation trace file (for the NAM tool) called "out.nam".

Structure of Trace files:

The trace file can be viewed with the cat command: cat out.tr

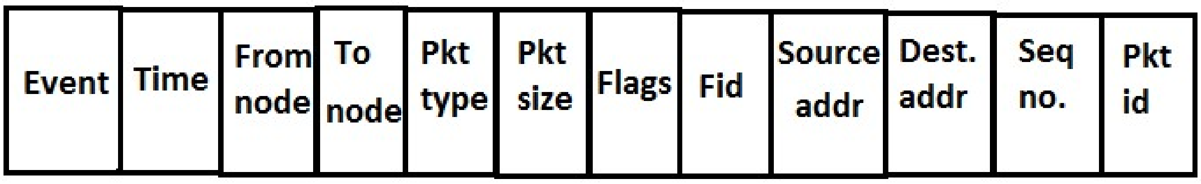
- Each line in the trace file is in the format:

<event>< time >< from ><to><pkt − type ><pkt − size >< flags >< fid ><src.port><dst.port><seq>< unique pkt id >

• Trace example:

```
+ 1 0 2 cbr 210 ------- 0 0.0 3.1 0 0
- 1 0 2 cbr 210 ------- 0 0.0 3.1 0 0
r 1.00234 0 2 cbr 210 ------- 0 0.0 3.1 0 0
```

• **Event**: s send, r receive, + enqueue, − dequeue, d drop, f forward,



Event	Time	From node	To node	Pkt type	Pkt size	Flags	Fid	Source addr	Dest. addr	Seq no.	Pkt id

Figure 4.6 Trace File Format

5. **EVENT:** The first field is event. It gives you four possible symbols '+' '-' 'r' 'd'. These four symbols correspond respectively to enqueued, dequeued, received and dropped.

6. **TIME**: The second field gives the time at which the event occurs.

7. **FROM NODE:** The third field gives you the input node of the link at which the event occurs.

8. **TO NODE:** The fourth field gives you the the output node at which the event occurs.

9. **PACKET TYPE**: The fifth field shows the information about the packet type.i.e whether the packet is UDP or TCP

10. **PACKET SIZE:** The sixth field gives the packet size.

11. **FLAGS:** The seventh field gives information about some flags.

12. **FLOW ID:** The eight field is the flow id(fid) for IPv6 that a user can set for each flow in a tcl script. It is also used for specifying the color of flow in NAM display.

13. **SOURCE ADDRESS:** The ninth field is the source address.

14. **DESTINATION ADDRESS**: The tenth field is the destination address.

15. **PACKET SEQUENCE NUMBER**: The eleventh field is the network layer protocol's packet sequence number.

16. **PACKET ID:** The last field shows the unique id of packet.

4.3 NAM- Network Animator:

NAM is a Tcl/TK based animation tool for viewing network simulation traces and real world packet trace data. The design theory behind nam was to create an animator that is able to read large animation data sets and be extensible enough so that it could be used indifferent network visualization situations. It supports topology layout, packet level animation, and various data inspection tools. The first step to use nam is to produce the trace file. The trace file contains topology information, e.g., nodes, links, as well as packet traces.

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, and then pause at time 0.

Things that can be done using NAM:

- Colouring nodes
- Shape of nodes
- Colouring links
- Adding and removing marks
- Adding labels
- Adding text
- Monitoring of queue size

