A
Summer Internship – II Report
On

"Network Topology Implementation Using NS2"

Prepared by
Amartya Singh (17EC088)

Under the guidance of

Prof. Brijesh Kundaliya

Submitted to
Charotar University of Science & Technology
EC446 – Summer Internship II
7th Semester of B.Tech (EC)

Submitted at

CHARUSAT
CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY
Accredited Grade "A" by NAAC, Govt of India
Accredited Grade "A" by KCG, Govt of Gujarat

V. T. PATEL DEPARTMENT OF ELECTRONICS & COMMUNICATION
Faculty of Technology & Engineering
Chandubhai S. Patel Institute of Technology
Charotar University of Science & Technology
At: Changa, Dist: Anand – 388421
June-July 2020

CHARUSAT
CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY
Accredited Grade "A" by NAAC, Govt of India
Accredited Grade "A" by KCG, Govt of Gujarat

CERTIFICATE

This is to certify that the report entitled "Network Topology Implementation Using NS2" is a bonafide work carried out by **Amartya Singh** under the guidance and supervision of **Prof. Brijesh Kundaliya** for the subject Summer Internship II **(EC446)** of 7th Semester of Bachelor of Technology in Electronics & Communication at Faculty of Technology & Engineering (C.S.P.I.T.) – CHARUSAT, Gujarat.

To the best of my knowledge and belief, this work embodies the work of candidate himself, has duly been completed, and fulfills the requirements of the course.

Under the supervision of,

Prof. Brijesh Kundaliya
Assistant Professor
Department of Electronics & Communication,
C.S.P.I.T., CHARUSAT-Changa.

Dr. Trushit K. Upadhyaya
Head of Department,
Department of Electronics & Communication
C.S.P.I.T., CHARUSAT- Changa, Gujarat.

Chandubhai S Patel Institute of Technology (C.S.P.I.T.)
Faculty of Technology & Engineering, CHARUSAT
At: Changa, Ta. Petlad, Dist. Anand, Gujarat – 388421

ABSTRACT

First, we learned about computer network basics like the type of network, different network topology, requirement of a network, OSI Model, TCP/IP Model, and different networking hardware components. Then we started the work on our project.

In this project, I have used NS2 which is a network simulator to implement differently wired topology like star, mesh, bus, ring of the wired network with different protocols like TCP, FTP, UDP by using TCL which is a scripting language. Later also implemented wireless network DSDV and AODV routing protocol.

NAM is the network animator which simulates the designed network graphically using the trace file that is generated while executing the TCL script. So, by using this we can easily visualize and analyze the flow of data packets with their respective behavior also the nodes and link between them.

AWK is a programming language that is used for processing text-based data. Here we run AWK on the generated trace file after the execution of TCL script which helps to sort out the data packets according to their behavior.

ACKNOWLEDGMENT

We have put in our best efforts in this project. However, it would not have been possible without the kind support and help of teachers of our college. We would like to extend our sincere thanks to all of them.

We are highly indebted to Prof. Brijesh Kundaliya for their guidance and constant supervision as well as for providing necessary information regarding the project and also for their support in completing the project.

Our thanks and appreciation also go to our colleagues in developing the project and people who have willingly helped us out with their abilities.

TABLE OF CONTENTS

ABSTRACT…… I

ACKNOWLEDGEMENT…… II

TABLE OF CONTENTS……..... III

LIST OF FIGURES……..... V

ABBREVIATIONS…… VII

1: Introduction to Computer Networking…….... 1

1.1: What is Computer Networking?..... 1

1.2: Requirement of Computer Network…… 1

1.3: Types of Computer Network…….. 1

1.3.1: LAN…… 2

1.3.2: MAN…….. 2

1.3.3: WAN……... 3

1.3.4: PAN…… 3

1.4: Hardware Components for Networking…….. 4

1.4.1: Switch……. 4

1.4.2: Repeater……. 4

1.4.3: Hub……. 5

1.4.4: Router……. 6

1.4.5: Bridges..……. 6

1.4.6: Gateways……. 7

1.4.7: Network Cables..……. 7

1.4.8: Network Interface Cards…...……. 8

2: OSI Model…… 9

2.1: What is OSI Model?..... 9

2.2: Physical Layer Functions……..... 10

2.3: Data Link Layer Functions……..... 11

2.4: Network Layer Functions……..... 12

2.5: Transport Layer Functions……..... 12

2.6: Session Layer Functions……..... 13

2.7: Presentation Layer Functions…..... 14

2.8: Application Layer Functions…..... 15

4: Topology Implementation in NS2……..……..... 18
4.1: Wired Topology…….....18
4.1.1: Bus Topology…… 18
4.1.2: Ring Topology……..…… 19
4.1.3: Star Topology…… 20
4.1.4: Mesh Topology……..…… 22
4.1.5: Tree Topology…… 23
4.2: Wireless Networks…….....24
4.2.1: Simulation using AODV routing protocol…… 24
4.2.2: Simulation using DSDV routing protocol.…… 25
4.3: Introduction to NS2..…….....26
4.3.1: Installation of NS2 in Linux…….……. 26
4.3.2: TCL Scripting…… 27
4.3.3: Starting NS2……... 28

References…….…….……..... 29
TCL Scripts for topology…….…….…….……..... 30

LIST OF FIGURES

Figure Number	Figure Name	Page Number
1	Types Of Computer Network	1
2	LAN	2
3	MAN	2
4	WAN	3
5	PAN	3
6	Switch	4
7	Repeater	5
8	Hub	5
9	Router	6
10	Bridges	6
11	Gateways	7
12	Network Cables	7
13	Network Interface Card	8
14	OSI Model	9
15	OSI Model	10
16	Physical Layer	10
17	Data Link Layer	11
18	Network Layer	12
19	Transport Layer	13
20	Session Layer	14
21	Presentation Layer	14
22	Application Layer	15
23	TCP/IP Model	16
24	Types Of Topology	18
25	Bus Topology	19
26	Bus topology output	19
27	Bus topology data flow	19
28	Ring Topology	20
29	Ring topology output	20
30	Ring topology data flow	20
31	Star Topology	21
32	Star topology output	21

Figure Number	Figure Name	Page Number
33	Star Topology data flow	21
34	Mesh Topology	22
35	Mesh Topology Output	22
36	Mesh Topology data flow	22
37	Tree Topology	23
38	Tree Topology Output	23
39	Tree Topology data flow	24
40	AODV Initial Position	24
41	AODV data flow	24
42	DSDV Initial Position	25
43	DSDV data flow	25

ABBREVIATIONS

OSI	Open Systems Interconnection
TCP	Transmission Control Protocol
IP	Internet Protocol
NS2	Network Simulator Version 2
TCL	Tickle
AWK	Aho, Weinberger and Kernighan
NAM	Network Animator
LAN	Local Area Network
ISO	International Standardization Organization
CRC	Cyclic Redundancy Check
DSDV	Destination-Sequenced Distance-Vector
AODV	Ad hoc On-Demand Distance Vector
TCP	Transmission Control Protocol
FDP	Foundry Discovery Protocol
UDP	User Datagram Protocol
MAN	Metropolitan Area Network
WAN	Wide Area Network
PAN	Personal Area Network
FTAM	File Transfer Access & Management
CBR	Constant Bitrate

1: Introduction to Computer Networking

1.1 What is Computer Networking?

Computer Network is a group of computers connected with each other through wires, optical fibres or optical links so that various devices can interact with each other through a network.

The aim of the computer network is the sharing of resources among various devices.

In the case of computer network technology, there are several types of networks that vary from simple to complex level.

1.2 Requirement of Computer Network

- For better communication speed
- To provide file sharing feature
- For taking backups in other systems
- To restore from backups which is stored in other system
- For Hardware & Software sharing
- To provide better security
- For Scalability
- For Reliability

1.3 Types of Computer Network

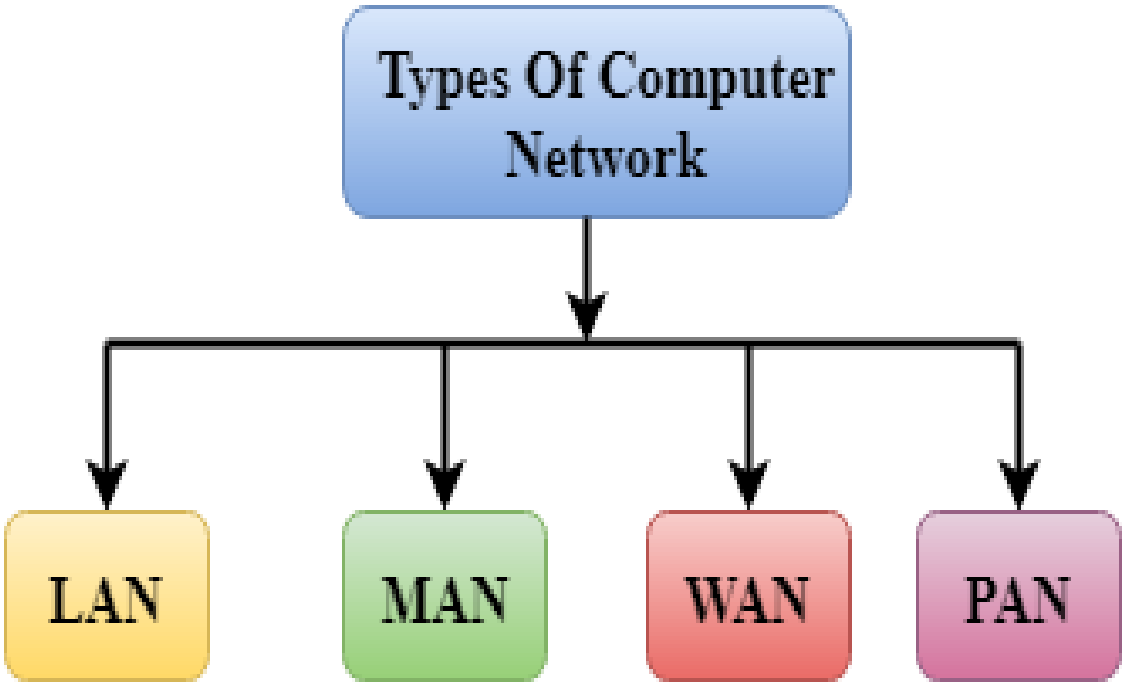

Figure 1: Types of Computer Network

1.3.1 LAN (Local Area Network)

- Local Area Network is a group of computers connected to each other in a small area such as building, office.
- LAN is used for connecting two or more personal computers through a communication medium such as twisted pair, coaxial cable, etc.
- It is less costly as it is built with inexpensive hardware such as hubs, network adapters, and ethernet cables.
- The data is transferred at an extremely faster rate in Local Area Network.
- Local Area Network provides higher security.

Figure 2: LAN

1.3.2 MAN Network (Metropolitan Area Network)

- A metropolitan area network is a network that covers a larger geographic area by interconnecting a different LAN to form a larger network.
- In MAN, various LANs are connected to each other through a telephone exchange line.
- It has a higher range than Local Area Network (LAN).

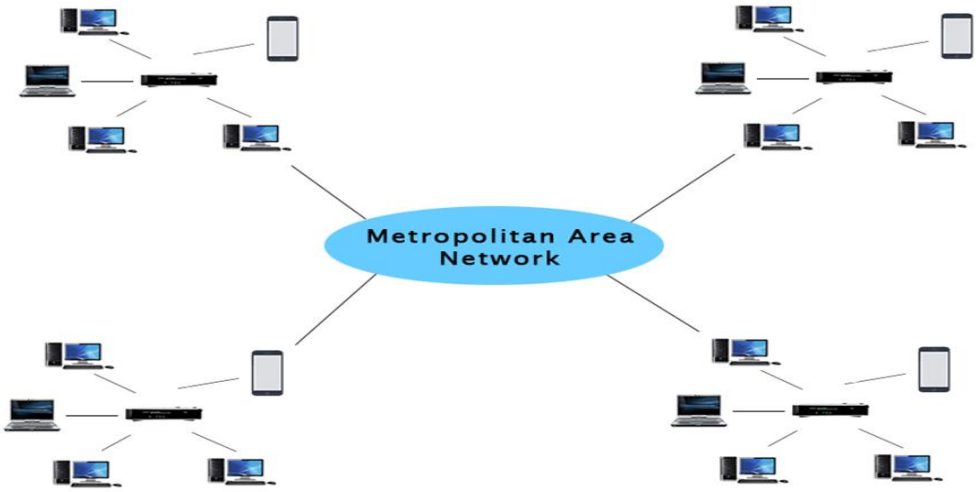

Figure 3: MAN

1.3.3 WAN Network (Wide Area Network)

- A Wide Area Network is a network that extends over a large geographical area such as states or countries.
- A Wide Area Network is quite bigger network than the LAN.
- A Wide Area Network is not limited to a single location, but it spans over a large geographical area through a telephone line, fiber optic cable or satellite links.
- The internet is one of the biggest WAN in the world.

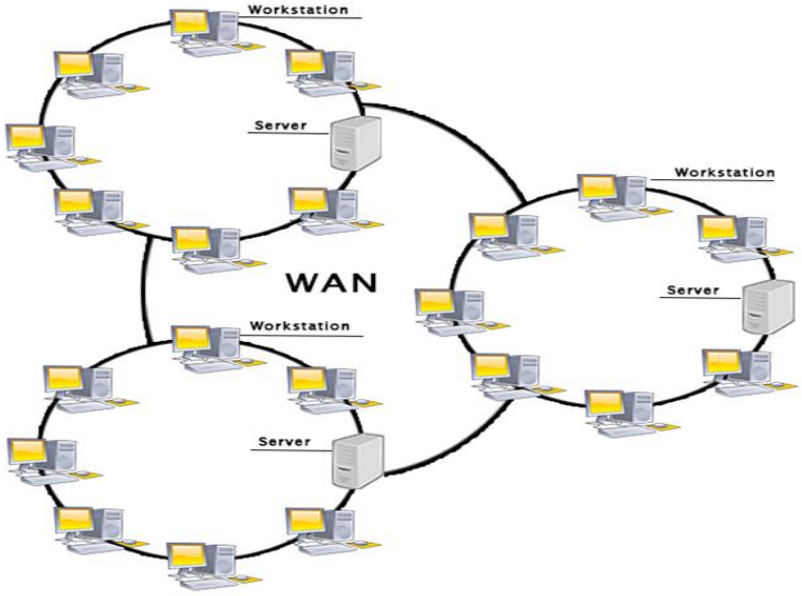

Figure 4: WAN

1.3.4 PAN Network (Personal Area Network)

- Personal Area Network is a network arranged within an individual person.
- Personal Area Network is used for connecting the computer devices of personal use is known as Personal Area Network.
- Personal computer devices that are used to develop the personal area network are the laptop, mobile phones, media player and play stations

Figure 5: PAN

1.4 Hardware Components for Networking

1.4.1 Switch

- A switch is a hardware device that connects multiple devices on a computer network.
- A Switch contains more advanced features than Hub.
- The Switch contains the updated table that decides where the data is transmitted or not.
- Switch delivers the message to the correct destination based on the physical address present in the incoming message. A Switch does not broadcast the message to the entire network like the Hub. It determines the device to whom the message is to be transmitted.
- Therefore, we can say that switch provides a direct connection between the source and destination. It increases the speed of the network.

Figure 6: Switch

1.4.2 Repeater

Repeaters are network devices operating at physical layer of the OSI model that amplify or regenerate an incoming signal before retransmitting it. They are incorporated in networks to expand its coverage area. They are also known as signal boosters.

When an electrical signal is transmitted via a channel, it gets attenuated depending upon the nature of the channel or the technology. This poses a limitation upon the length of the LAN or coverage area of cellular networks. This problem is alleviated by installing repeaters at certain intervals.

Repeaters amplifies the attenuated signal and then retransmits it. Digital repeaters can even reconstruct signals distorted by transmission loss. So, repeaters are popularly incorporated to connect between two LANs thus forming a large single LAN.