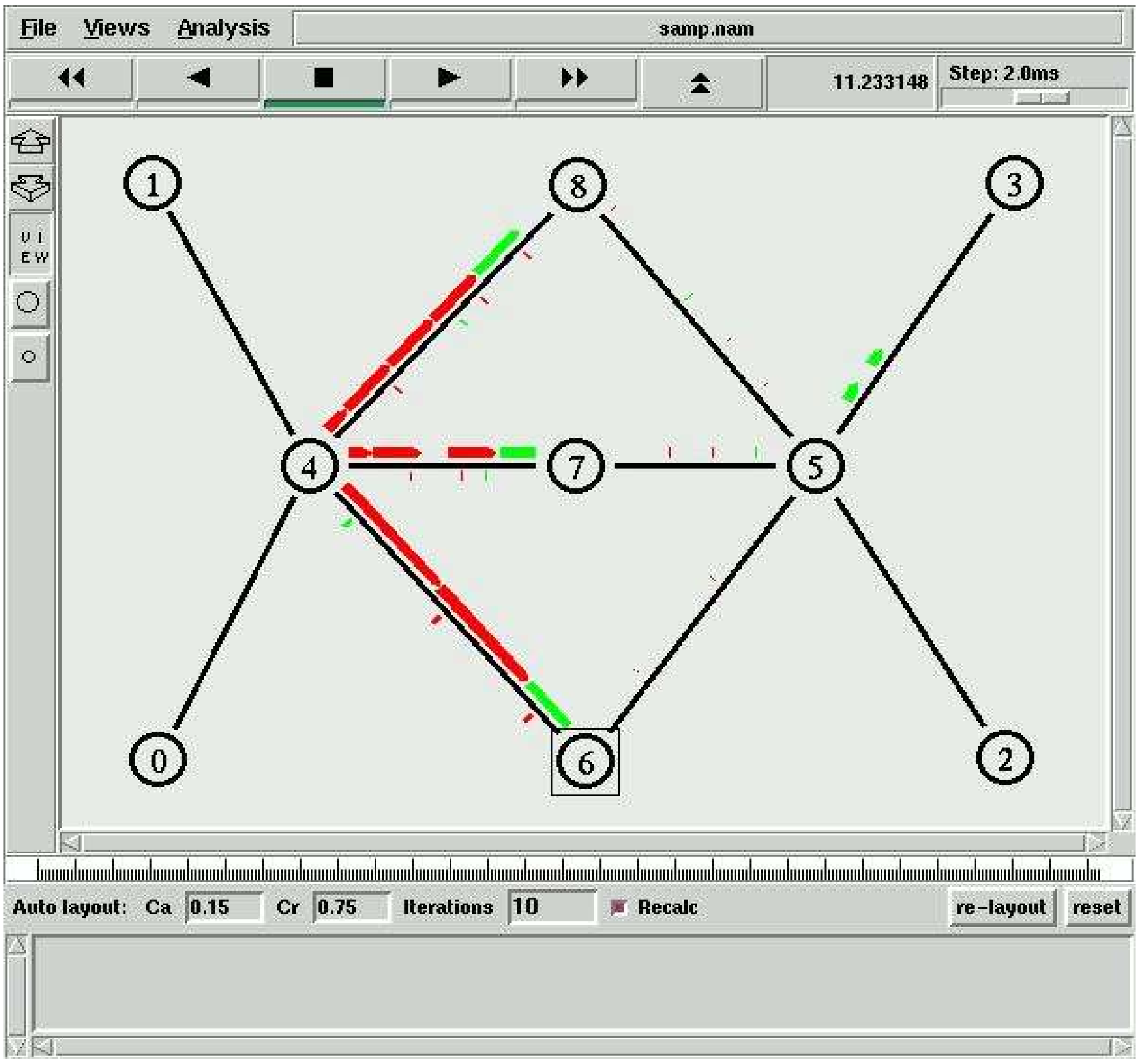


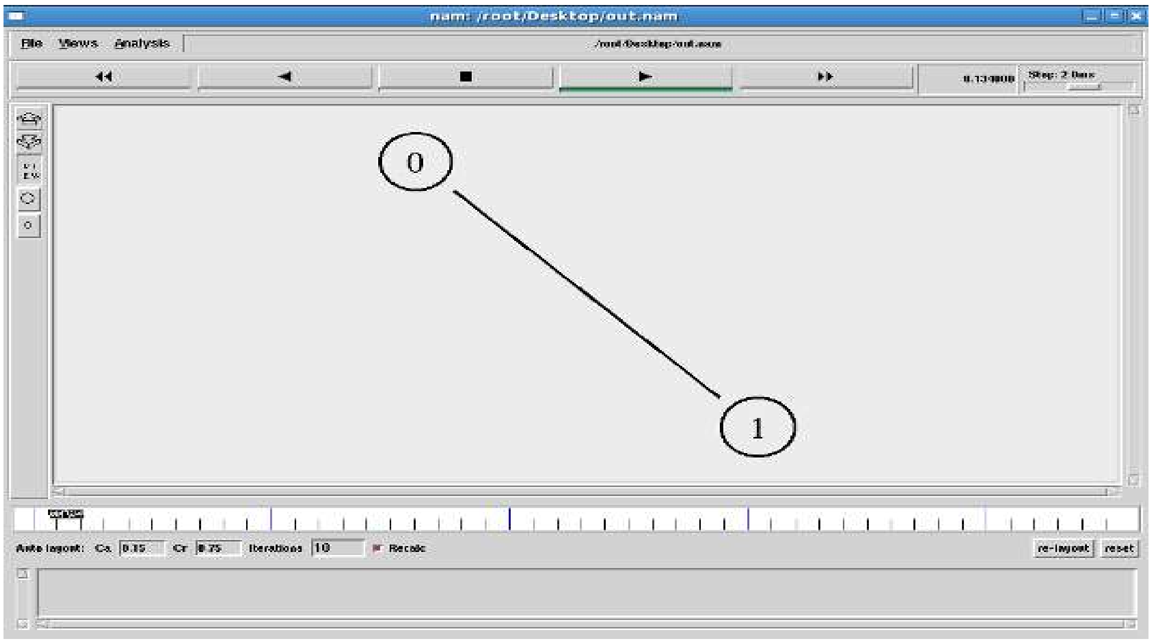
Figure 4.7: Visualisation of Network using NAM

Example :(CBR over UDP)

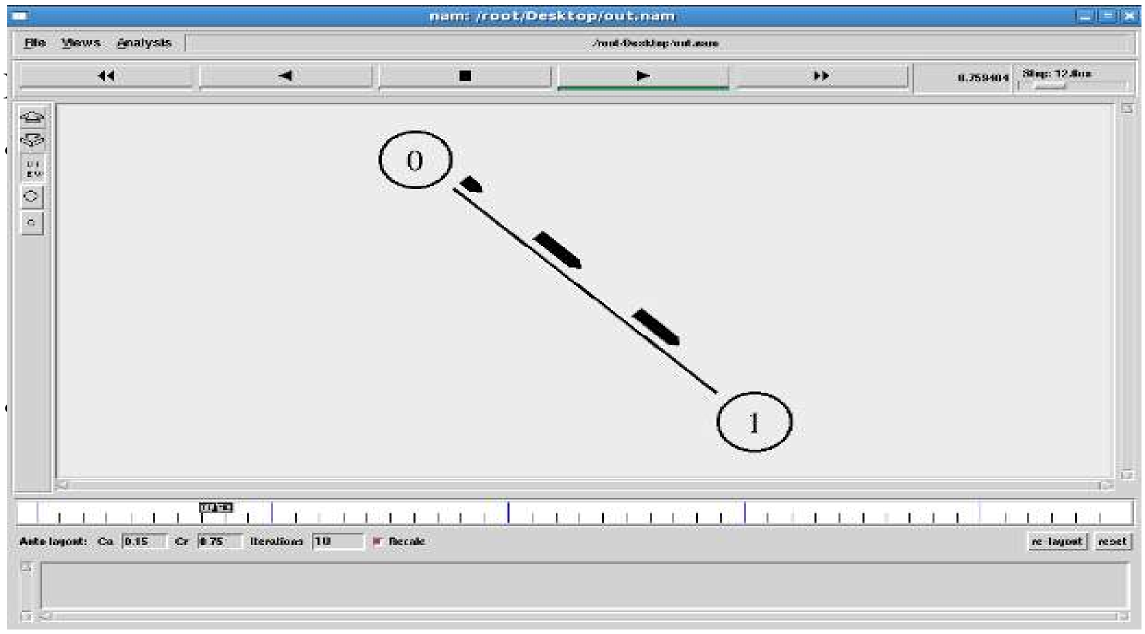
```
set ns [new Simulator]----------Creating  ns simulator object
settracefile [open out.tr w]-----------Open trace file
$ns trace-all $tracefile
setnf [open out.nam w]-------------Open the nam trace file
$ns namtrace-all $nf
proc finish {}------------- 'Finish' procedure
{
global ns tracefilenf
     $ns flush-trace
close $nf
close $tracefile
execnamout.nam&
exit 0
}
set n0 [$ns node]-----------Create your topology
set n1 [$ns node]
$ns simplex-link $n0 $n1 1Mb 10ms DropTail
set udp0 [new Agent/UDP]----------Create your agents(transport and apllication layer)
$ns attach-agent $n0 $udp0
setcbr[new Application/Traffic/CBR]
$cbr attach-agent $udp0
set null0 [new Agent/Null]
$ns attach-agent $n1 $null0
$ns connect $udp0 $null0
$ns at 1.0 "$cbr start"---------Scheduling Events
$ns at 3.0 "finish"              (ns starts at 1.0 and finishes at 3.0)
- $ns at 1.0 start
and at 3.0 finish
$ns run------------- starts the simulation
```

Result:

Before 1.0ms



After .0 ms



Conditions for getting efficiency in X-graph:

```
if ( event == "+" && source =="0.0")
sendspkt++;
if ( event == "r" &&dest == "3.0" )
recevepkt++;
if ( event == "d" &&pkt == "tcp")
drop++;
if ( seq_no>maxseq_no ) maxseq_no = seq_no;
if ( ! ( seq_no in timeIn ) ) timeIn[seq_no] = time;
if ( event != "d" ) {
if ( event == "r" ) timeFim[seq_no] = time;

  } else timeFim[seq_no] = 0;

}
```

4.4 MOTIVATION FOR SIMULATIONS:

1. Cheap does not require costly equipment
2. Complex scenarios can be easily tested
3. Results can be quickly obtained – more ideas can be tested in a smaller timeframe
4. The real thing isn't yet available
5. Controlled experimental condition– Repeatability helps aid debugging
6. Disadvantages: Real systems too complex to model.

CHAPTER-5

SIMULATION SETUP AND RESULTS:

5.1 Simulation parameters are as follows:

The IEEE 802.11 protocol is used as the MAC layer protocol. The radio channel model follows a Two Ray Ground with a bit rate of 2 Mbps and the antenna used is Omni directional antenna. We consider constant bit rate (CBR) data traffic and randomly choose different source-destination connections. Every source sends two CBR packets whose size is 1MB per second. The Drop Tail Queue management scheme is used in every link of the network. The mobility model is based on the random waypoint model in a field of 1000m*1000m. In this mobility model, each node moves to a random selected destination with a constant speed from a uniform distribution amongst the nodes. After the node reaches its destination, it stops for a pause-time

Simulation Parameter	**Value**
Simulator	NS-2
Routing Protocol	AODV
Topology size	1000 * 1000
Number of nodes	20
Traffic type	FTP / CBR
Traffic rate	1Mb
Antenna	OMNI DIRECTIONAL

interval and chooses a new destination and constant speed. The detailed simulation parameters are shown in the table.

TABLE- 1

USEFUL FILES:

(1) TRACE FILE.

(2) NAM FILE.

(3) AWK FILE.

MEASURED PARAMETERS:

(1) THROUGHPUT.

(2) OVERHEAD.

(3) DELAY.
(4) PACKET DELIVERY RATIO.

The performance of routing protocols can be evaluated by using the following performance metrics:

5.1.1 THROUGHPUT:

Network throughput refers to the average data rate of successful data or message delivery over a specific communications link. Network throughput is measured in bits per second (bps). Throughput is a measure of how many units of information a system can process in a given amount of time. It is applied broadly to systems ranging from various aspects of computer and network systems to organizations.

BASIC FORMULA:
Using Little's Law, one can calculate throughput with the equation:
I=R*T

Where:
I is the number of units contained within the system, inventory;
T is the time it takes for all the inventory to go through the process, flow time;
R is the rate at which the process is delivering throughput, flow rate or throughput.
If you solve for R, you will get:
R=I/T.

Run this script with command:
```
AWK -F <awk file> <trace file>
```

5.1.2 PACKET DELIVARY RATIO:

The ratio of the number of delivered data packet to the destination. This illustrates the level of delivered data to the destination.
Pdr=Number of packet receive / Number of packet send

The greater value of packet delivery ratio means the better performance of the protocol.