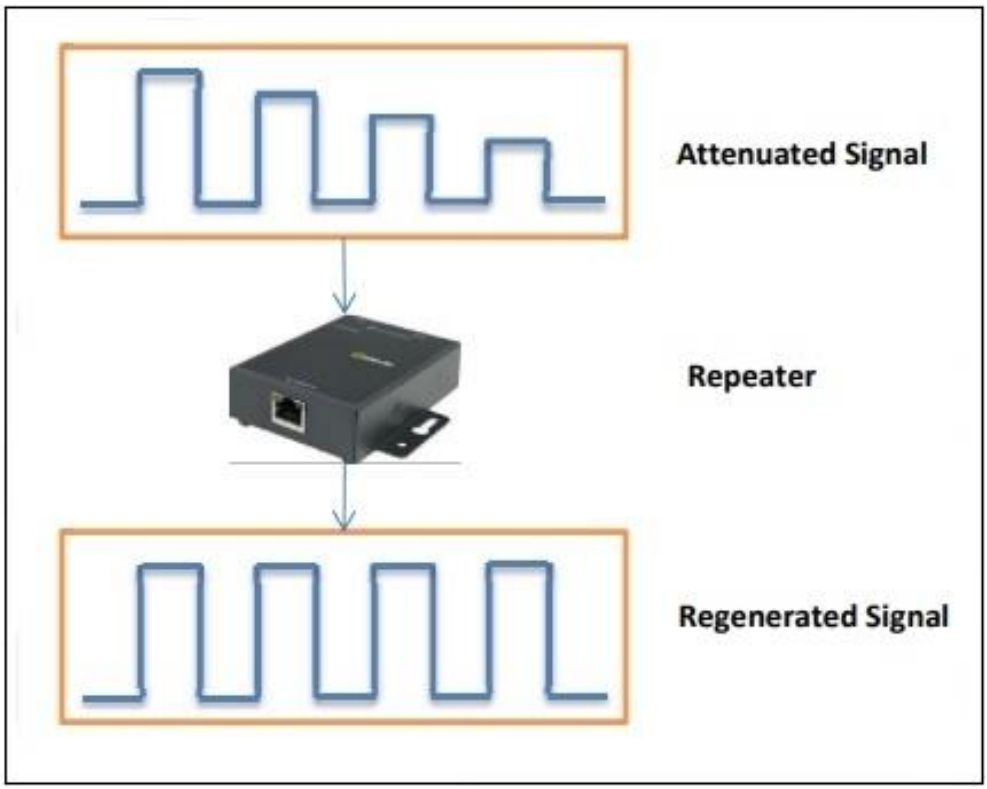

Figure 7: Repeater

1.4.3 Hub

- A Hub is a hardware device that divides the network connection among multiple devices.
- When computer requests for some information from a network, it first sends the request to the Hub through cable. Hub will broadcast this request to the entire network. All the devices will check whether the request belongs to them or not. If not, the request will be dropped.
- The process used by the Hub consumes more bandwidth and limits the amount of communication. Nowadays, the use of hub is obsolete, and it is replaced by more advanced computer network components such as Switches, Routers.

Figure 8: Hub

1.4.4 Router

- A router is a hardware device which is used to connect a LAN with an internet connection. It is used to receive, analyze and forward the incoming packets to another network.
- A router works in a Layer 3 (Network layer) of the OSI Reference model.
- A router forwards the packet based on the information available in the routing table.
- It determines the best path from the available paths for the transmission of the packet.

Figure 9: Router

1.4.5 Bridges

A bridge operates at data link layer. A bridge is a repeater, with add on the functionality of filtering content by reading the MAC addresses of source and destination. It is also used for interconnecting two LANs working on the same protocol. It has a single input and single output port, thus making it a 2-port device.

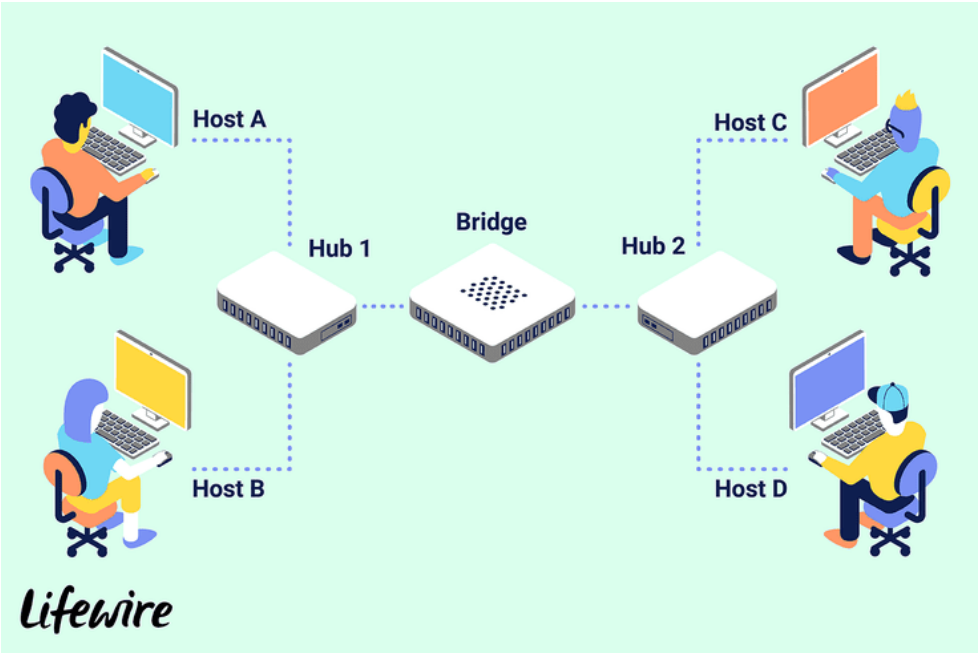

Figure 10: Bridge

1.4.6 Gateways

A gateway, as the name suggests, is a passage to connect two networks together that may work upon different networking models. They basically work as the messenger agents that take data from one system, interpret it, and transfer it to another system. Gateways are also called protocol converters and can operate at any network layer. Gateways are generally more complex than switch or router.

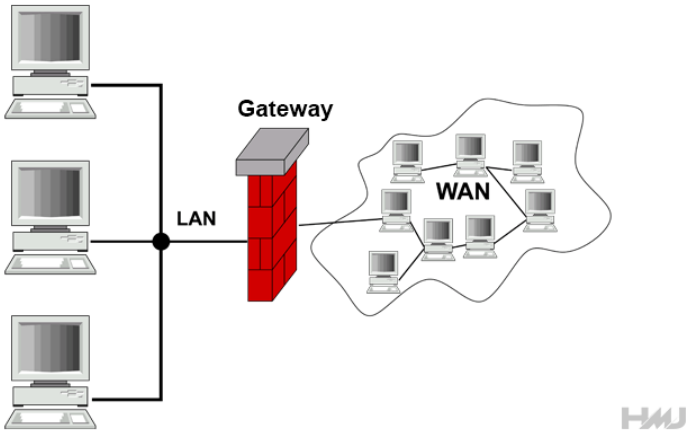

Figure 11: Gateways

1.4.7 Network Cables

Cable is a transmission media used for transmitting a signal.

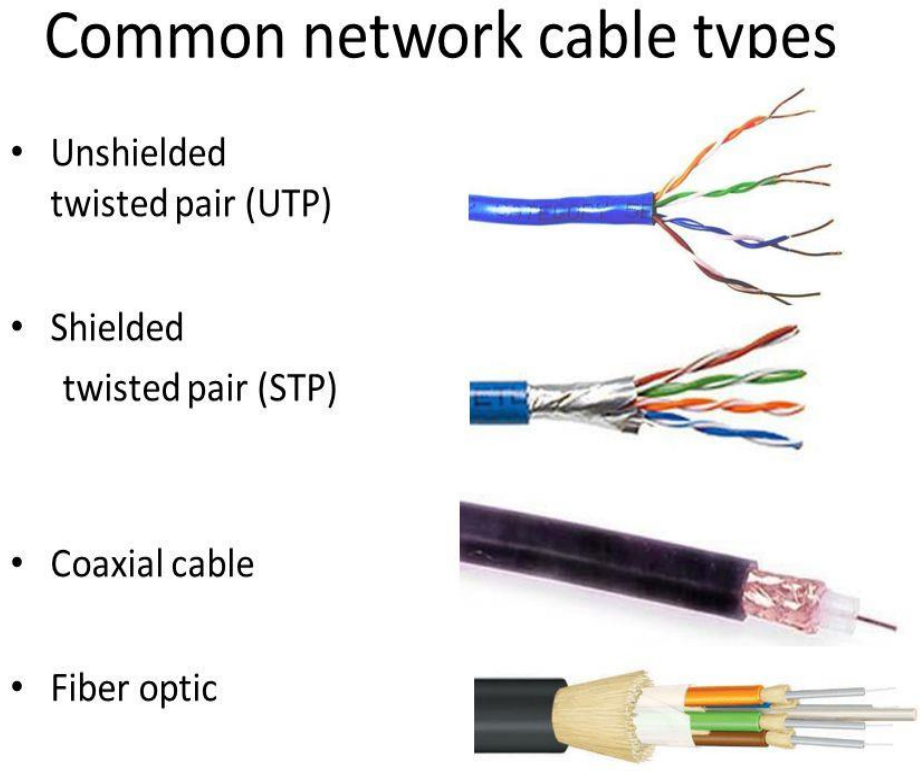

Figure 12: Network Cables

1.4.8 Network Interface Cards

- NIC stands for network interface card.
- NIC is a hardware component used to connect a computer with another computer onto a network
- The MAC address or physical address is encoded on the network card chip which is assigned by the IEEE to identify a network card uniquely. The MAC address is stored in the PROM (Programmable read-only memory)

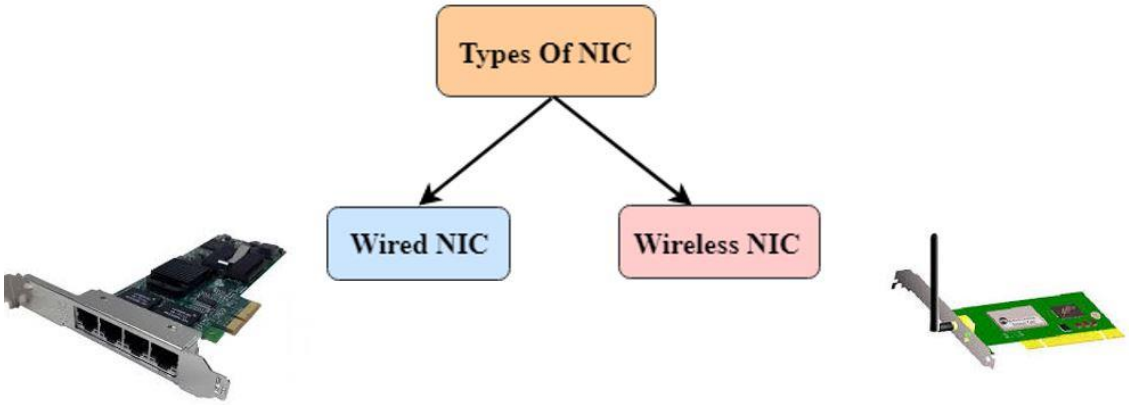

Figure 13: Network Interface Card

2: OSI Model

2.1 What is OSI Model?

- OSI stands for Open System Interconnection is a reference model that describes how information from a software application in one computer moves through a physical medium to the software application in another computer.
- OSI consists of seven layers, and each layer performs a particular network function.
- OSI model was developed by the International Organization for Standardization (ISO) in 1984, and it is now considered as an architectural model for the inter-computer communications.
- OSI model divides the whole task into seven smaller and manageable tasks. Each layer is assigned a particular task.
- Each layer is self-contained, so that task assigned to each layer can be performed independently.

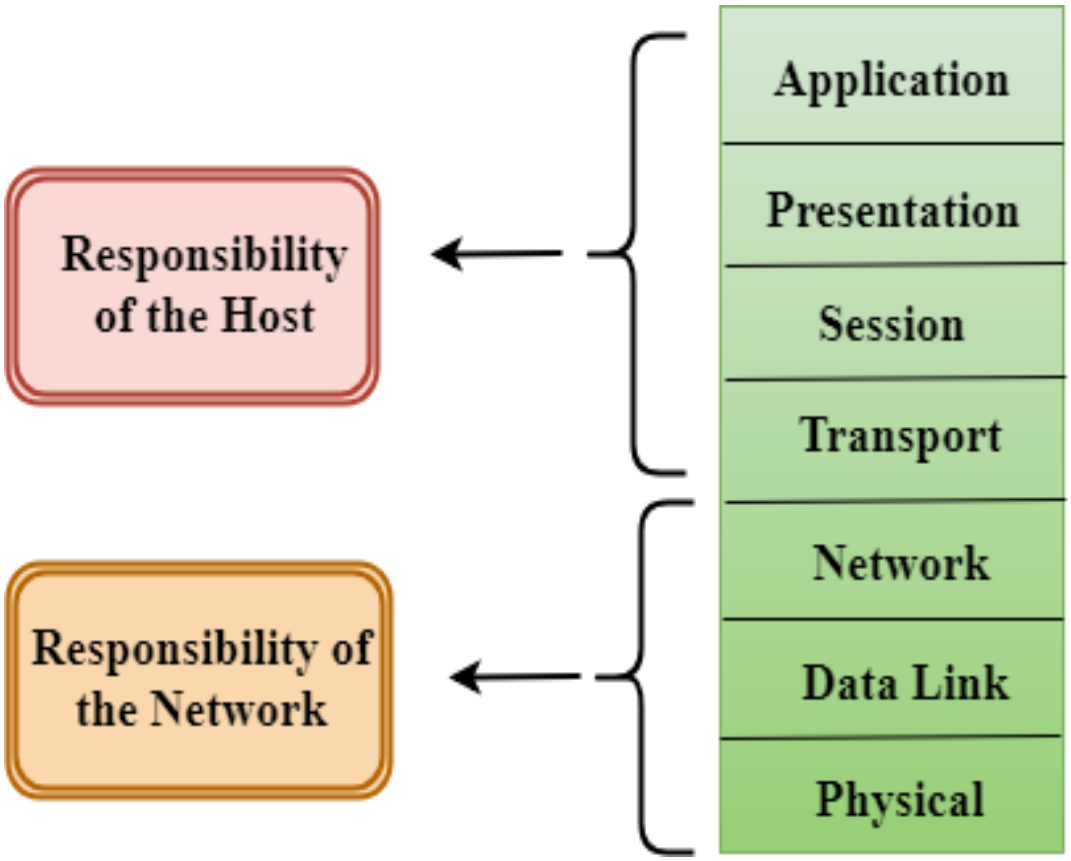

Figure 14: OSI Model

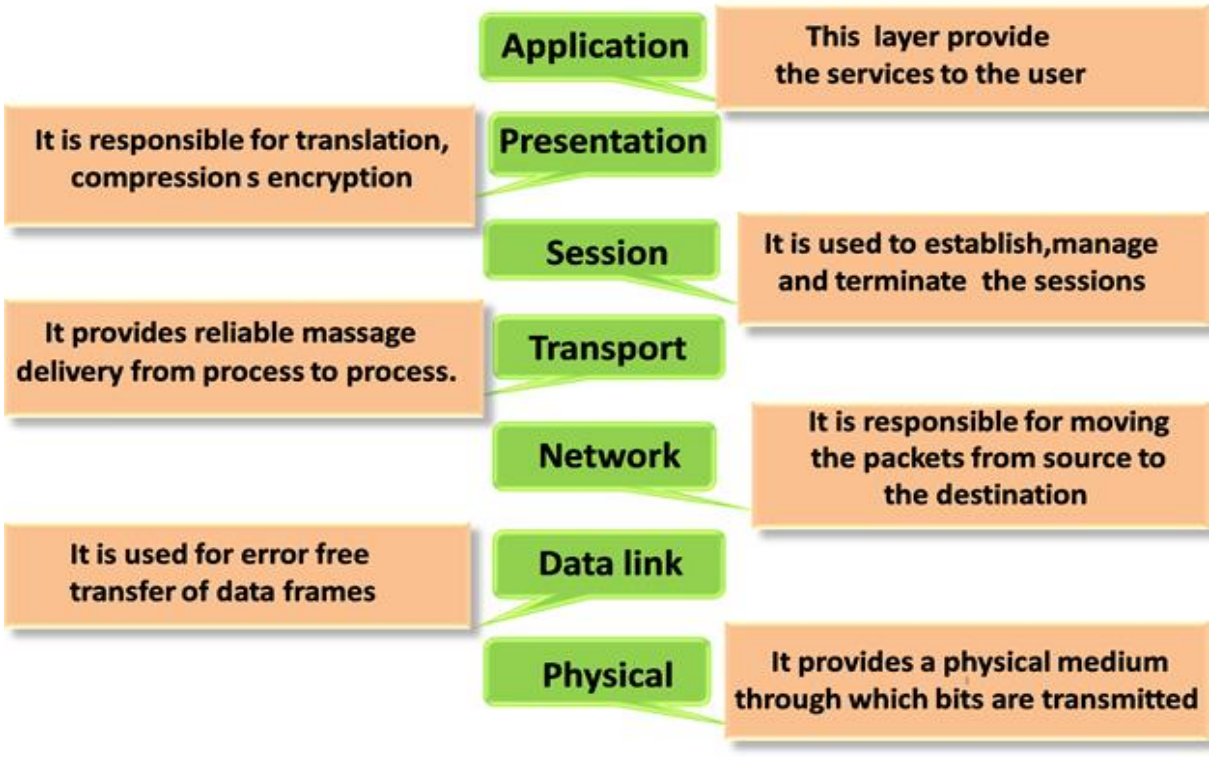

Figure 15: OSI Model

2.2 Physical Layer Functions

- **Line Configuration:** It defines the way how two or more devices can be connected physically.
- **Data Transmission:** It defines the transmission mode whether it is simplex, half-duplex or full-duplex mode between the two devices on the network.
- **Topology:** It defines the way how network devices are arranged.
- **Signals:** It determines the type of the signal used for transmitting the information.