Run this script with command:

AWK -F <awk file> <trace file>

5.1.3 END-TO-END DELAY:

The average time taken by a data packet to arrive in the destination. It also includes the delay caused by route discovery process and the queue in data packet transmission. Only the data packets that successfully delivered to destinations that counted.

delay[I]= End time[I] - start time[I];

Run this script with command:

AWK -F <awk file> <trace file>

5.1.4 ROUTING OVERHEAD:

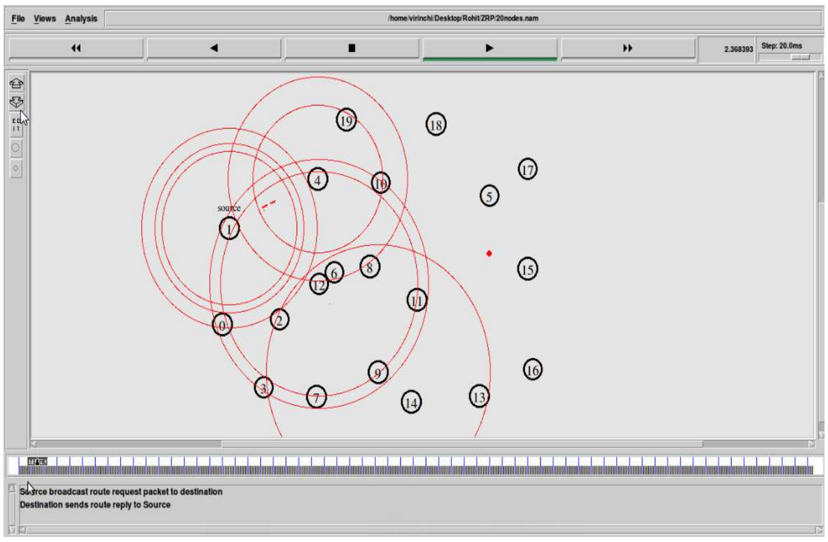
The ratio of the total packet size of control packets to the total packet size of data packets delivered to the destinations.

NOL = (Total Number of Routing packets)/ (Number of Successfully Delivered Packet)

Run this script with command:

AWK -F <awk file> <trace file>

5.2 SIMULATION OF 20 NODES USING AODV:



Source broadcast route request packet to destination
Destination sends route reply to Source

FIG 5.1

The below are the results for various network performance parameters using AODV protocol. Here we considered a network consisting of 20 nodes; we started analyzing the performance of the network by sending data from multiple sources to destinations. And we noted down corresponding network parameters like THROUGHPUT, DELAY, OVERHEAD, and PACKET DELIVARY RATIO.

S.NO	PATH (source-destination)	THROUGHPUT (Kbps)	DELAY (ms)	OVERHEAD	PACKET DELIVARY RATIO
1	1 – 5	31.639	158.616	3.124	81.03
2	2 – 7	171.146	109.392	0.024	98.83
3	3 – 8	84.2852	232.024	0.050	97.65
4	0 – 15	27.52	114.20	4.572	82.17
5	4 – 13	42.46	116.01	1.473	88.05
6	9 – 17	41.6178	508.557	0.110	95.37
7	6 – 14	84.25	252.238	0.050	97.65
8	11 – 18	84.10	230.64	0.050	97.65
9	14 – 19	35.03	200.759	2.446	80.65
10	10 – 16	55.62	387.325	0.079	96.48

5.2.1 SIMULATION RESULTS FOR AODV:

TABLE-2

We observed that data has been sent over different ROUTING PATHS. As we learnt before in the introduction of NS-2 simulator that every node will be having its own routing table, so at the starting time of the simulation, the source will broadcast the route request packets (RREQ) over the network. The route request packets contain the destination address. All the nodes will send back the route reply packets (RREP). The source will accordingly find the path to the destination through the intermediate nodes, depending on the capacity of the individual node, the packets will be received which effects the throughput. Depending on the path in different cases the delay is varied as in the TABLE-2.

From the TABLE-2;

When the THROUGHPUT is HIGH, the PDR is HIGH. Also OVERHEAD is decreased, with increase in THROUGHPUT. In the case -2, THROUGHPUT is HIGH compared to the rest.

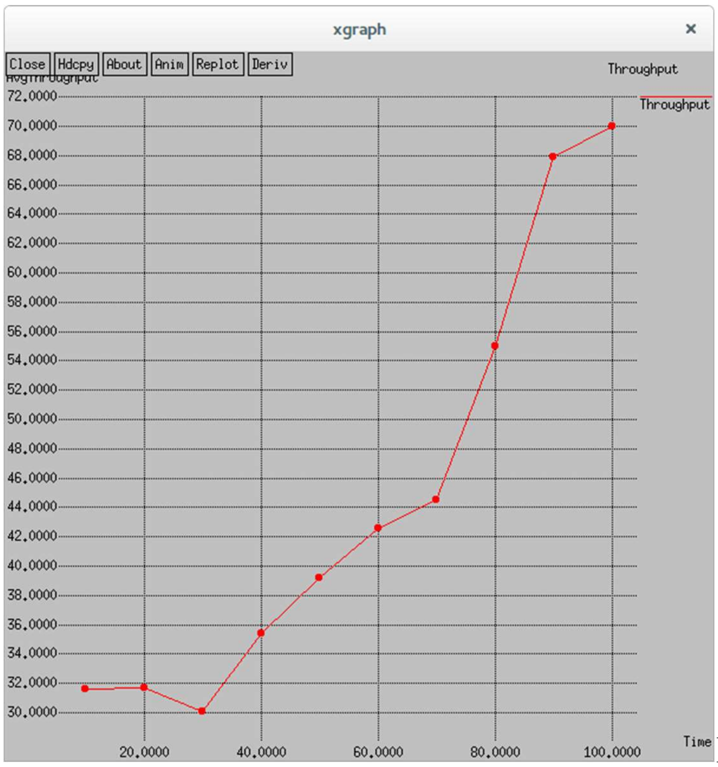
We can conclude that, this is the case where the performance of the network is HIGH, with HIGH THROUGHPUT, LOW DELAY, LOW OVERHEAD and MAXIMUM PACKET DELIVERY RATIO.