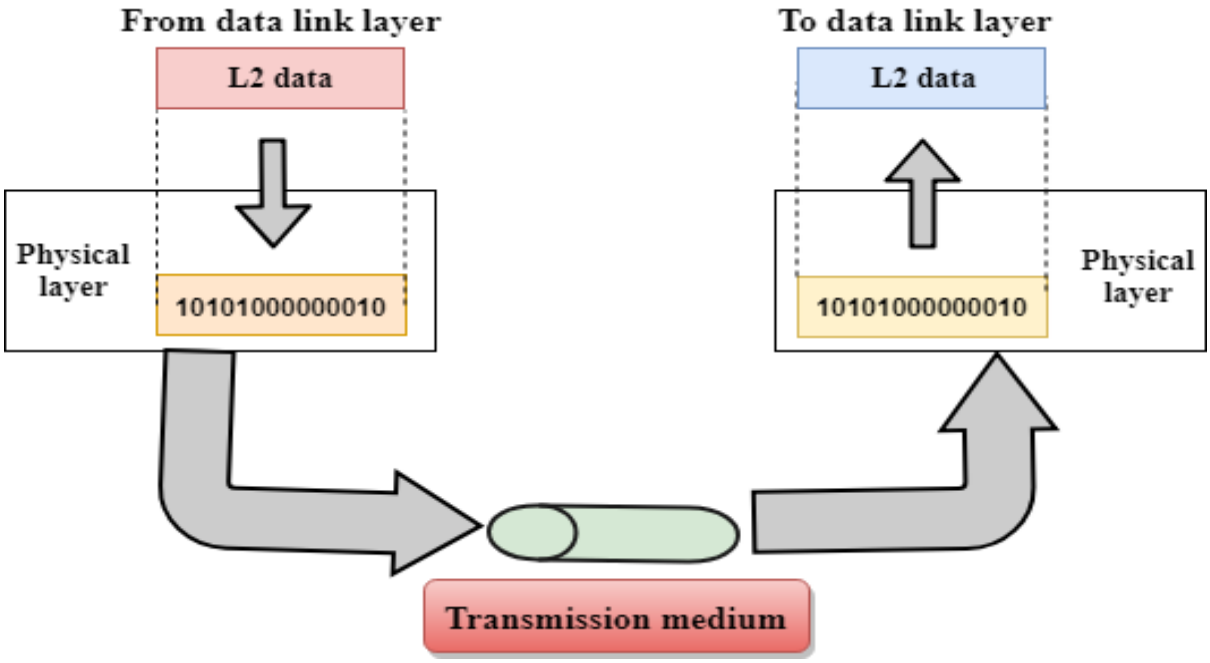

Figure 16: Physical Layer

2.3 Data Link Layer Functions

- **Framing:** The data link layer translates the physical's raw bit stream into packets known as Frames. The Data link layer adds the header and trailer to the frame. The header which is added to the frame contains the hardware destination and source address.
- **Physical Addressing:** The Data link layer adds a header to the frame that contains a destination address. The frame is transmitted to the destination address mentioned in the header.
- **Flow Control**: Flow control is the main functionality of the Data-link layer. It is the technique through which the constant data rate is maintained on both the sides so that no data get corrupted. It ensures that the transmitting station such as a server with higher processing speed does not exceed the receiving station, with lower processing speed.
- **Error Control:** Error control is achieved by adding a calculated value CRC (Cyclic Redundancy Check) that is placed to the Data link layer's trailer which is added to the message frame before it is sent to the physical layer. If any error seems to occurr, then the receiver sends the acknowledgment for the retransmission of the corrupted frames.
- **Access Control:** When two or more devices are connected to the same communication channel, then the data link layer protocols are used to determine which device has control over the link at a given time.

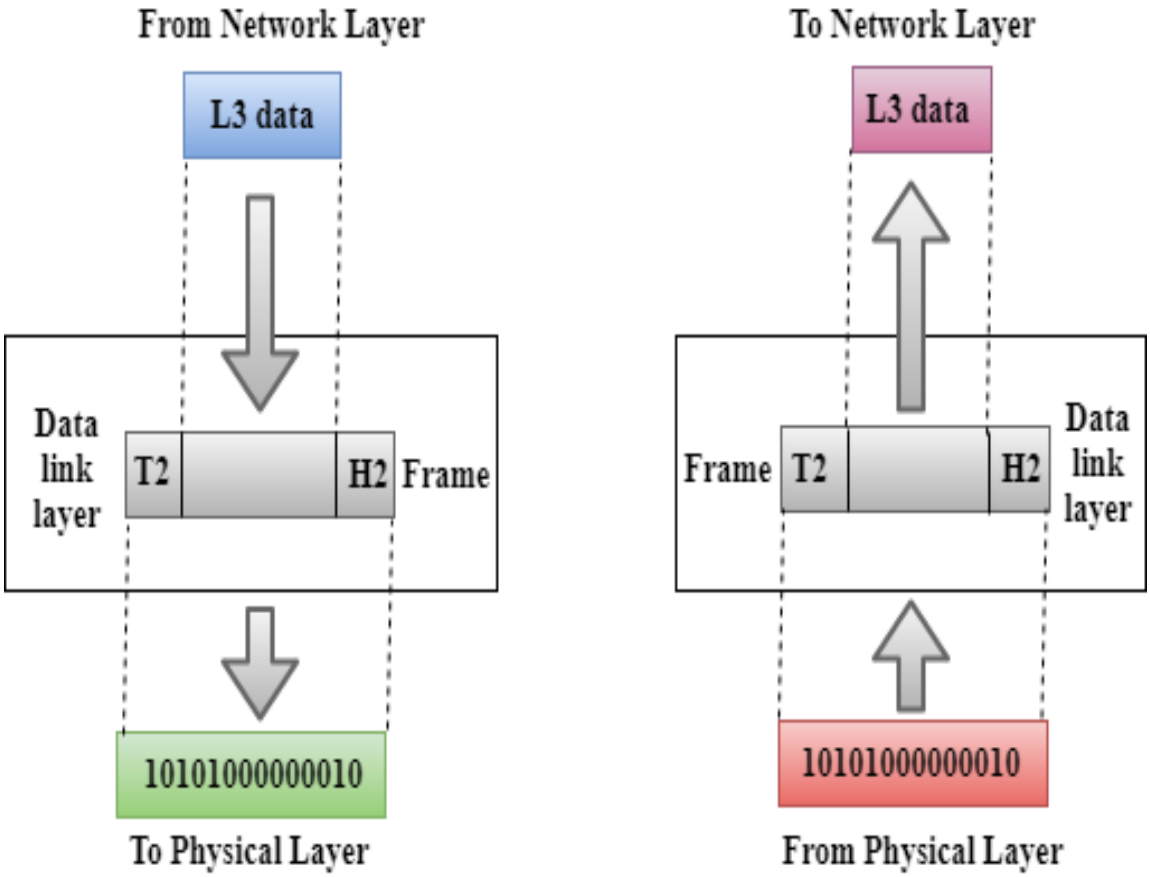

Figure 17: Data Link Layer

2.4 Network Layer Functions

- **Internetworking:** An internetworking is the main responsibility of the network layer. It provides a logical connection between different devices.
- **Addressing:** A Network layer adds the source and destination address to the header of the frame. Addressing is used to identify the device on the internet.
- **Routing:** Routing is the major component of the network layer, and it determines the best optimal path out of the multiple paths from source to the destination.
- **Packetizing:** A Network Layer receives the packets from the upper layer and converts them into packets. This process is known as Packetizing. It is achieved by internet protocol (IP).

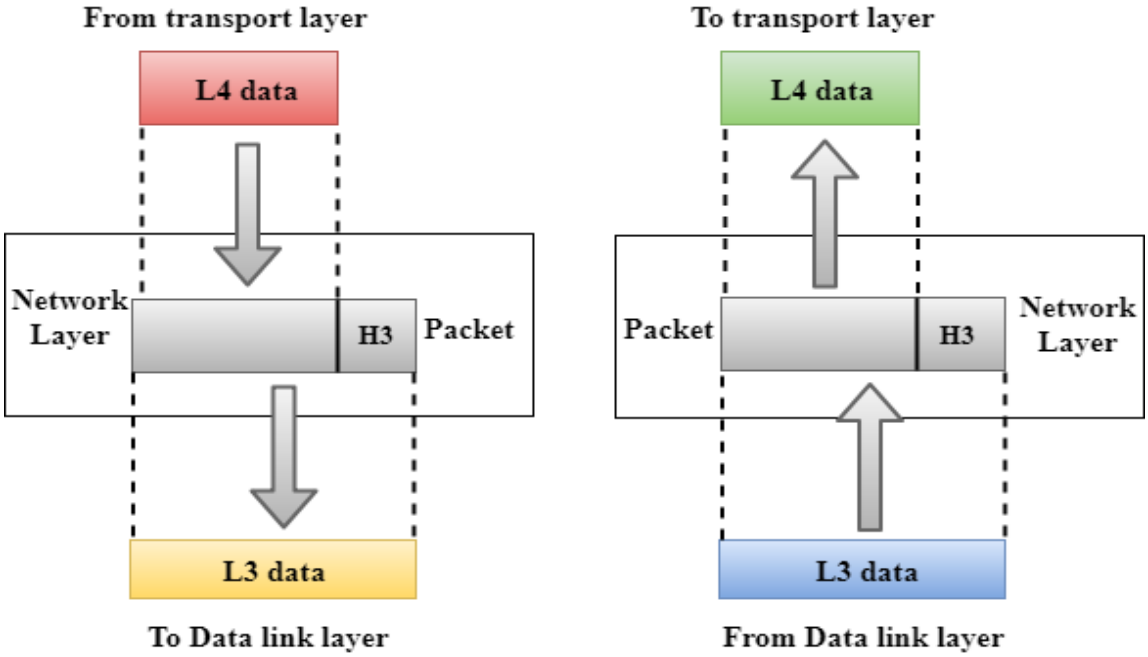

Figure 18: Network Layer

2.5 Transport Layer Functions

- **Service-point addressing:** Computers run several programs simultaneously due to this reason, the transmission of data from source to the destination not only from one computer to another computer but also from one process to another process. The transport layer adds the header that contains the address known as a service-point address or port address. The responsibility of the network layer is to transmit the data from one computer to another computer and the responsibility of the transport layer is to transmit the message to the correct process.
- **Segmentation and reassembly:** When the transport layer receives the message from the upper layer, it divides the message into multiple segments, and each segment is assigned with a sequence number that uniquely identifies each segment. When the message has arrived at the destination, then the transport layer reassembles the message based on their sequence numbers.

- **Connection control:** Transport layer provides two services Connection-oriented service and connectionless service. A connectionless service treats each segment as an individual packet, and they all travel in different routes to reach the destination. A connection-oriented service makes a connection with the transport layer at the destination machine before delivering the packets. In connection-oriented service, all the packets travel in the single route.
- **Flow control:** It is performed end-to-end rather than across a single link.
- **Error control:** It is performed end-to-end rather than across the single link. The sender transport layer ensures that message reach at the destination without any error.

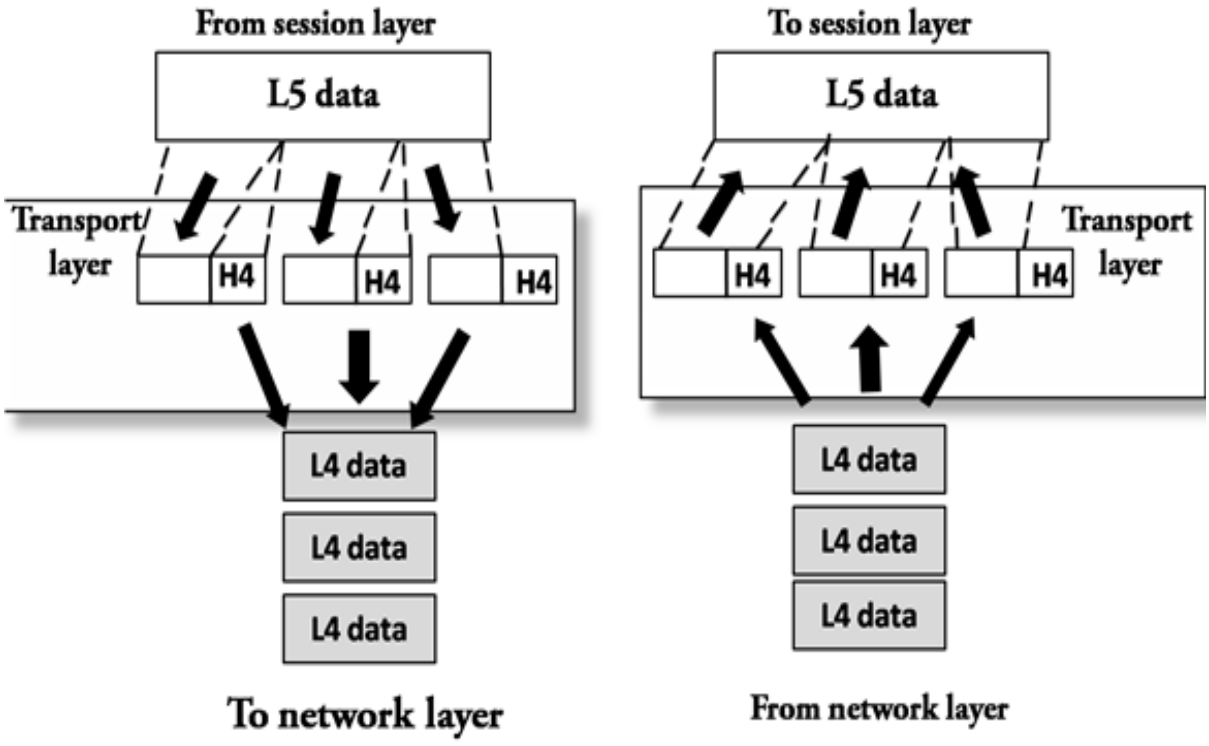

Figure 19: Transport Layer

2.6 Session Layer Functions

- **Dialog control:** Session layer acts as a dialog controller that creates a dialog between two processes or we can say that it allows the communication between two processes which can be either half-duplex or full-duplex.
- **Synchronization:** Session layer adds some checkpoints when transmitting the data in a sequence. If some error occurs in the middle of the transmission of data, then the transmission will take place again from the checkpoint. This process is known as Synchronization and recovery.

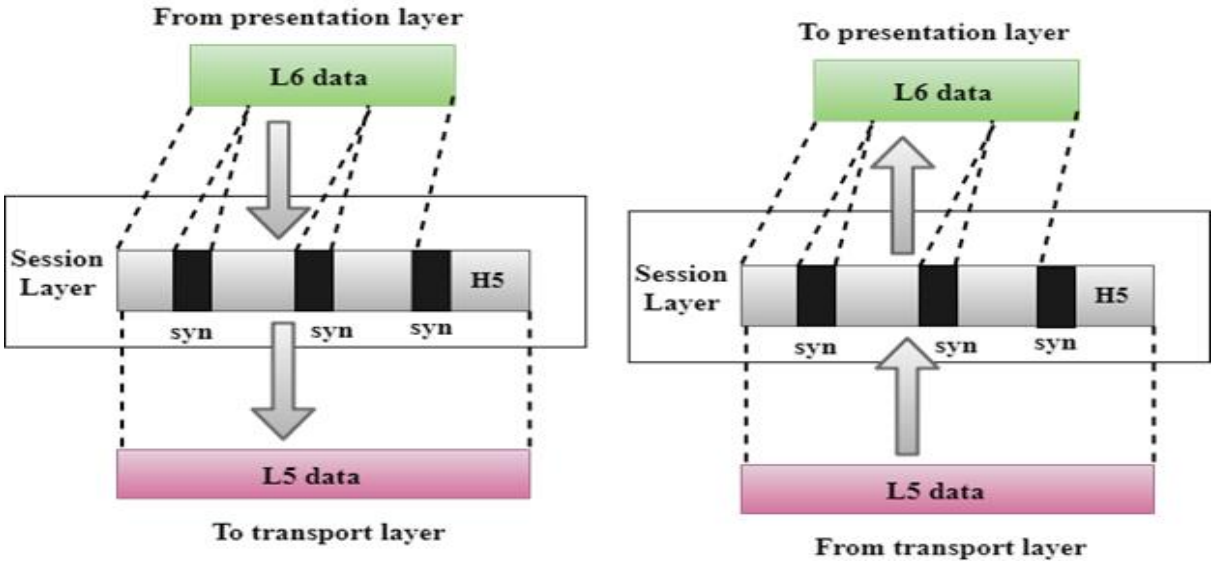

Figure 20: Session Layer

2.7 Presentation Layer Functions

- **Translation:** The processes in two systems exchange the information in the form of character strings, numbers and so on. Different computers use different encoding methods, the presentation layer handles the interoperability between the different encoding methods. It converts the data from sender-dependent format into a common format and changes the common format into receiver-dependent format at the receiving end.
- **Encryption:** Encryption is needed to maintain privacy. Encryption is a process of converting the sender-transmitted information into another form and sends the resulting message over the network.
- **Compression:** Data compression is a process of compressing the data, i.e., it reduces the number of bits to be transmitted. Data compression is very important in multimedia such as text, audio, video.

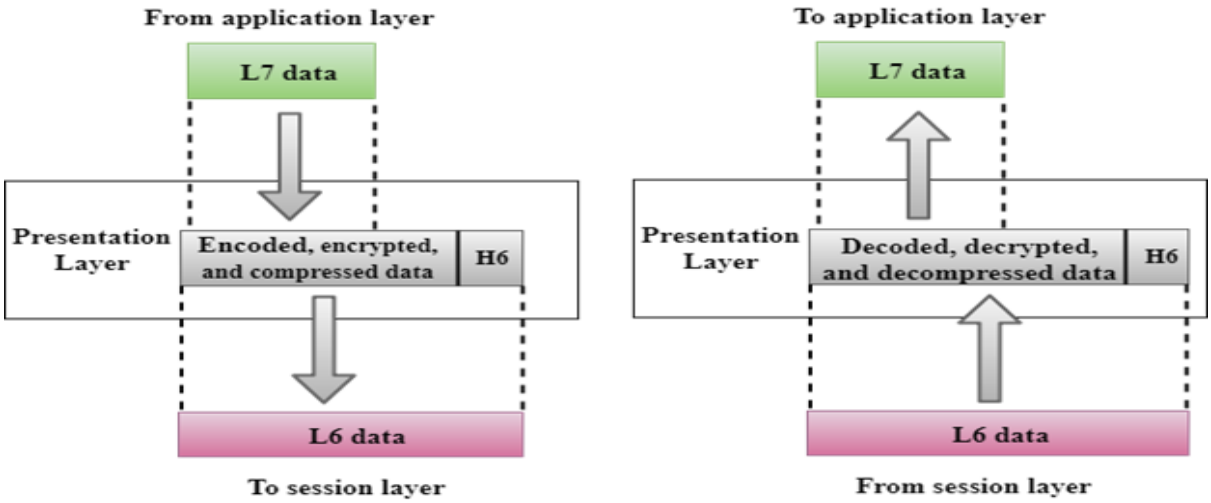

Figure 21: Presentation Layer Function

2.8 Application Layer Functions

- **File transfer, access, and management (FTAM):** An application layer allows a user to access the files in a remote computer, to retrieve the files from a computer and to manage the files in a remote computer.
- **Mail services:** An application layer provides the facility for email forwarding and storage.
- **Directory services:** An application provides the distributed database sources and is used to provide that global information about various objects.

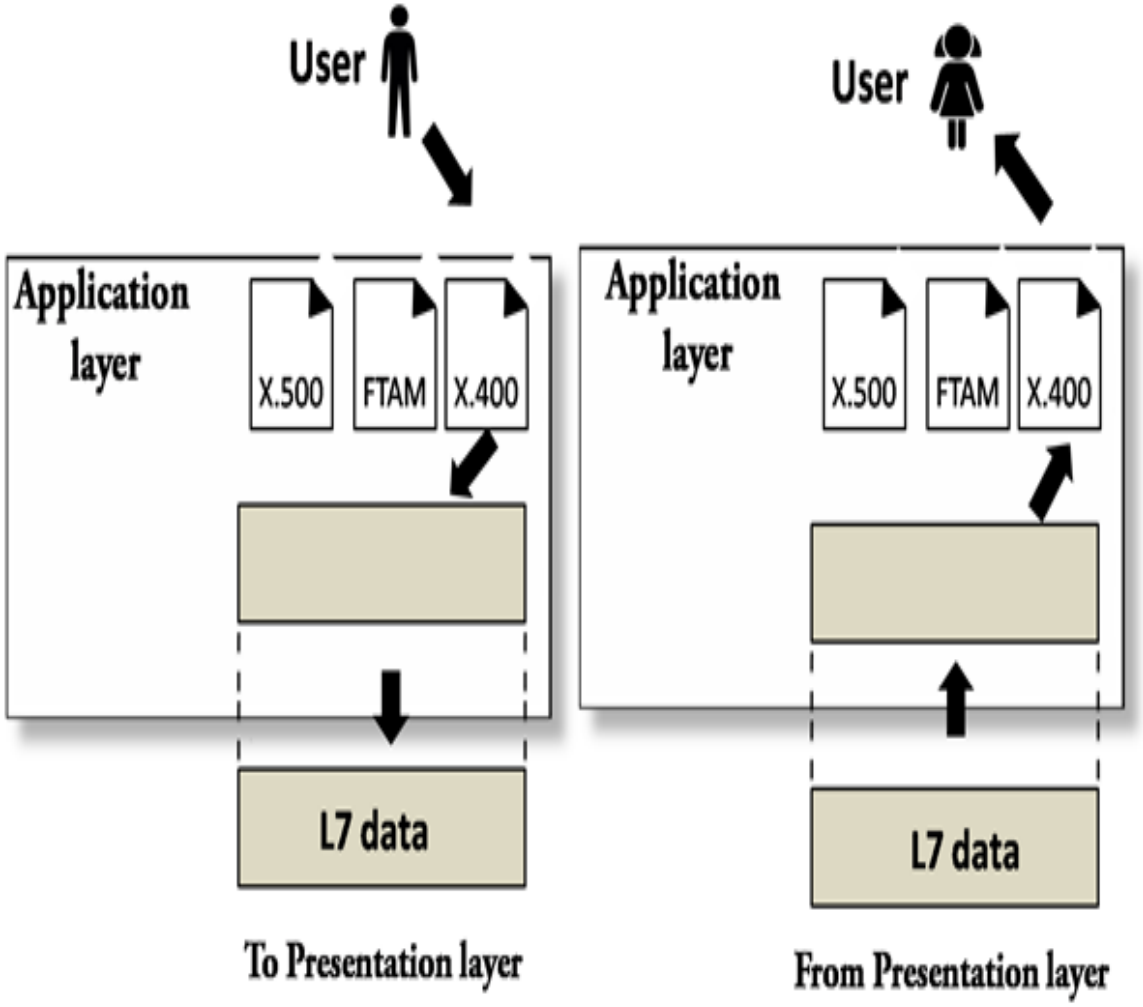

Figure 22: Application Layer

3: TCP/IP Model

3.1 What is TCP/IP Model?

TCP/IP stands for Transmission Control Protocol/ Internet Protocol. It is specifically designed as a model to offer highly reliable and end-to-end byte stream over an unreliable internetwork.

CHARACTERISTICS:

- Support for a flexible architecture i.e. adding more system to a network is easy.
- The network remains intact until the source, and destination machines were functioning properly.
- TCP is a connection-oriented protocol.
- TCP offers reliability and ensures that data which arrives out of sequence should put back into order.
- TCP allows you to implement flow control, so sender never overpowers a receiver with data.

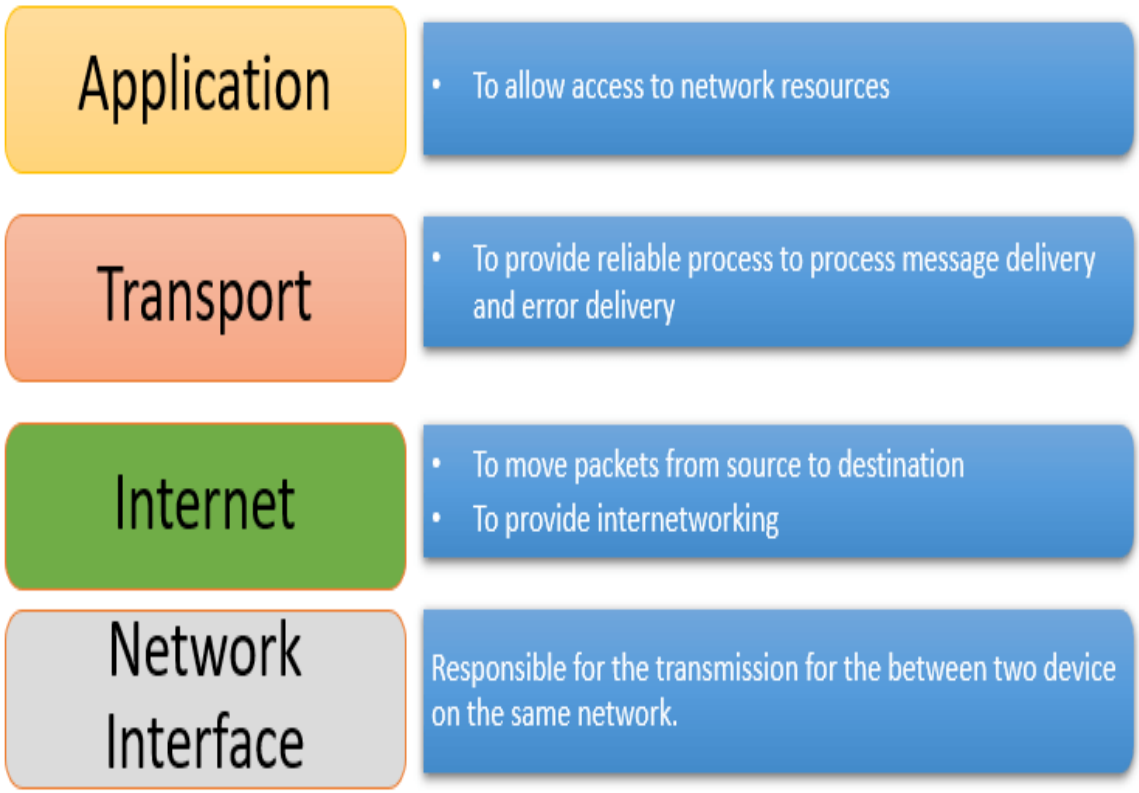

Figure 23: TCP/IP Model

3.2 Network Interface Layer Functions

- It helps you to defines details of how data should be sent using the network.
- It also includes how bits should optically be signaled by hardware devices which directly interfaces with a network medium, like coaxial, optical, coaxial, fiber, or twisted-pair cables.

3.3 Internet Layer Functions

- The main work of this layer is to send the packets from any network, and any computer still they reach the destination irrespective of the route they take.
- The Internet layer offers the functional and procedural method for transferring variable length data sequences from one node to another with the help of various networks.

3.4 Transport Layer Functions

- It divides the message received from the session layer into segments and numbers them to make a sequence.
- Transport layer makes sure that the message is delivered to the correct process on the destination machine.
- It also makes sure that the entire message arrives without any error else it should be retransmitted.

3.5 Application Layer Functions

- Application-layer helps you to identify communication partners, determining resource availability, and synchronizing communication.
- It allows users to log on to a remote host.
- This layer provides various e-mail services.
- This application offers distributed database sources and access for global information about various objects and services.

4: Topology Implementation in NS2

4.1 Wired Topology

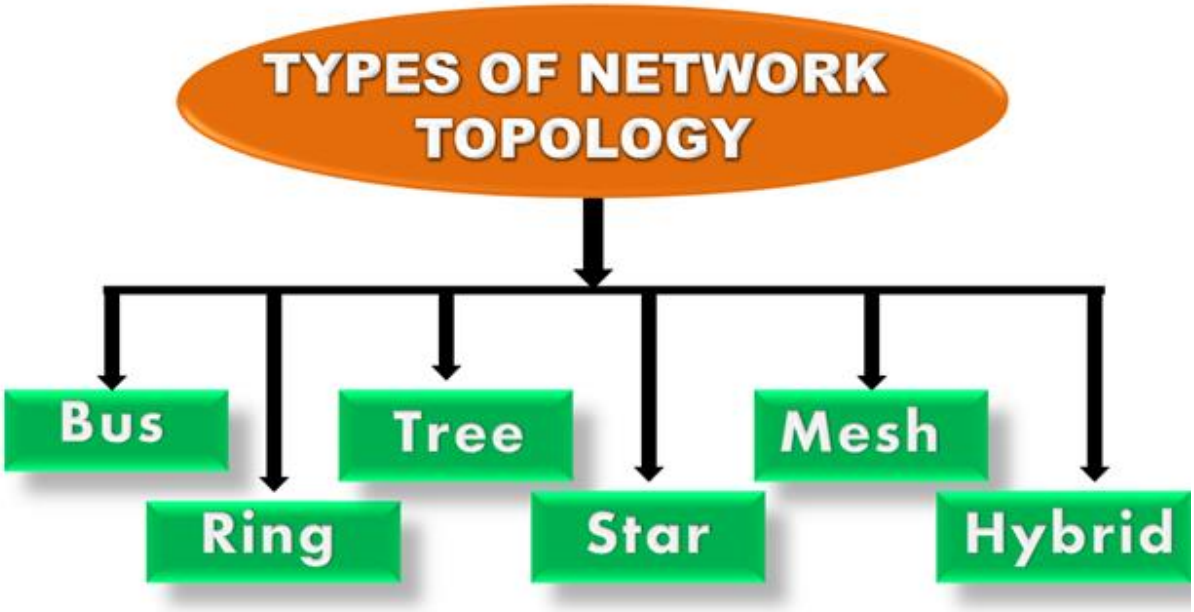

Figure 24: Types of Topology

4.1.1 Bus Topology

- The bus topology is designed in such a way that all the stations are connected through a single cable known as a backbone cable.
- Each node is either connected to the backbone cable by drop cable or directly connected to the backbone cable.
- When a node wants to send a message over the network, it puts a message over the network. All the stations available in the network will receive the message whether it has been addressed or not.
- The bus topology is mainly used in 802.3 (ethernet) and 802.4 standard networks.
- The backbone cable is considered as a "single lane" through which the message is broadcast to all the stations.
- The most common access method of the bus topologies is CSMA (Carrier Sense Multiple Access).