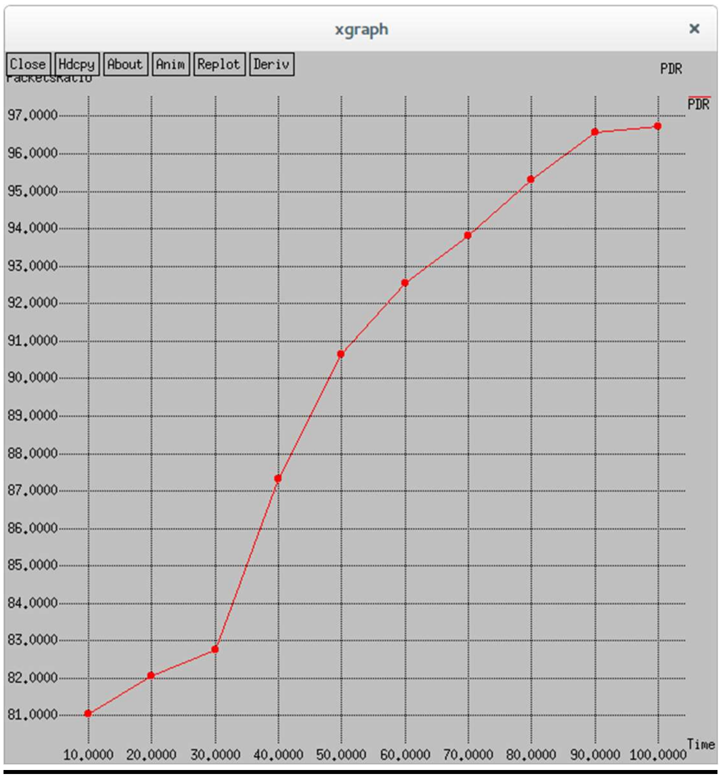
5.2.2 GRAPHICAL REPRESENTATION

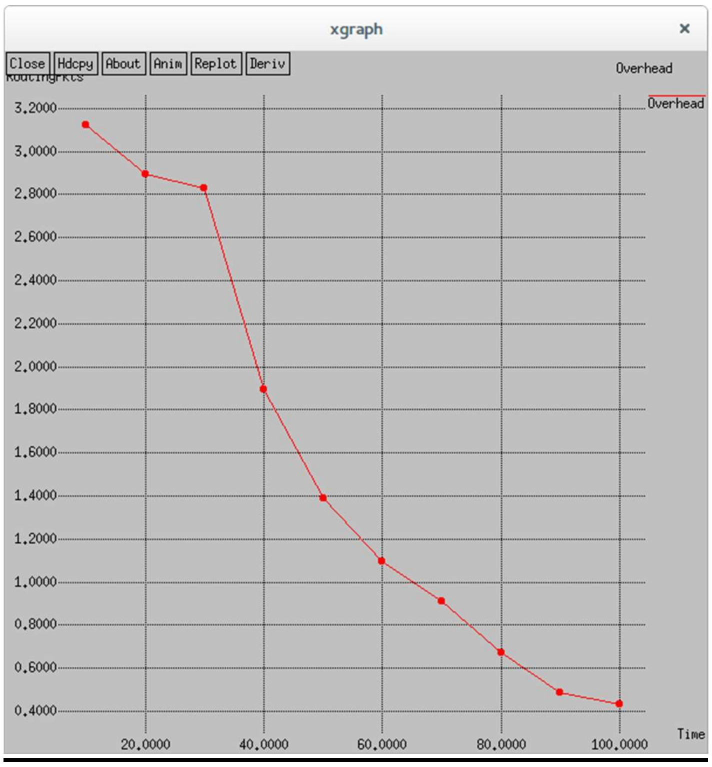
With the obtained parameter values tabulated above, we got the graphs, using the following commands:

```
Exec graph - P - t “THROUGHPUT” - x Time -y Throughput.tr  -brb  -geometry 800*400 &
Exec graph - P - t “OVERHEAD” - x Time -y overhead.tr  -brb  -geometry 800*400 &
Exec graph - P - t “DELAY” - x Time -y delay.tr -brb -geometry 800*400 &
Exec graph - P - t “Packet Delivery Ratio” - x Time -y pdr.tr -brb  -geometry 800*400&
```

The below graphs are obtained by taking Parameters Vs Time.