Figure 25: Bus Topology

Topology:

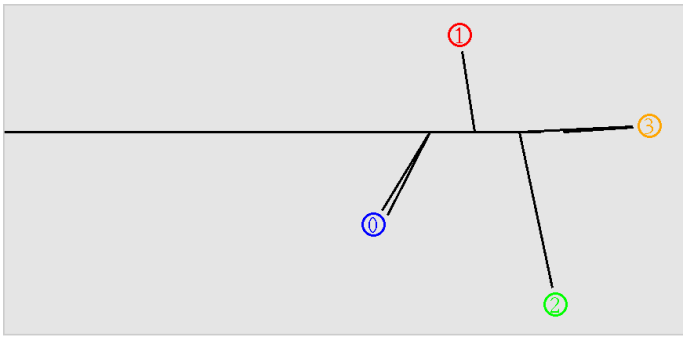

Figure 26: Bus topology output

Data Flow:

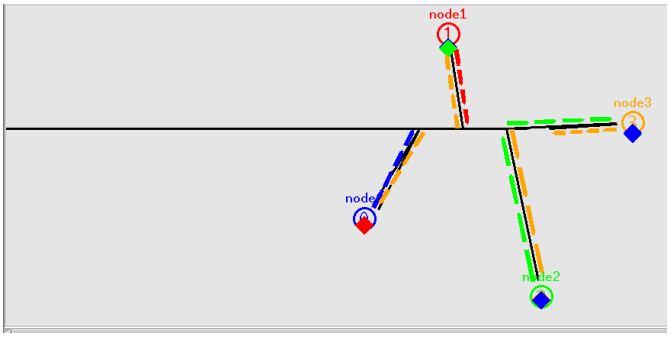

Figure 27: Bus topology data flow

4.1.2 Ring Topology

- Ring topology is like a bus topology, but with connected ends.
- The node that receives the message from the previous computer will retransmit to the next node.
- The data flows in one direction, i.e., it is unidirectional.
- The data flows in a single loop continuously known as an endless loop.
- It has no terminated ends, i.e., each node is connected to other node and having no termination point.
- The data in a ring topology flow in a clockwise direction.
- The most common access method of the ring topology is token passing.

Figure 28: Ring Topology

Topology:

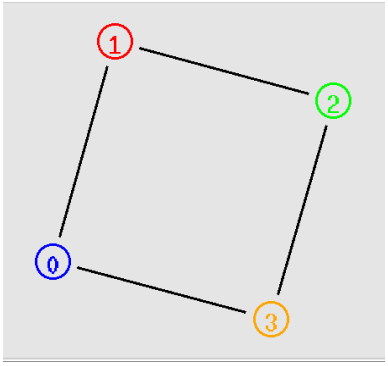

Figure 29: Ring topology output

Data Flow:

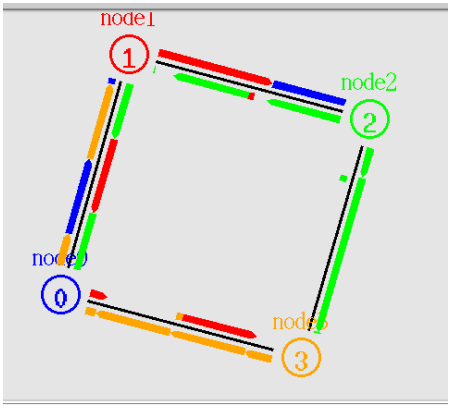

Figure 30: Ring topology data flow

4.1.3 Star Topology

- Star topology is an arrangement of the network in which every node is connected to the central hub, switch or a central computer.
- The central computer is known as a server, and the peripheral devices attached to the server are known as clients.
- Coaxial cable or RJ-45 cables are used to connect the computers.
- Hubs or Switches are mainly used as connection devices in a physical star topology.

Figure 31: Star Topology

Topology:

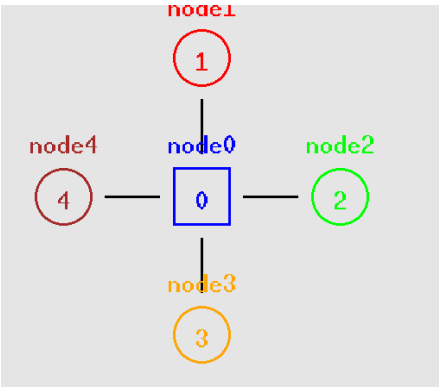

Figure 32: Star topology output

Data Flow:

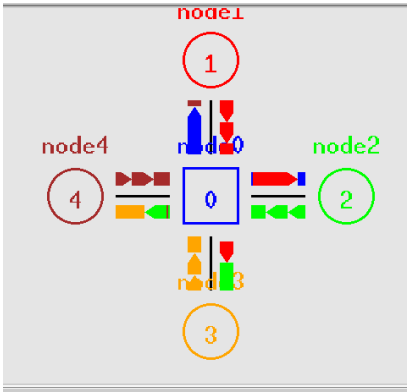

Figure 33: Star topology data flow

4.1.4 Mesh Topology

- Mesh technology is an arrangement of the network in which computers are interconnected with each other through various redundant connections.
- There are multiple paths from one computer to another computer.
- It does not contain the switch, hub or any central computer which acts as a central point of communication.
- The Internet is an example of the mesh topology.
- Mesh topology is mainly used for WAN implementations where communication failures are a critical concern.
- Number of cables = (n*(n-1))/2; Where n is the number of nodes that represents the network.

Figure 34: Mesh Topology

Topology:

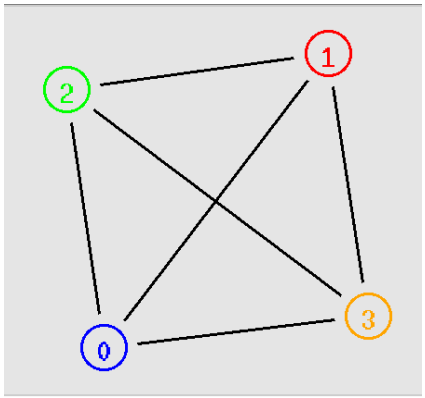

Figure 35: Mesh topology output

Data Flow:

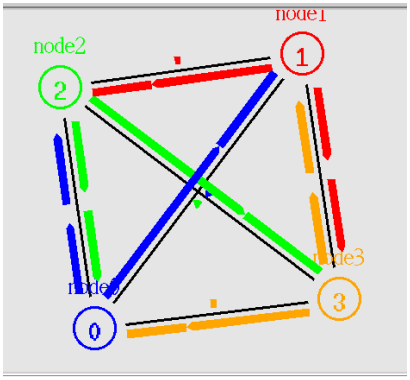

Figure 36: Mesh topology data flow

4.1.5 Tree Topology

- Tree topology combines the characteristics of bus topology and star topology.
- A tree topology is a type of structure in which all the computers are connected with each other in hierarchical fashion.
- The top-most node in tree topology is known as a root node, and all other nodes are the descendants of the root node.
- There is only one path exists between two nodes for the data transmission. Thus, it forms a parent-child hierarchy.

Figure 37: Tree Topology

Topology:

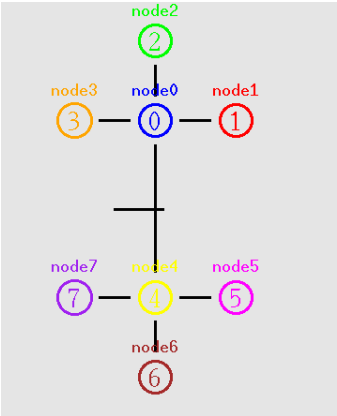

Figure 38: Tree Topology

Data Flow:

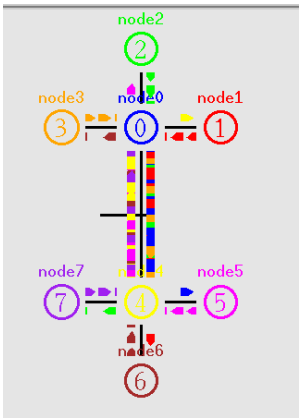

Figure 39: Tree topology data flow

4.2 Wireless Networks

Computer networks that are not connected by cables are called wireless networks. They generally use radio waves for communication between the network nodes. They allow devices to be connected to the network while roaming around within the network coverage.

4.2.1 Simulation using AODV routing protocol

Initial Node Position:

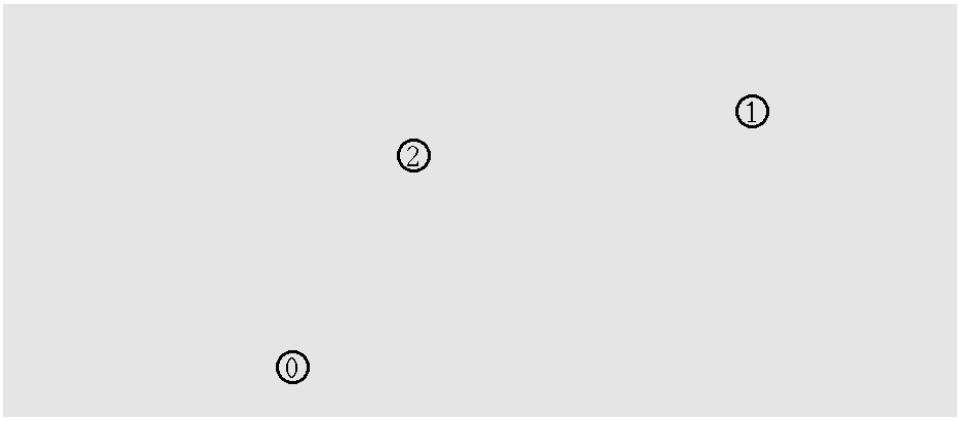

Figure 40: AODV initial node position

Data flow:

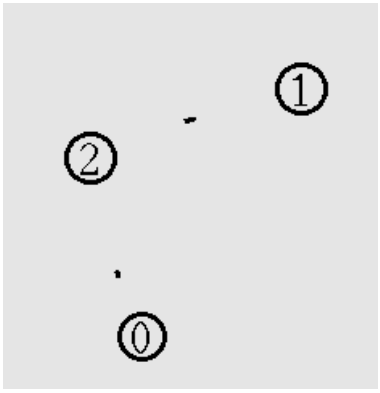

Figure 41: AODV Data Flow

4.2.2 Simulation using DSDV routing protocol

Initial Node Position:

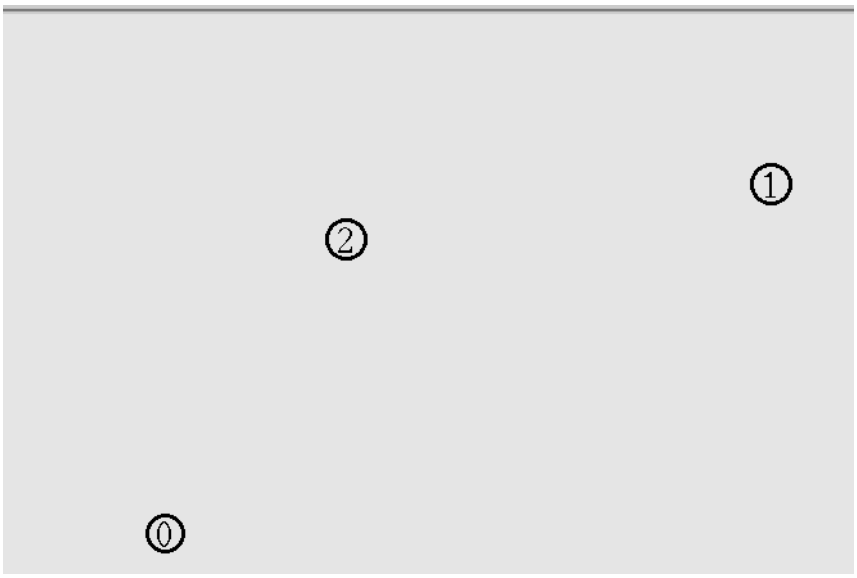

Figure 42: DSDV Initial Node Position

Data Flow:

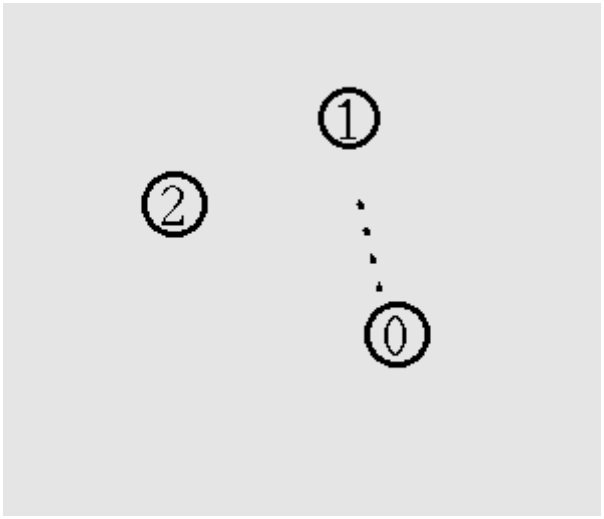

Figure 43: DSDV data flow

4.3 Introduction to NS2

Network simulation **(NS)** is one of the types of simulation, which is used to simulate the networks such as in MANETs, VANETs etc. NS-2 can be used to implement network protocols such as TCP and UPD, traffic source behavior such as FTP, Telnet, Web, CBR and VBR, router queue management mechanism such as Drop Tail, RED and CBQ, routing algorithms and many more.

4.3.1 Installation of NS2 in Linux

Open Terminal in linux then type

```
sudo apt-get install ns2
```

This will then prompt the user for a y/n to proceed to install then press y and enter.

Nam is also needed to install. Nam (Network Animator) is an animation tool to graphically represent the network and packet traces. Use this command :

```
sudo apt-get install nam
```

This will then prompt the user for a y/n to proceed to install then press y and enter.

4.3.2 TCL Scripting

Now we are going to write a 'template' that you can use for all of the first TCL scripts. You can write your TCL scripts in any text editor like joe or emacs. I suggest that you call this first example 'example1.TCL'.

First of all, you need to create a simulator object. This is done with the command

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

Now we open a file for writing that is going to be used for the nam trace data.

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

The first line opens the file 'out.nam' for writing and gives it the file handle 'nf'. In the second line we tell the simulator object that we created above to write all simulation data that is going to be relevant for nam into this file.

The next step is to add a 'finish' procedure that closes the trace file and starts nam.

```
proc finish {} {
global ns nf
$ns flush-trace
close $nf
exec nam out.nam &
exit 0
}
```

The next line tells the simulator object to execute the 'finish' procedure after 5.0 seconds of simulation time.

```
$ns at 5.0 "finish"
```

The last line finally starts the simulation.

```
$ns run
```

4.3.3 Starting NS2

You start ns with the command 'ns <tclscript>' (assuming that you are in the directory with the ns executable, or that your path points to that directory), where '<tclscript>' is the name ofa Tcl script file which defines the simulation scenario (i.e. the topology and the events). You could also just start ns without any arguments and enter the Tcl commands in the Tcl shell,but that is definitely less comfortable.

References

- http://www.jgyan.com/ns2/
- http://www.ijesrt.com/issues%20pdf%20file/Archive-2016/December-2016/36.pdf
- https://www.isi.edu/nsnam/ns/tutorial/
- https://www.youtube.com/watch?v=ZAjYDHpVplg&list=PLtRbi3COBc5lx3st89v1Z3Tzpc4wjYx3W
- https://www.javatpoint.com/computer-network-tutorial
- https://www.tutorialspoint.com/data_communication_computer_network/index.htm
- https://www.geeksforgeeks.org/computer-network-tutorials/
- https://www.amazon.com/Computer-Networks-Andrew-S-Tanenbaum-ebook/dp/B006Y1BKGC

TCL Scripts for Topology

Bus Topology:

```
#creating simulator object
set ns [new Simulator]

#Open NAM file in write mode (nf is NAM file handle)
set nf [open out.nam w]
#to write all details in the file handler i.e. NAM file
$ns namtrace-all $nf

#open trace file in write mode (nt is TRACE file handler)
set nt [open test.tr w]
#to write all detials in the trace file
$ns trace-all $nt

#define  a 'finish' procedure
proc finish {} {
      global ns nf nt
      $ns flush-trace
      #close the NAM & trace file
      close $nf
      close $nt
      #Excute NAM on trace file
      exec nam out.nam &
      exit 0
}

#creating four nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

#seting color & shape of nodes
$n0 color blue
$n1 color red
$n2 color green
$n3 color orange

#defining colors for data flow
$ns color 0 Blue
$ns color 1 Red
$ns color 2 Green
$ns color 3 Orange

#creating links between the nodes with bandwidth of 1 MB, delay of
10 milli-seconds
#droptail & SFQ are active queue management algorithm
set lan [$ns newLan "$n0 $n1 $n2 $n3" 1Mb 40ms LL Queue/DropTail
MAC/Csma/Cd Channel]

#orientation of nodes


#label the nodes
$ns at 0.0 "$n0 label node0"
$ns at 0.0 "$n1 label node1"
$ns at 0.0 "$n2 label node2"
$ns at 0.0 "$n3 label node3"
```

```
##############################Node n0##############################

#create a UDP agent and attach it to node n0
set udp0 [new Agent/UDP]
$udp0 set fid_ 0
$ns attach-agent $n0 $udp0

#create a CBR traffic source & attach it to UDP0 i.e node n0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0

#create a TCP agent and attach it to node n0
set tcp0 [new Agent/TCP]
$tcp0 set fid_ 0
$ns attach-agent $n0 $tcp0

#create a FTP source & attach it to TCP0 i.e node n0
set ftp0 [new Application/FTP]
$ftp0 set packetSize_ 500
$ftp0 set interval_ 0.005
$ftp0 attach-agent $tcp0

#Create a UDP SINK agent (a traffic sink) & attach it to node n0
set udp_sink0 [new Agent/Null]
$ns attach-agent $n0 $udp_sink0

#Create a TCP SINK agent (a traffic sink) & attach it to node n0
set tcp_sink0 [new Agent/TCPSink]
$ns attach-agent $n0 $tcp_sink0

##################################Node n1##########################

#create a UDP agent and attach it to node n1
set udp1 [new Agent/UDP]
$udp1 set fid_ 1
$ns attach-agent $n1 $udp1

#create a CBR traffic source & attach it to UDP1 i.e node n1
set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 500
$cbr1 set interval_ 0.005
$cbr1 attach-agent $udp1

#create a TCP agent and attach it to node n1
set tcp1 [new Agent/TCP]
$tcp1 set fid_ 1
$ns attach-agent $n1 $tcp1

#create a FTP source & attach it to TCP1 i.e node n1
set ftp1 [new Application/FTP]
$ftp1 set packetSize_ 500
$ftp1 set interval_ 0.005
$ftp1 attach-agent $tcp1

#Create a UDP SINK agent (a traffic sink) & attach it to node n1
set udp_sink1 [new Agent/Null]
$ns attach-agent $n1 $udp_sink1

#Create a TCP SINK agent (a traffic sink) & attach it to node n1
set tcp_sink1 [new Agent/TCPSink]
$ns attach-agent $n1 $tcp_sink1
```

```
##############################Node n2##########################

#create a UDP agent and attach it to node n2
set udp2 [new Agent/UDP]
$udp2 set fid_ 2
$ns attach-agent $n2 $udp2

#create a CBR traffic source & attach it to UDP2 i.e node n2
set cbr2 [new Application/Traffic/CBR]
$cbr2 set packetSize_ 500
$cbr2 set interval_ 0.005
$cbr2 attach-agent $udp2

#create a TCP agent and attach it to node n2
set tcp2 [new Agent/TCP]
$tcp2 set fid_ 2
$ns attach-agent $n2 $tcp2

#create a FTP source & attach it to TCP2 i.e node n2
set ftp2 [new Application/FTP]
$ftp2 set packetSize_ 500
$ftp2 set interval_ 0.005
$ftp2 attach-agent $tcp2

#Create a UDP SINK agent (a traffic sink) & attach it to node n2
set udp_sink2 [new Agent/Null]
$ns attach-agent $n2 $udp_sink2

#Create a TCP SINK agent (a traffic sink) & attach it to node n2
set tcp_sink2 [new Agent/TCPSink]
$ns attach-agent $n2 $tcp_sink2

############################Node n3##############################

#create a UDP agent and attach it to node n3
set udp3 [new Agent/UDP]
$udp3 set fid_ 3
$ns attach-agent $n3 $udp3

#create a CBR traffic source & attach it to UDP3 i.e node n3
set cbr3 [new Application/Traffic/CBR]
$cbr3 set packetSize_ 500
$cbr3 set interval_ 0.005
$cbr3 attach-agent $udp3

#create a TCP agent and attach it to node n3
set tcp3 [new Agent/TCP]
$tcp3 set fid_ 3
$ns attach-agent $n3 $tcp3

#create a FTP source & attach it to TCP3 i.e node n3
set ftp3 [new Application/FTP]
$ftp3 set packetSize_ 500
$ftp3 set interval_ 0.005
$ftp3 attach-agent $tcp3

#Create a UDP SINK agent (a traffic sink) & attach it to node n3
set udp_sink3 [new Agent/Null]
$ns attach-agent $n3 $udp_sink3
#Create a TCP SINK agent (a traffic sink) & attach it to node n3
set tcp_sink3 [new Agent/TCPSink]
$ns attach-agent $n3 $tcp_sink3
```

```
#################################################################

#connect the traffic sources with the traffic sink
$ns connect $tcp0 $tcp_sink1
$ns connect $udp0 $udp_sink2

$ns connect $tcp1 $tcp_sink2
$ns connect $udp1 $udp_sink3

$ns connect $tcp2 $tcp_sink3
$ns connect $udp2 $udp_sink0

$ns connect $tcp3 $tcp_sink0
$ns connect $udp3 $udp_sink1

#Schedule events for the FTP agents. Simulation time are given in
seconds
$ns at 0.5 "$ftp0 start"
$ns at 0.5 "$cbr0 start"
$ns at 0.5 "$ftp1 start"
$ns at 0.5 "$cbr1 start"
$ns at 0.5 "$ftp2 start"
$ns at 0.5 "$cbr2 start"
$ns at 0.5 "$ftp3 start"
$ns at 0.5 "$cbr3 start"

$ns at 4.5 "$ftp0 stop"
$ns at 4.5 "$cbr0 stop"
$ns at 4.5 "$ftp1 stop"
$ns at 4.5 "$cbr1 stop"
$ns at 4.5 "$ftp2 stop"
$ns at 4.5 "$cbr2 stop"
$ns at 4.5 "$ftp3 stop"
$ns at 4.5 "$cbr3 stop"

#Call the finish procedure after 5 seconds of simulation time
#finish procedure closes the nf which is file handler of NAM trace
file and opens the network animator
$ns at 5.0 "finish"

#Run the simulation
$ns run
```

Ring Topology:

```
#creating simulator object
set ns [new Simulator]

#Open NAM file in write mode (nf is NAM file handle)
set nf [open out.nam w]
#to write all details in the file handler i.e. NAM file
$ns namtrace-all $nf

#open trace file in write mode (nt is TRACE file handler)
set nt [open test.tr w]
#to write all detials in the trace file
$ns trace-all $nt

#define  a 'finish' procedure
proc finish {} {
     global ns nf nt
     $ns flush-trace
     #close the NAM & trace file
     close $nf
     close $nt
     #Excute NAM on trace file
     exec nam out.nam &
     exit 0
}

#creating four nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

#seting color & shape of nodes
$n0 color blue
$n1 color red
$n2 color green
$n3 color orange

#defining colors for data flow
$ns color 0 Blue
$ns color 1 Red
$ns color 2 Green
$ns color 3 Orange

#creating links between the nodes with bandwidth of 1 MB, delay of
10 milli-seconds
#droptail & SFQ are active queue management algorithm
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n2 $n3 1Mb 10ms DropTail
$ns duplex-link $n3 $n0 1Mb 10ms DropTail

#orientation of nodes
$ns duplex-link-op $n0 $n1 orient right
$ns duplex-link-op $n1 $n2 orient down
$ns duplex-link-op $n2 $n3 orient left

#label the nodes
$ns at 0.0 "$n0 label node0"
$ns at 0.0 "$n1 label node1"
$ns at 0.0 "$n2 label node2"
$ns at 0.0 "$n3 label node3"
```

```
##########################Node n0#################################

#create a UDP agent and attach it to node n0
set udp0 [new Agent/UDP]
$udp0 set fid_ 0
$ns attach-agent $n0 $udp0

#create a CBR traffic source & attach it to UDP0 i.e node n0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0

#create a TCP agent and attach it to node n0
set tcp0 [new Agent/TCP]
$tcp0 set fid_ 0
$ns attach-agent $n0 $tcp0

#create a FTP source & attach it to TCP0 i.e node n0
set ftp0 [new Application/FTP]
$ftp0 set packetSize_ 500
$ftp0 set interval_ 0.005
$ftp0 attach-agent $tcp0

#Create a UDP SINK agent (a traffic sink) & attach it to node n0
set udp_sink0 [new Agent/Null]
$ns attach-agent $n0 $udp_sink0

#Create a TCP SINK agent (a traffic sink) & attach it to node n0
set tcp_sink0 [new Agent/TCPSink]
$ns attach-agent $n0 $tcp_sink0

##############################Node n1#########################

#create a UDP agent and attach it to node n1
set udp1 [new Agent/UDP]
$udp1 set fid_ 1
$ns attach-agent $n1 $udp1

#create a CBR traffic source & attach it to UDP1 i.e node n1
set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 500
$cbr1 set interval_ 0.005
$cbr1 attach-agent $udp1

#create a TCP agent and attach it to node n1
set tcp1 [new Agent/TCP]
$tcp1 set fid_ 1
$ns attach-agent $n1 $tcp1

#create a FTP source & attach it to TCP1 i.e node n1
set ftp1 [new Application/FTP]
$ftp1 set packetSize_ 500
$ftp1 set interval_ 0.005
$ftp1 attach-agent $tcp1

#Create a UDP SINK agent (a traffic sink) & attach it to node n1
set udp_sink1 [new Agent/Null]
$ns attach-agent $n1 $udp_sink1

#Create a TCP SINK agent (a traffic sink) & attach it to node n1
set tcp_sink1 [new Agent/TCPSink]
$ns attach-agent $n1 $tcp_sink1
```

```
################################Node n2#########################

#create a UDP agent and attach it to node n2
set udp2 [new Agent/UDP]
$udp2 set fid_ 2
$ns attach-agent $n2 $udp2

#create a CBR traffic source & attach it to UDP2 i.e node n2
set cbr2 [new Application/Traffic/CBR]
$cbr2 set packetSize_ 500
$cbr2 set interval_ 0.005
$cbr2 attach-agent $udp2

#create a TCP agent and attach it to node n2
set tcp2 [new Agent/TCP]
$tcp2 set fid_ 2
$ns attach-agent $n2 $tcp2

#create a FTP source & attach it to TCP2 i.e node n2
set ftp2 [new Application/FTP]
$ftp2 set packetSize_ 500
$ftp2 set interval_ 0.005
$ftp2 attach-agent $tcp2

#Create a UDP SINK agent (a traffic sink) & attach it to node n2
set udp_sink2 [new Agent/Null]
$ns attach-agent $n2 $udp_sink2

#Create a TCP SINK agent (a traffic sink) & attach it to node n2
set tcp_sink2 [new Agent/TCPSink]
$ns attach-agent $n2 $tcp_sink2

##############################Node n3#################################
#create a UDP agent and attach it to node n3
set udp3 [new Agent/UDP]
$udp3 set fid_ 3
$ns attach-agent $n3 $udp3

#create a CBR traffic source & attach it to UDP3 i.e node n3
set cbr3 [new Application/Traffic/CBR]
$cbr3 set packetSize_ 500
$cbr3 set interval_ 0.005
$cbr3 attach-agent $udp3

#create a TCP agent and attach it to node n3
set tcp3 [new Agent/TCP]
$tcp3 set fid_ 3
$ns attach-agent $n3 $tcp3

#create a FTP source & attach it to TCP3 i.e node n3
set ftp3 [new Application/FTP]
$ftp3 set packetSize_ 500
$ftp3 set interval_ 0.005
$ftp3 attach-agent $tcp3

#Create a UDP SINK agent (a traffic sink) & attach it to node n3
set udp_sink3 [new Agent/Null]
$ns attach-agent $n3 $udp_sink3
#Create a TCP SINK agent (a traffic sink) & attach it to node n3
set tcp_sink3 [new Agent/TCPSink]
$ns attach-agent $n3 $tcp_sink3

#################################################################
```

```
#connect the traffic sources with the traffic sink
$ns connect $tcp0 $tcp_sink1
$ns connect $udp0 $udp_sink2

$ns connect $tcp1 $tcp_sink2
$ns connect $udp1 $udp_sink3

$ns connect $tcp2 $tcp_sink3
$ns connect $udp2 $udp_sink0

$ns connect $tcp3 $tcp_sink0
$ns connect $udp3 $udp_sink1

#Schedule events for the FTP agents. Simulation time are given in
seconds
$ns at 0.5 "$ftp0 start"
$ns at 0.5 "$cbr0 start"
$ns at 0.5 "$ftp1 start"
$ns at 0.5 "$cbr1 start"
$ns at 0.5 "$ftp2 start"
$ns at 0.5 "$cbr2 start"
$ns at 0.5 "$ftp3 start"
$ns at 0.5 "$cbr3 start"

$ns at 4.5 "$ftp0 stop"
$ns at 4.5 "$cbr0 stop"
$ns at 4.5 "$ftp1 stop"
$ns at 4.5 "$cbr1 stop"
$ns at 4.5 "$ftp2 stop"
$ns at 4.5 "$cbr2 stop"
$ns at 4.5 "$ftp3 stop"
$ns at 4.5 "$cbr3 stop"

#Call the finish procedure after 5 seconds of simulation time
#finish procedure closes the nf which is file handler of NAM trace
file and opens the network animator
$ns at 5.0 "finish"

#Run the simulation
$ns run
```