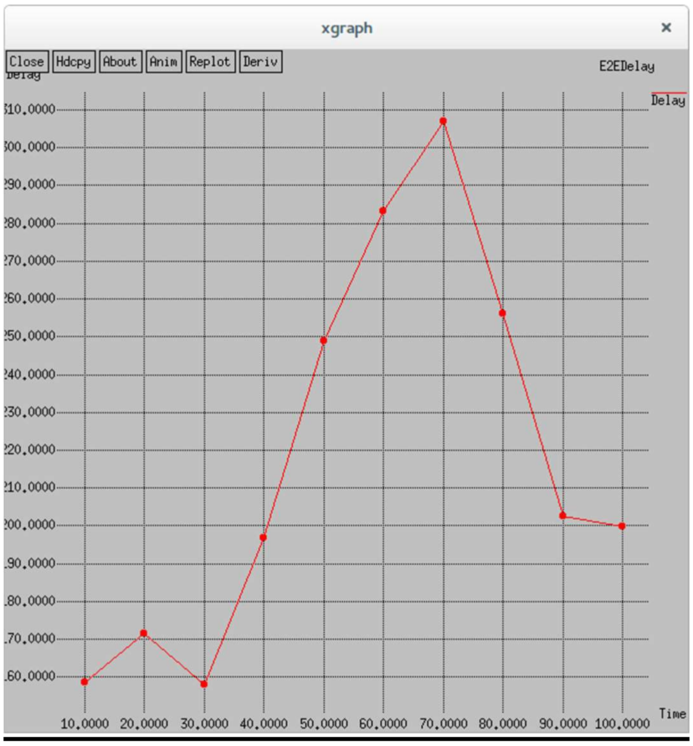

xgraph
Close
Hdcpy
About
Anim
Replot
Deriv
Throughput
AvgThroughput
Throughput
72.0000
70.0000
68.0000
66.0000
64.0000
62.0000
60.0000
58.0000
56.0000
54.0000
52.0000
50.0000
48.0000
46.0000
44.0000
42.0000
40.0000
38.0000
36.0000
34.0000
32.0000
30.0000
20.0000
40.0000
60.0000
80.0000
100.0000
Time
FIG5.2


xgraph
Close
Hdcpy
About
Anim
Replot
Deriv
PDR
PacketsRatio
PDR
97.0000
96.0000
95.0000
94.0000
93.0000
92.0000
91.0000
90.0000
89.0000
88.0000
87.0000
86.0000
85.0000
84.0000
83.0000
82.0000
81.0000
10.0000
20.0000
30.0000
40.0000
50.0000
60.0000
70.0000
80.0000
90.0000
100.0000
Time
FIG 5.3



xgraph
Close
Hdcpy
About
Anim
Replot
Deriv
RoutingPkts
Overhead
3.2000
3.0000
2.8000
2.6000
2.4000
2.2000
2.0000
1.8000
1.6000
1.4000
1.2000
1.0000
0.8000
0.6000
0.4000
20.0000
40.0000
60.0000
80.0000
100.0000
Time

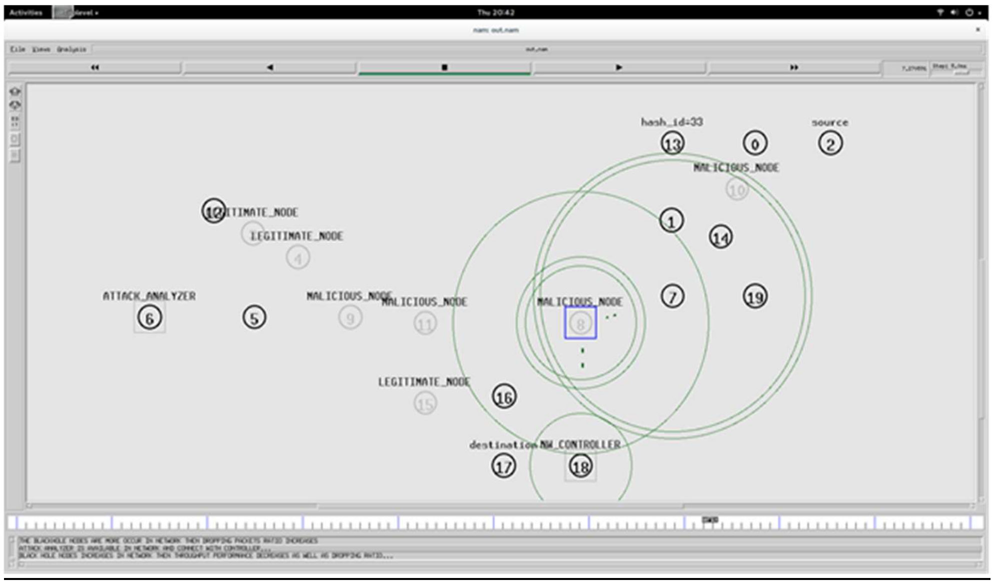
FIG 5.4



xgraph
Close
Hdcpy
About
Anim
Replot
Deriv
Delay
E2EDelay
310.0000
300.0000
290.0000
280.0000
270.0000
260.0000
250.0000
240.0000
230.0000
220.0000
210.0000
200.0000
190.0000
180.0000
170.0000
160.0000
10.0000
20.0000
30.0000
40.0000
50.0000
60.0000
70.0000
80.0000
90.0000
100.0000
Time

FIG 5.5

5.3 SIMULATION OF 20 NODES USING AODV IMPLEMENTING BLACK HOLE ATTACK:



```
THE BLACKHOLE NODES ARE MORE OCCUR IN NETWORK THEN DROPPING PACKETS RATIO INCREASES
ATTACK ANALYZER IS AVAILABLE IN NETWORK AND CONNECT WITH CONTROLLER...
BLACK HOLE NODES INCREASES IN NETWORK THEN THROUGHPUT PERFORMANCE DECREASES AS WELL AS DROPPING RATIO...
```

FIG 5.6

5.3.1

S.NO	PATH (source-destination)	THROUGHPUT (Kbps)	DELAY (ms)	OVERHEAD	PACKET DELIVARY RATIO
1	2 - 17	17.2490	-nanmsec	2.442	99.224
2	1 – 17	17.1138	-nanmsec	2.066	93.821
3	3 – 17	17.1253	-nanmsec	2.255	94.394
4	9 – 17	17.1564	-nanmsec	2.344	96.664
5	14 – 17	17.2132	-nanmsec	2.398	98.524

SIMULATION RESULTS FOR AODV WITH BLACK HOLE ATTACK:

In the previous table, we had analyzed the performance of the network using AODV protocol with multiple sources to destinations, we have observed the case where there is HIGH network performance. Then we started thinking what if any node turns to be MALICIOUS (Implementing the black hole attack) effecting the network performance. So, similarly as in previous case we started analyzing effect of the parameters using multiple sources to destinations, on the network.