Star Topology:

```
#creating simulator object
set ns [new Simulator]

#Open NAM file in write mode (nf is NAM file handle)
set nf [open out.nam w]
#to write all details in the file handler i.e. NAM file
$ns namtrace-all $nf

#open trace file in write mode (nt is TRACE file handler)
set nt [open test.tr w]
#to write all detials in the trace file
$ns trace-all $nt

#define  a 'finish' procedure
proc finish {} {
     global ns nf nt
     $ns flush-trace
     #close the NAM & trace file
     close $nf
     close $nt
     #Excute NAM on trace file
     exec nam out.nam &
     exit 0
}

#creating four nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]

#seting color & shape of nodes
$n0 color blue
$n0 shape box

$n1 color red
$n2 color green
$n3 color orange
$n4 color brown

#defining colors for data flow
$ns color 0 Blue
$ns color 1 Red
$ns color 2 Green
$ns color 3 Orange
$ns color 4 Brown

#creating links between the nodes with bandwidth of 1 MB, delay of
10 milli-seconds
#droptail & SFQ are active queue management algorithm
$ns duplex-link $n1 $n0 1Mb 10ms DropTail
$ns duplex-link $n2 $n0 1Mb 10ms DropTail
$ns duplex-link $n3 $n0 1Mb 10ms DropTail
$ns duplex-link $n4 $n0 1Mb 10ms DropTail

#orientation of nodes
$ns duplex-link-op $n0 $n1 orient up
$ns duplex-link-op $n0 $n2 orient right
$ns duplex-link-op $n0 $n3 orient down
$ns duplex-link-op $n0 $n4 orient left
```

```
#label the nodes
$ns at 0.0 "$n0 label node0"
$ns at 0.0 "$n1 label node1"
$ns at 0.0 "$n2 label node2"
$ns at 0.0 "$n3 label node3"
$ns at 0.0 "$n4 label node4"

################################Node n0##############################

#create a UDP agent and attach it to node n0
set udp0 [new Agent/UDP]
$udp0 set fid_ 0
$ns attach-agent $n0 $udp0

#create a CBR traffic source & attach it to UDP0 i.e node n0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0

#create a TCP agent and attach it to node n0
set tcp0 [new Agent/TCP]
$tcp0 set fid_ 0
$ns attach-agent $n0 $tcp0

#create a FTP source & attach it to TCP0 i.e node n0
set ftp0 [new Application/FTP]
$ftp0 set packetSize_ 500
$ftp0 set interval_ 0.005
$ftp0 attach-agent $tcp0

#Create a UDP SINK agent (a traffic sink) & attach it to node n0
set udp_sink0 [new Agent/Null]
$ns attach-agent $n0 $udp_sink0

#Create a TCP SINK agent (a traffic sink) & attach it to node n0
set tcp_sink0 [new Agent/TCPSink]
$ns attach-agent $n0 $tcp_sink0

#####################################Node n1######################
#create a UDP agent and attach it to node n1
set udp1 [new Agent/UDP]
$udp1 set fid_ 1
$ns attach-agent $n1 $udp1

#create a CBR traffic source & attach it to UDP1 i.e node n1
set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 500
$cbr1 set interval_ 0.005
$cbr1 attach-agent $udp1

#create a TCP agent and attach it to node n1
set tcp1 [new Agent/TCP]
$tcp1 set fid_ 1
$ns attach-agent $n1 $tcp1

#create a FTP source & attach it to TCP1 i.e node n1
set ftp1 [new Application/FTP]
$ftp1 set packetSize_ 500
$ftp1 set interval_ 0.005
$ftp1 attach-agent $tcp1

#Create a UDP SINK agent (a traffic sink) & attach it to node n1
set udp_sink1 [new Agent/Null]
```

```
$ns attach-agent $n1 $udp_sink1

#Create a TCP SINK agent (a traffic sink) & attach it to node n1
set tcp_sink1 [new Agent/TCPSink]
$ns attach-agent $n1 $tcp_sink1

##############################Node n2################################

#create a UDP agent and attach it to node n2
set udp2 [new Agent/UDP]
$udp2 set fid_ 2
$ns attach-agent $n2 $udp2

#create a CBR traffic source & attach it to UDP2 i.e node n2
set cbr2 [new Application/Traffic/CBR]
$cbr2 set packetSize_ 500
$cbr2 set interval_ 0.005
$cbr2 attach-agent $udp2

#create a TCP agent and attach it to node n2
set tcp2 [new Agent/TCP]
$tcp2 set fid_ 2
$ns attach-agent $n2 $tcp2

#create a FTP source & attach it to TCP2 i.e node n2
set ftp2 [new Application/FTP]
$ftp2 set packetSize_ 500
$ftp2 set interval_ 0.005
$ftp2 attach-agent $tcp2

#Create a UDP SINK agent (a traffic sink) & attach it to node n2
set udp_sink2 [new Agent/Null]
$ns attach-agent $n2 $udp_sink2

#Create a TCP SINK agent (a traffic sink) & attach it to node n2
set tcp_sink2 [new Agent/TCPSink]
$ns attach-agent $n2 $tcp_sink2

##################################Node n3##############################
#create a UDP agent and attach it to node n3
set udp3 [new Agent/UDP]
$udp3 set fid_ 3
$ns attach-agent $n3 $udp3

#create a CBR traffic source & attach it to UDP3 i.e node n3
set cbr3 [new Application/Traffic/CBR]
$cbr3 set packetSize_ 500
$cbr3 set interval_ 0.005
$cbr3 attach-agent $udp3

#create a TCP agent and attach it to node n3
set tcp3 [new Agent/TCP]
$tcp3 set fid_ 3
$ns attach-agent $n3 $tcp3

#create a FTP source & attach it to TCP3 i.e node n3
set ftp3 [new Application/FTP]
$ftp3 set packetSize_ 500
$ftp3 set interval_ 0.005
$ftp3 attach-agent $tcp3

#Create a UDP SINK agent (a traffic sink) & attach it to node n3
set udp_sink3 [new Agent/Null]
$ns attach-agent $n3 $udp_sink3
```

```
#Create a TCP SINK agent (a traffic sink) & attach it to node n3
set tcp_sink3 [new Agent/TCPSink]
$ns attach-agent $n3 $tcp_sink3

##############################Node n4 ############################

#create a UDP agent and attach it to node n4
set udp4 [new Agent/UDP]
$udp4 set fid_ 4
$ns attach-agent $n4 $udp4

#create a CBR traffic source & attach it to UDP4 i.e node n4
set cbr4 [new Application/Traffic/CBR]
$cbr4 set packetSize_ 500
$cbr4 set interval_ 0.005
$cbr4 attach-agent $udp4

#create a TCP agent and attach it to node n4
set tcp4 [new Agent/TCP]
$tcp4 set fid_ 4
$ns attach-agent $n4 $tcp4

#create a FTP source & attach it to TCP4 i.e node n4
set ftp4 [new Application/FTP]
$ftp4 set packetSize_ 500
$ftp4 set interval_ 0.005
$ftp4 attach-agent $tcp4

#Create a UDP SINK agent (a traffic sink) & attach it to node n4
set udp_sink4 [new Agent/Null]
$ns attach-agent $n4 $udp_sink4
#Create a TCP SINK agent (a traffic sink) & attach it to node n4
set tcp_sink4 [new Agent/TCPSink]
$ns attach-agent $n4 $tcp_sink4

#####################################################################

#Connect the traffic sources with the traffic sink
$ns connect $tcp0 $tcp_sink1
$ns connect $udp0 $udp_sink2

$ns connect $tcp1 $tcp_sink2
$ns connect $udp1 $udp_sink3

$ns connect $tcp2 $tcp_sink3
$ns connect $udp2 $udp_sink4

$ns connect $tcp3 $tcp_sink4
$ns connect $udp3 $udp_sink0

$ns connect $tcp4 $tcp_sink0
$ns connect $udp4 $udp_sink1

#Schedule events for the CBR agents. Simulation time are given in
seconds
$ns at 0.5 "$ftp0 start"
$ns at 0.5 "$cbr0 start"
$ns at 0.5 "$ftp1 start"
$ns at 0.5 "$cbr1 start"
$ns at 0.5 "$ftp2 start"
$ns at 0.5 "$cbr2 start"
$ns at 0.5 "$ftp3 start"
$ns at 0.5 "$cbr3 start"
$ns at 0.5 "$ftp4 start"
```

```
$ns at 0.5 "$cbr4 start"

$ns at 4.5 "$ftp0 stop"
$ns at 4.5 "$cbr0 stop"
$ns at 4.5 "$ftp1 stop"
$ns at 4.5 "$cbr1 stop"
$ns at 4.5 "$ftp2 stop"
$ns at 4.5 "$cbr2 stop"
$ns at 4.5 "$ftp3 stop"
$ns at 4.5 "$cbr3 stop"
$ns at 4.5 "$ftp4 stop"
$ns at 4.5 "$cbr4 stop"

#Call the finish procedure after 5 seconds of simulation time
#finish procedure closes the nf which is file handler of NAM trace
file and opens the network animator
$ns at 5.0 "finish"

#Run the simulation
$ns run
```

Mesh Topology:

```
#creating simulator object
set ns [new Simulator]

#Open NAM file in write mode (nf is NAM file handle)
set nf [open out.nam w]
#to write all details in the file handler i.e. NAM file
$ns namtrace-all $nf

#open trace file in write mode (nt is TRACE file handler)
set nt [open test.tr w]
#to write all detials in the trace file
$ns trace-all $nt

#define  a 'finish' procedure
proc finish {} {
      global ns nf nt
      $ns flush-trace
      #close the NAM & trace file
      close $nf
      close $nt
      #Excute NAM on trace file
      exec nam out.nam &
      exit 0
}

#creating four nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

#seting color & shape of nodes
$n0 color blue
$n1 color red
$n2 color green
$n3 color orange

#defining colors for data flow
$ns color 0 Blue
$ns color 1 Red
$ns color 2 Green
$ns color 3 Orange

#creating links between the nodes with bandwidth of 1 MB, delay of
10 milli-seconds
#droptail & SFQ are active queue management algorithm
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
$ns duplex-link $n0 $n2 1Mb 10ms DropTail
$ns duplex-link $n0 $n3 1Mb 10ms DropTail

$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n1 $n3 1Mb 10ms DropTail

$ns duplex-link $n2 $n3 1Mb 10ms DropTail

#orientation of nodes
$ns duplex-link-op $n0 $n1 orient right
$ns duplex-link-op $n1 $n2 orient down
$ns duplex-link-op $n2 $n3 orient left

#label the nodes
$ns at 0.0 "$n0 label node0"
```

```
$ns at 0.0 "$n1 label node1"
$ns at 0.0 "$n2 label node2"
$ns at 0.0 "$n3 label node3"

##############################Node n0##################

#create a UDP agent and attach it to node n0
set udp0 [new Agent/UDP]
$udp0 set fid_ 0
$ns attach-agent $n0 $udp0

#create a CBR traffic source & attach it to UDP0 i.e node n0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0

#create a TCP agent and attach it to node n0
set tcp0 [new Agent/TCP]
$tcp0 set fid_ 0
$ns attach-agent $n0 $tcp0

#create a FTP source & attach it to TCP0 i.e node n0
set ftp0 [new Application/FTP]
$ftp0 set packetSize_ 500
$ftp0 set interval_ 0.005
$ftp0 attach-agent $tcp0

#Create a UDP SINK agent (a traffic sink) & attach it to node n0
set udp_sink0 [new Agent/Null]
$ns attach-agent $n0 $udp_sink0

#Create a TCP SINK agent (a traffic sink) & attach it to node n0
set tcp_sink0 [new Agent/TCPSink]
$ns attach-agent $n0 $tcp_sink0

##############################Node n1##################################

#create a UDP agent and attach it to node n1
set udp1 [new Agent/UDP]
$udp1 set fid_ 1
$ns attach-agent $n1 $udp1

#create a CBR traffic source & attach it to UDP1 i.e node n1
set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 500
$cbr1 set interval_ 0.005
$cbr1 attach-agent $udp1

#create a TCP agent and attach it to node n1
set tcp1 [new Agent/TCP]
$tcp1 set fid_ 1
$ns attach-agent $n1 $tcp1

#create a FTP source & attach it to TCP1 i.e node n1
set ftp1 [new Application/FTP]
$ftp1 set packetSize_ 500
$ftp1 set interval_ 0.005
$ftp1 attach-agent $tcp1

#Create a UDP SINK agent (a traffic sink) & attach it to node n1
set udp_sink1 [new Agent/Null]
$ns attach-agent $n1 $udp_sink1
```

```
#Create a TCP SINK agent (a traffic sink) & attach it to node n1
set tcp_sink1 [new Agent/TCPSink]
$ns attach-agent $n1 $tcp_sink1

###################################Node n2############################

#create a UDP agent and attach it to node n2
set udp2 [new Agent/UDP]
$udp2 set fid_ 2
$ns attach-agent $n2 $udp2

#create a CBR traffic source & attach it to UDP2 i.e node n2
set cbr2 [new Application/Traffic/CBR]
$cbr2 set packetSize_ 500
$cbr2 set interval_ 0.005
$cbr2 attach-agent $udp2

#create a TCP agent and attach it to node n2
set tcp2 [new Agent/TCP]
$tcp2 set fid_ 2
$ns attach-agent $n2 $tcp2

#create a FTP source & attach it to TCP2 i.e node n2
set ftp2 [new Application/FTP]
$ftp2 set packetSize_ 500
$ftp2 set interval_ 0.005
$ftp2 attach-agent $tcp2

#Create a UDP SINK agent (a traffic sink) & attach it to node n2
set udp_sink2 [new Agent/Null]
$ns attach-agent $n2 $udp_sink2

#Create a TCP SINK agent (a traffic sink) & attach it to node n2
set tcp_sink2 [new Agent/TCPSink]
$ns attach-agent $n2 $tcp_sink2

###################################Node n3#######################

#create a UDP agent and attach it to node n3
set udp3 [new Agent/UDP]
$udp3 set fid_ 3
$ns attach-agent $n3 $udp3

#create a CBR traffic source & attach it to UDP3 i.e node n3
set cbr3 [new Application/Traffic/CBR]
$cbr3 set packetSize_ 500
$cbr3 set interval_ 0.005
$cbr3 attach-agent $udp3

#create a TCP agent and attach it to node n3
set tcp3 [new Agent/TCP]
$tcp3 set fid_ 3
$ns attach-agent $n3 $tcp3

#create a FTP source & attach it to TCP3 i.e node n3
set ftp3 [new Application/FTP]
$ftp3 set packetSize_ 500
$ftp3 set interval_ 0.005
$ftp3 attach-agent $tcp3

#Create a UDP SINK agent (a traffic sink) & attach it to node n3
set udp_sink3 [new Agent/Null]
$ns attach-agent $n3 $udp_sink3
#Create a TCP SINK agent (a traffic sink) & attach it to node n3
```

```
set tcp_sink3 [new Agent/TCPSink]
$ns attach-agent $n3 $tcp_sink3

#####################################################################

#connect the traffic sources with the traffic sink
$ns connect $tcp0 $tcp_sink1
$ns connect $udp0 $udp_sink2

$ns connect $tcp1 $tcp_sink2
$ns connect $udp1 $udp_sink3

$ns connect $tcp2 $tcp_sink3
$ns connect $udp2 $udp_sink0

$ns connect $tcp3 $tcp_sink0
$ns connect $udp3 $udp_sink1

#Schedule events for the FTP agents. Simulation time are given in
seconds
$ns at 0.5 "$ftp0 start"
$ns at 0.5 "$cbr0 start"
$ns at 0.5 "$ftp1 start"
$ns at 0.5 "$cbr1 start"
$ns at 0.5 "$ftp2 start"
$ns at 0.5 "$cbr2 start"
$ns at 0.5 "$ftp3 start"
$ns at 0.5 "$cbr3 start"

$ns at 4.5 "$ftp0 stop"
$ns at 4.5 "$cbr0 stop"
$ns at 4.5 "$ftp1 stop"
$ns at 4.5 "$cbr1 stop"
$ns at 4.5 "$ftp2 stop"
$ns at 4.5 "$cbr2 stop"
$ns at 4.5 "$ftp3 stop"
$ns at 4.5 "$cbr3 stop"

#Call the finish procedure after 5 seconds of simulation time
#finish procedure closes the nf which is file handler of NAM trace
file and opens the network animator
$ns at 5.0 "finish"

#Run the simulation
$ns run
```