From the TABLE-3,

We observed, in case-1, there is HIGH THROUGHPUT, LOW DELAY, HIGH OVERHEAD and MAXIMUM PACKET DELIVERY RATIO.

5.3.2 GRAPHICAL REPRESENTATION

With the obtained parameter values tabulated above, we got the graphs, using the following commands:

Exec graph - P - t "THROUGHPUT" - x Time -y Throughput.tr -brb -geometry 800*400 &

Exec graph - P - t "OVERHEAD" - x Time -y overhead.tr -brb -geometry 800*400 &

Exec graph - P - t "DELAY" - x Time -y delay.tr -brb -geometry 800*400 &

Exec graph - P - t "Packet Delivery Ratio" - x Time -y pdr.tr -brb -geometry 800*400&

The below graphs are obtained by taking Parameters Vs Time.

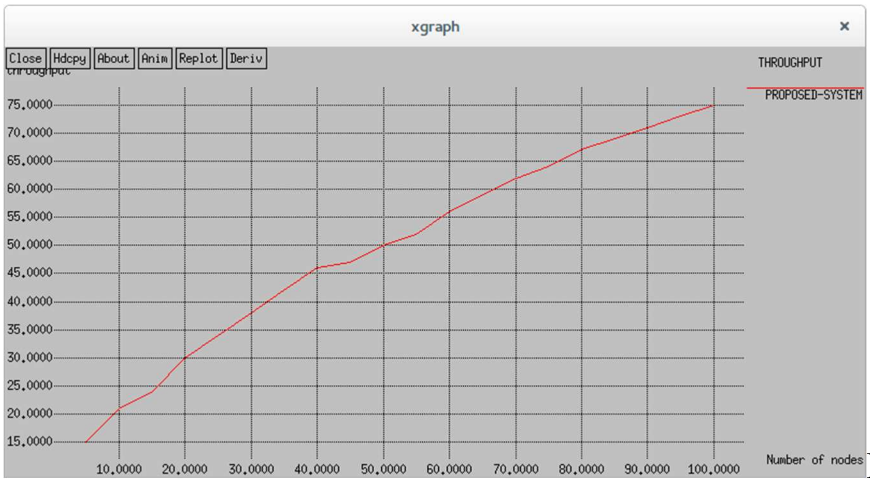


FIG 5.7

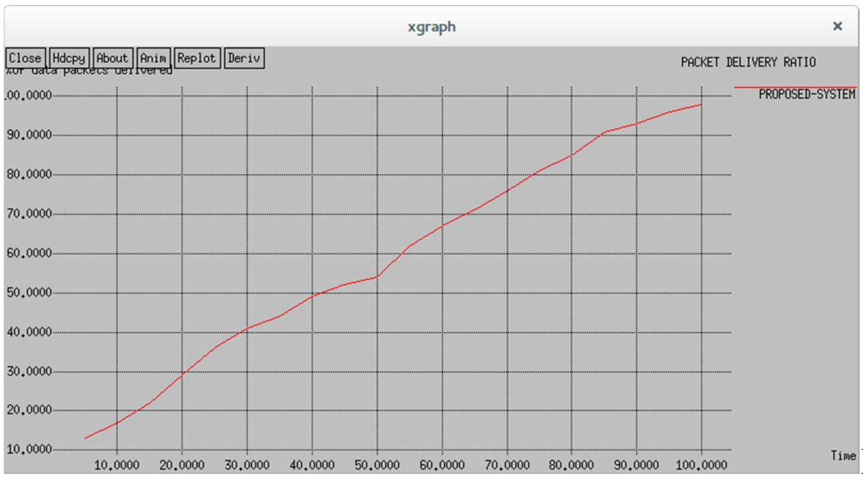


FIG 5.8

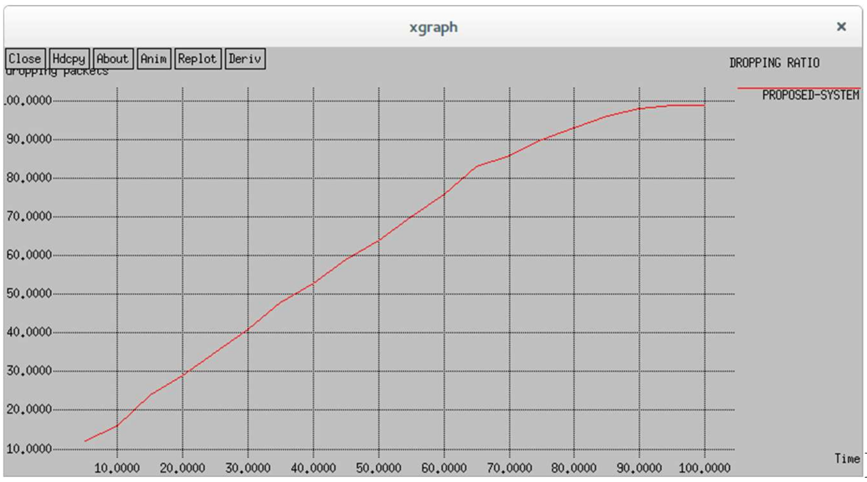


FIG 5.9

5.4 COMPARISION OF VARIOUS PERFORMANCE METRICS USING AD-HOC ON DEMAND DISTANCE VECTOR ROUTING PROTOCOL WITH AND WITHOUT BLACK HOLE ATTACK IN MOBILE AD-HOC NETWORKS:

NETWORK PARAMETERS	AODV WITHOUT BLACK HOLE ATTACK	AODV WITH BLACK HOLE ATTACK
Throughput (Kbps)	171.146	17.249
Delay (ms)	109.390	-nanmsec
Overhead	0.024	2.442
Packet delivery ratio (%)	98.83	99.224

TABLE-4

The above table, shows the comparison between network parameters using AODV protocol and in the presence of black hole attack. We observed that THROUGHPUT is LOW in the presence of black hole attack. OVERHEAD is HIGH in the presence of black hole attack, as the malicious node tries to divert the routing of packets to destination.

Chapter 6

CONCLUSION

This project evaluated and we presented the parameters that improves the **NETWORK PERFORMANCE** of mobile ad-hoc networks (MANET) using AODV protocol.
We also studied the effect of parameters in case of **BLACK HOLE ATTACK** in MOBILE ADHOC NETWORKS (MANET'S).
We compared the evaluated simulation results in **MOBILE AD-HOC NETWORKS (MANET)** using AODV protocol and performance metrics of BLACK HOLE ATTACK in MOBILE ADHOC NETWORKS (MANET'S).
Thus, analyzed various performance metrics using **AD-HOC ON DEMAND DISTANCE VECTOR (AODV)** routing protocol, with and without BLACK HOLE ATTACK in MOBILE ADHOC NETWORKS (MANET'S).

FUTURE SCOPE

The **RECEIVE REPLY METHOD (RREP) ALGORITHM** can be used to find the secure routes and **PREVENT THE BLACK HOLE NODES** in the MANET by checking whether there is large difference between the sequence number of source node or intermediate node who has sent back RREP or not.
To **AVOID MULTIPLE BLACK HOLES** acting in the group, there is a **PCBHA technique** to identify multiple black holes cooperating with each other and a solution to discover a safe route avoiding cooperative black hole attack.
As future work, research work intend to develop simulations to analyze the performance of the proposed solution based on the various security parameters like mean delay time, packet overhead, memory usage, mobility, increasing number of malicious node, increasing number of nodes and scope of the black hole nodes and also focusing on resolving the problem of multiple attacks against AODV.

REFERENCES:

1. Harman deep Singh and Manpreeet Singh, “Effect of Black Hole Attack on AODV, OLSR and ZRP Protocol in MANETs”, International Journal of Advanced Trends in Computer Science and Engineering Volume 2, No.3, May - June 2013.
2. IrshadUllah and Shahzad Anwar, “Effects of Black Hole Attack on MANET Using Reactive and Proactive Protocols” IJCSI International Journal of Computer Science Issues, Vol. 10, Issue 3, No 1,-May 2013.
3. Jane Y. Yu and Peter H. J. Chong, "A Survey of Clustering Schemes for Mobile Ad Hoc Networks," IEEE Communication Surveys, Volume 7, No. 1, pp. 32-48, First Quarter 2005.
4. R. Bruno, M. Conti, and E. Gregori, "Mesh Networks: Commodity Multihop Ad Hoc Networks," IEEE Communications Magazine, pp. 123-131, March 2005.
5. Baoxian Zhang and Hussein T. Mouftah, " QoS Routing of Wireless Ad Hoc Networks: Problems, Algorithms, and Protocols," IEEE Communications Magazine, pp. 110-117, Oct. 2005.
6. Djenouri, D., Khelladi, L., and Badache A.N., "A survey of security issues in mobile ad hoc and sensor networks," IEEE Communications Surveys & Tutorials, Volume 7, Issue 4, pp. 2-28, Fourth Quarter 2005.
7. Raghvendra Prasad, KuntalBarua, ”Implementation, Detection and Prevention of Black hole Attack for Mobile Ad-hoc Network Scenario using NS-2”, International Journal of Science and Research (IJSR)
8. Tarandeep Kaur and Amarvir Singh, “Performance Evaluation of MANET with Black Hole Attack Using Routing Protocols”, International Journal of Engineering Research and Applications(IJERA) Vol. 3, Issue 4, Jul-Aug 2013,
9. T. H. Tie, C. E. Tan and S. P. Lau, "Alternate link maximum energy level ad hoc distance vector scheme for energy efficient ad hoc networks routing," International conference on computer and communication engineering (ICCCE), 2010.
10. K. Akkaya and M. Younis, "A survey on routing protocols for wireless sensor networks," Ad Hoc Networks, Volume 3, Issue 3, Pages 325-349, 2005.
11. D. B. Johnson, D. A. Maltz and Y. C. Hu, "The dynamic source routing protocol for mobile ad hoc networks (DSR)" IETF MANET Working Group, 2004.
12. L. Zhanjun, W. Rui, L. Qilie, L. Yun, C. Qianbin, and W. Ping, "An energy-constrained routing protocol for mobile ad hoc networks" International Conference on Communication Software and Networks, IEEE Computer Society, 2009.