Tree Topology:

```
#creating simulator object
set ns [new Simulator]

#Open NAM file in write mode (nf is NAM file handle)
set nf [open out.nam w]
#to write all details in the file handler i.e. NAM file
$ns namtrace-all $nf

#open trace file in write mode (nt is TRACE file handler)
set nt [open test.tr w]
#to write all detials in the trace file
$ns trace-all $nt

#define  a 'finish' procedure
proc finish {} {
     global ns nf nt
     $ns flush-trace
     #close the NAM & trace file
     close $nf
     close $nt
     #Excute NAM on trace file
     exec nam out.nam &
     exit 0
}

#creating four nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]
set n7 [$ns node]

#seting color & shape of nodes
$n0 color blue
$n1 color red
$n2 color green
$n3 color orange
$n4 color yellow
$n5 color magenta
$n6 color brown
$n7 color purple

#defining colors for data flow
$ns color 0 Blue
$ns color 1 Red
$ns color 2 Green
$ns color 3 Orange
$ns color 4 Yellow
$ns color 5 Magenta
$ns color 6 brown
$ns color 7 purple

#creating links between the nodes with bandwidth of 1 MB, delay of
10 milli-seconds
#droptail & SFQ are active queue management algorithm
set lan [$ns newLan "$n0 $n4" 1Mb 40ms LL Queue/DropTail MAC/Csma/Cd
Channel]
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
$ns duplex-link $n0 $n2 1Mb 10ms DropTail
```

```
$ns duplex-link $n0 $n3 1Mb 10ms DropTail

$ns duplex-link $n4 $n5 1Mb 10ms DropTail
$ns duplex-link $n4 $n6 1Mb 10ms DropTail
$ns duplex-link $n4 $n7 1Mb 10ms DropTail

#orientation of nodes
$ns duplex-link-op $n0 $n1 orient right
$ns duplex-link-op $n0 $n2 orient up
$ns duplex-link-op $n0 $n3 orient left

$ns duplex-link-op $n4 $n5 orient right
$ns duplex-link-op $n4 $n6 orient down
$ns duplex-link-op $n4 $n7 orient left

#label the nodes
$ns at 0.0 "$n0 label node0"
$ns at 0.0 "$n1 label node1"
$ns at 0.0 "$n2 label node2"
$ns at 0.0 "$n3 label node3"
$ns at 0.0 "$n4 label node4"
$ns at 0.0 "$n5 label node5"
$ns at 0.0 "$n6 label node6"
$ns at 0.0 "$n7 label node7"

###################################Node n0################

#create a UDP agent and attach it to node n0
set udp0 [new Agent/UDP]
$udp0 set fid_ 0
$ns attach-agent $n0 $udp0

#create a CBR traffic source & attach it to UDP0 i.e node n0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0

#create a TCP agent and attach it to node n0
set tcp0 [new Agent/TCP]
$tcp0 set fid_ 0
$ns attach-agent $n0 $tcp0

#create a FTP source & attach it to TCP0 i.e node n0
set ftp0 [new Application/FTP]
$ftp0 set packetSize_ 500
$ftp0 set interval_ 0.005
$ftp0 attach-agent $tcp0

#Create a UDP SINK agent (a traffic sink) & attach it to node n0
set udp_sink0 [new Agent/Null]
$ns attach-agent $n0 $udp_sink0

#Create a TCP SINK agent (a traffic sink) & attach it to node n0
set tcp_sink0 [new Agent/TCPSink]
$ns attach-agent $n0 $tcp_sink0

#######################Node n1################################

#create a UDP agent and attach it to node n1
set udp1 [new Agent/UDP]
$udp1 set fid_ 1
$ns attach-agent $n1 $udp1
```

```
#create a CBR traffic source & attach it to UDP1 i.e node n1
set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 500
$cbr1 set interval_ 0.005
$cbr1 attach-agent $udp1

#create a TCP agent and attach it to node n1
set tcp1 [new Agent/TCP]
$tcp1 set fid_ 1
$ns attach-agent $n1 $tcp1

#create a FTP source & attach it to TCP1 i.e node n1
set ftp1 [new Application/FTP]
$ftp1 set packetSize_ 500
$ftp1 set interval_ 0.005
$ftp1 attach-agent $tcp1

#Create a UDP SINK agent (a traffic sink) & attach it to node n1
set udp_sink1 [new Agent/Null]
$ns attach-agent $n1 $udp_sink1

#Create a TCP SINK agent (a traffic sink) & attach it to node n1
set tcp_sink1 [new Agent/TCPSink]
$ns attach-agent $n1 $tcp_sink1

#######################################Node n2#####################
#create a UDP agent and attach it to node n2
set udp2 [new Agent/UDP]
$udp2 set fid_ 2
$ns attach-agent $n2 $udp2

#create a CBR traffic source & attach it to UDP2 i.e node n2
set cbr2 [new Application/Traffic/CBR]
$cbr2 set packetSize_ 500
$cbr2 set interval_ 0.005
$cbr2 attach-agent $udp2

#create a TCP agent and attach it to node n2
set tcp2 [new Agent/TCP]
$tcp2 set fid_ 2
$ns attach-agent $n2 $tcp2

#create a FTP source & attach it to TCP2 i.e node n2
set ftp2 [new Application/FTP]
$ftp2 set packetSize_ 500
$ftp2 set interval_ 0.005
$ftp2 attach-agent $tcp2

#Create a UDP SINK agent (a traffic sink) & attach it to node n2
set udp_sink2 [new Agent/Null]
$ns attach-agent $n2 $udp_sink2

#Create a TCP SINK agent (a traffic sink) & attach it to node n2
set tcp_sink2 [new Agent/TCPSink]
$ns attach-agent $n2 $tcp_sink2

#################################Node n3 ##########################

#create a UDP agent and attach it to node n3
set udp3 [new Agent/UDP]
$udp3 set fid_ 3
$ns attach-agent $n3 $udp3

#create a CBR traffic source & attach it to UDP3 i.e node n3
```

```
set cbr3 [new Application/Traffic/CBR]
$cbr3 set packetSize_ 500
$cbr3 set interval_ 0.005
$cbr3 attach-agent $udp3

#create a TCP agent and attach it to node n3
set tcp3 [new Agent/TCP]
$tcp3 set fid_ 3
$ns attach-agent $n3 $tcp3

#create a FTP source & attach it to TCP3 i.e node n3
set ftp3 [new Application/FTP]
$ftp3 set packetSize_ 500
$ftp3 set interval_ 0.005
$ftp3 attach-agent $tcp3

#Create a UDP SINK agent (a traffic sink) & attach it to node n3
set udp_sink3 [new Agent/Null]
$ns attach-agent $n3 $udp_sink3
#Create a TCP SINK agent (a traffic sink) & attach it to node n3
set tcp_sink3 [new Agent/TCPSink]
$ns attach-agent $n3 $tcp_sink3

##################################Node n4 ########################

#create a UDP agent and attach it to node n4
set udp4 [new Agent/UDP]
$udp4 set fid_ 4
$ns attach-agent $n4 $udp4

#create a CBR traffic source & attach it to UDP4 i.e node n4
set cbr4 [new Application/Traffic/CBR]
$cbr4 set packetSize_ 500
$cbr4 set interval_ 0.005
$cbr4 attach-agent $udp4

#create a TCP agent and attach it to node n4
set tcp4 [new Agent/TCP]
$tcp4 set fid_ 4
$ns attach-agent $n4 $tcp4

#create a FTP source & attach it to TCP4 i.e node n4
set ftp4 [new Application/FTP]
$ftp4 set packetSize_ 500
$ftp4 set interval_ 0.005
$ftp4 attach-agent $tcp4

#Create a UDP SINK agent (a traffic sink) & attach it to node n4
set udp_sink4 [new Agent/Null]
$ns attach-agent $n4 $udp_sink4
#Create a TCP SINK agent (a traffic sink) & attach it to node n4
set tcp_sink4 [new Agent/TCPSink]
$ns attach-agent $n4 $tcp_sink4

########################################Node n5 ##################
#create a UDP agent and attach it to node n5
set udp5 [new Agent/UDP]
$udp5 set fid_ 5
$ns attach-agent $n5 $udp5

#create a CBR traffic source & attach it to UDP5 i.e node n5
set cbr5 [new Application/Traffic/CBR]
$cbr5 set packetSize_ 500
$cbr5 set interval_ 0.005
```

```
$cbr5 attach-agent $udp5

#create a TCP agent and attach it to node n5
set tcp5 [new Agent/TCP]
$tcp5 set fid_ 5
$ns attach-agent $n5 $tcp5

#create a FTP source & attach it to TCP5 i.e node n5
set ftp5 [new Application/FTP]
$ftp5 set packetSize_ 500
$ftp5 set interval_ 0.005
$ftp5 attach-agent $tcp5

#Create a UDP SINK agent (a traffic sink) & attach it to node n5
set udp_sink5 [new Agent/Null]
$ns attach-agent $n5 $udp_sink5
#Create a TCP SINK agent (a traffic sink) & attach it to node n5
set tcp_sink5 [new Agent/TCPSink]
$ns attach-agent $n5 $tcp_sink5

####################################Node n6#########################

#create a UDP agent and attach it to node n6
set udp6 [new Agent/UDP]
$udp6 set fid_ 6
$ns attach-agent $n6 $udp6

#create a CBR traffic source & attach it to UDP6 i.e node n6
set cbr6 [new Application/Traffic/CBR]
$cbr6 set packetSize_ 500
$cbr6 set interval_ 0.005
$cbr6 attach-agent $udp6

#create a TCP agent and attach it to node n6
set tcp6 [new Agent/TCP]
$tcp6 set fid_ 6
$ns attach-agent $n6 $tcp6

#create a FTP source & attach it to TCP6 i.e node n6
set ftp6 [new Application/FTP]
$ftp6 set packetSize_ 500
$ftp6 set interval_ 0.005
$ftp6 attach-agent $tcp6

#Create a UDP SINK agent (a traffic sink) & attach it to node n6
set udp_sink6 [new Agent/Null]
$ns attach-agent $n6 $udp_sink6
#Create a TCP SINK agent (a traffic sink) & attach it to node n6
set tcp_sink6 [new Agent/TCPSink]
$ns attach-agent $n6 $tcp_sink6

#########################################Node n7 ##################
#create a UDP agent and attach it to node n7
set udp7 [new Agent/UDP]
$udp7 set fid_ 7
$ns attach-agent $n7 $udp7

#create a CBR traffic source & attach it to UDP7 i.e node n7
set cbr7 [new Application/Traffic/CBR]
$cbr7 set packetSize_ 500
$cbr7 set interval_ 0.005
$cbr7 attach-agent $udp7

#create a TCP agent and attach it to node n7
```

```
set tcp7 [new Agent/TCP]
$tcp7 set fid_ 7
$ns attach-agent $n7 $tcp7

#create a FTP source & attach it to TCP7 i.e node n7
set ftp7 [new Application/FTP]
$ftp7 set packetSize_ 500
$ftp7 set interval_ 0.005
$ftp7 attach-agent $tcp7

#Create a UDP SINK agent (a traffic sink) & attach it to node n7
set udp_sink7 [new Agent/Null]
$ns attach-agent $n7 $udp_sink7
#Create a TCP SINK agent (a traffic sink) & attach it to node n7
set tcp_sink7 [new Agent/TCPSink]
$ns attach-agent $n7 $tcp_sink7
#####################################################################

#connect the traffic sources with the traffic sink
$ns connect $tcp0 $tcp_sink4
$ns connect $udp0 $udp_sink5

$ns connect $tcp1 $tcp_sink5
$ns connect $udp1 $udp_sink6

$ns connect $tcp2 $tcp_sink6
$ns connect $udp2 $udp_sink7

$ns connect $tcp3 $tcp_sink7
$ns connect $udp3 $udp_sink4

$ns connect $tcp4 $tcp_sink0
$ns connect $udp4 $udp_sink1

$ns connect $tcp5 $tcp_sink1
$ns connect $udp5 $udp_sink2

$ns connect $tcp6 $tcp_sink2
$ns connect $udp6 $udp_sink3

$ns connect $tcp7 $tcp_sink3
$ns connect $udp7 $udp_sink0

#Schedule events for the FTP agents. Simulation time are given in
seconds
$ns at 0.5 "$ftp0 start"
$ns at 0.5 "$cbr0 start"
$ns at 0.5 "$ftp1 start"
$ns at 0.5 "$cbr1 start"
$ns at 0.5 "$ftp2 start"
$ns at 0.5 "$cbr2 start"
$ns at 0.5 "$ftp3 start"
$ns at 0.5 "$cbr3 start"

$ns at 0.5 "$ftp4 start"
$ns at 0.5 "$cbr4 start"
$ns at 0.5 "$ftp5 start"
$ns at 0.5 "$cbr5 start"
$ns at 0.5 "$ftp6 start"
$ns at 0.5 "$cbr6 start"
$ns at 0.5 "$ftp7 start"
$ns at 0.5 "$cbr7 start"

$ns at 4.5 "$ftp0 stop"
```

```
$ns at 4.5 "$cbr0 stop"
$ns at 4.5 "$ftp1 stop"
$ns at 4.5 "$cbr1 stop"
$ns at 4.5 "$ftp2 stop"
$ns at 4.5 "$cbr2 stop"
$ns at 4.5 "$ftp3 stop"
$ns at 4.5 "$cbr3 stop"

$ns at 4.5 "$ftp4 stop"
$ns at 4.5 "$cbr4 stop"
$ns at 4.5 "$ftp5 stop"
$ns at 4.5 "$cbr5 stop"
$ns at 4.5 "$ftp6 stop"
$ns at 4.5 "$cbr6 stop"
$ns at 4.5 "$ftp7 stop"
$ns at 4.5 "$cbr7 stop"

#Call the finish procedure after 5 seconds of simulation time
#finish procedure closes the nf which is file handler of NAM trace
file and opens the network animator
$ns at 5.0 "finish"

#Run the simulation
$ns run
```

Simulation using AODV routing protocol:

```
# A 3-node example for ad-hoc simulation with AODV
# Define options
set val(chan)           Channel/WirelessChannel    ;# channel type
set val(prop)           Propagation/TwoRayGround   ;# radio-
propagation model
set val(netif)          Phy/WirelessPhy            ;# network
interface type
set val(mac)            Mac/802_11                 ;# MAC type
set val(ifq)            Queue/DropTail             ;# 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)             3                          ;# number of
mobilenodes
set val(rp)             AODV                       ;# routing
protocol
set val(x)              500                ;# X dimension of
topography
set val(y)              400                ;# Y dimension of
topography
set val(stop)         150                  ;# time of simulation end

set ns             [new Simulator]
set tracefd       [open simple-dsdv.tr w]
set windowVsTime2 [open win.tr w]
set namtrace      [open simwrls1.nam w]

$ns trace-all $tracefd
$ns use-newtrace
$ns namtrace-all-wireless $namtrace $val(x) $val(y)

# set up topography object
set topo       [new Topography]

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

create-god $val(nn)

#
#  Create nn mobilenodes [$val(nn)] and attach them to the channel.
#

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

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

```
            set node_($i) [$ns node]
      }

# Provide initial location of mobilenodes
$node_(0) set X_ 5.0
$node_(0) set Y_ 5.0
$node_(0) set Z_ 0.0

$node_(1) set X_ 490.0
$node_(1) set Y_ 285.0
$node_(1) set Z_ 0.0

$node_(2) set X_ 150.0
$node_(2) set Y_ 240.0
$node_(2) set Z_ 0.0

# Generation of movements
$ns at 10.0 "$node_(0) setdest 250.0 250.0 3.0"
$ns at 20.0 "$node_(1) setdest 45.0 285.0 5.0"
$ns at 30.0 "$node_(0) setdest 480.0 300.0 5.0"

# Set a TCP connection between node_(0) and node_(1)
set tcp [new Agent/TCP/Newreno]
$tcp set class_ 2
set sink [new Agent/TCPSink]
$ns attach-agent $node_(0) $tcp
$ns attach-agent $node_(1) $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 10.0 "$ftp start"



# Define node initial position in nam
for {set i 0} {$i < $val(nn)} { incr i } {
# 30 defines the node size for nam
$ns initial_node_pos $node_($i) 30
}

# Telling nodes when the simulation ends
for {set i 0} {$i < $val(nn) } { incr i } {
    $ns at $val(stop) "$node_($i) reset";
}

# ending nam and the simulation
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "stop"
$ns at 1000.01 "puts \"end simulation\" ; $ns halt"
proc stop {} {
    global ns tracefd namtrace
    $ns flush-trace
    close $tracefd
    close $namtrace
    exec nam simwrls1.nam &
}

$ns run
```

Simulation using DSDV routing protocol:

```
# A 3-node example for ad-hoc simulation with DSDV

# Define options
set val(chan)           Channel/WirelessChannel    ;# channel type
set val(prop)           Propagation/TwoRayGround   ;# radio-
propagation model
set val(netif)          Phy/WirelessPhy            ;# network
interface type
set val(mac)            Mac/802_11                 ;# MAC type
set val(ifq)            Queue/DropTail    ;# 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)             3                          ;# number of
mobilenodes
set val(rp)             DSDV                       ;# routing
protocol
set val(x)              500                   ;# X dimension of
topography
set val(y)              400                   ;# Y dimension of
topography
set val(stop)         150                 ;# time of simulation end

set ns             [new Simulator]
set tracefd       [open simple-dsdv.tr w]
set windowVsTime2 [open win.tr w]
set namtrace      [open simwrls.nam w]

$ns trace-all $tracefd
$ns use-newtrace
$ns namtrace-all-wireless $namtrace $val(x) $val(y)

# set up topography object
set topo       [new Topography]

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

create-god $val(nn)

#
#  Create nn mobilenodes [$val(nn)] and attach them to the channel.
#

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

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

```
            set node_($i) [$ns node]
      }

# Provide initial location of mobilenodes
$node_(0) set X_ 5.0
$node_(0) set Y_ 5.0
$node_(0) set Z_ 0.0

$node_(1) set X_ 490.0
$node_(1) set Y_ 285.0
$node_(1) set Z_ 0.0

$node_(2) set X_ 150.0
$node_(2) set Y_ 240.0
$node_(2) set Z_ 0.0

# Generation of movements
$ns at 10.0 "$node_(0) setdest 250.0 250.0 3.0"
$ns at 20.0 "$node_(1) setdest 45.0 285.0 5.0"
$ns at 30.0 "$node_(0) setdest 480.0 300.0 5.0"

# Set a TCP connection between node_(0) and node_(1)
set tcp [new Agent/TCP/Newreno]
$tcp set class_ 2
set sink [new Agent/TCPSink]
$ns attach-agent $node_(0) $tcp
$ns attach-agent $node_(1) $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 10.0 "$ftp start"



# Define node initial position in nam
for {set i 0} {$i < $val(nn)} { incr i } {
# 30 defines the node size for nam
$ns initial_node_pos $node_($i) 30
}

# Telling nodes when the simulation ends
for {set i 0} {$i < $val(nn) } { incr i } {
    $ns at $val(stop) "$node_($i) reset";
}

# ending nam and the simulation
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "stop"
$ns at 150.01 "puts \"end simulation\" ; $ns halt"
proc stop {} {
    global ns tracefd namtrace
    $ns flush-trace
    close $tracefd
    close $namtrace
    exec nam simwrls.nam &
}

$ns run
```