



**GURU GOBIND SINGH FOUNDATION'S  
GURU GOBIND SINGH COLLEGE OF ENGINEERING AND  
RESEARCH CENTRE, NASHIK**

# *Certificate*

**DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**

*This is certify that Mr./Miss \_\_\_\_\_*

*of Second /Third Year Artificial Intelligence and Data Science : \_\_\_\_\_ Roll No.:*

*-----University Examination Seat No. : \_\_\_\_\_ has*

*completed all the practical work/Term work in Subject \_\_\_\_\_ satisfactorily in Department of*

*Artificial Intelligence and Data Science as prescribed by the*

*Savitribai Phule Pune University in the academic year 20 \_\_\_\_\_ 20*

**Practical In-charge**

**Head of the Department**

**Principal**



**GURU GOBIND SINGH FOUNDATION'S  
GURU GOBIND SINGH COLLEGE OF ENGINEERING AND  
RESEARCH CENTRE, NASHIK**

**DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**

**INDEX**

**Name of Subject:**

<b>Expt No.</b>	<b>Name of Experiment</b>	<b>Date of Performance</b>	<b>Date of Completion</b>	<b>Grade/Marks</b>	<b>Sign</b>
1	Demonstrate the different types of topologies and types of transmission media by using a packet tracer tool.				
2	Setup a wired LAN using Layer 2 Switch. It includes preparation of cable, testing of cable using line tester, configuration machine using IP addresses, testing using PING utility and demonstrating the PING packets captured traces using Wireshark Packet Analyzer Tool.				
3	Write a program to demonstrate Sub-netting and find subnet masks.				
4	Write a program to implement link state /Distance vector routing protocol to find a suitable path for transmission.				
5	Write a program using TCP socket for wired network for following a. Say Hello to Each other b. File transfer				

<b>Expt No.</b>	<b>Name of Experiment</b>	<b>Date of Performance</b>	<b>Date of Completion</b>	<b>Grade/Marks</b>	<b>Sign</b>
6	Write a program using UDP Sockets to enable file transfer (Script, Text, Audio and Video one file each) between two machines.				
7	Capture packets using Wireshark and accomplish the following and save the output in file:				
8	Study and Analyze the performance of HTTP, HTTPS and FTP protocol using Packet tracer tool.				
9	Illustrate the steps for implementation of S/MIME email security, POP3 through Microsoft® Office Outlook.				
10	To study the IPsec (ESP and AH) protocol by capturing the packets using Wireshark tool.				
11	Installing and configuring DHCP server and assign IP addresses to client machines using DHCP server.				
12	Write a program for DNS lookup. Given an IP address input, it should return URL and vice versa.				

**Date of Submission:**

**Practical In-charge**

# Practical No. 1

**Title:** Demonstrate the different types of topologies and types of transmission media by using a packet tracer tool.

**Objectives:**

1. To understand the structure and working of various networks network topologies.

**Problem Statement:** Demonstrate the different types of topologies and types of transmission media by using a packet tracer tool.

**Theory:**

## NETWORK TOPOLOGY

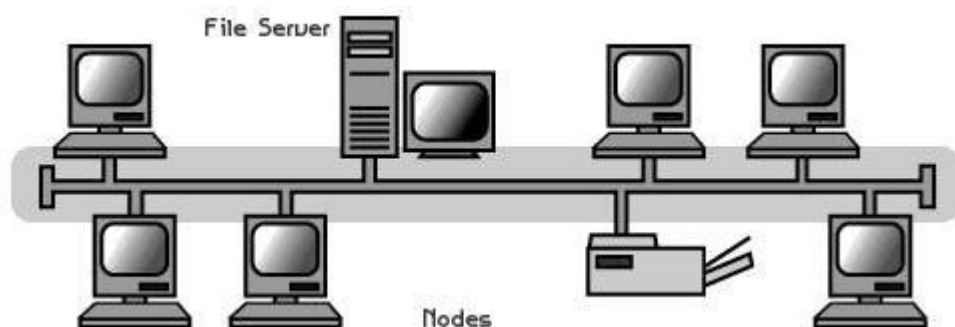
The study of network topology recognizes seven basic topologies: [3]

- Point-to-point topology
- Bus (point-to-multipoint) topology
- Star topology
- Ring topology
- Tree topology
- Mesh topology
- Hybrid topology

This classification is based on the interconnection between computers — be it physical or logical.

The physical topology of a network is determined by the capabilities of the network access devices and media, the level of control or fault tolerance desired, and the cost associated with cabling or telecommunications circuits.

## Bus



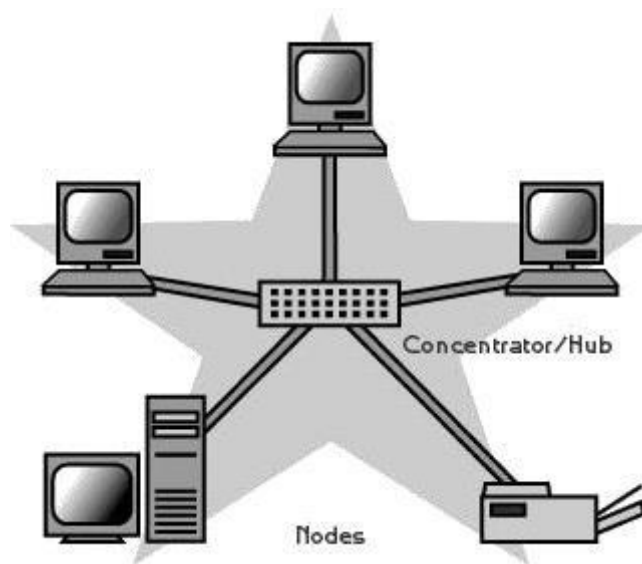
## Bus network topology

In local area networks where bus topology is used, each machine is connected to a single cable. Each computer or server is connected to the single bus cable through some kind of connector. A terminator is required at each end of the bus cable to prevent the signal from bouncing back and forth on the bus cable. A signal from the source travels in both directions to all machines connected on the bus cable until it finds the MAC address or IP address on the network that is the intended recipient. If the machine address does not match the intended address for the data, the machine ignores the data. Alternatively, if the data does match the machine address, the data is accepted. Since the bus topology consists of only one wire, it is rather inexpensive to implement when compared to other topologies. However, the low cost of implementing the technology is offset by the high cost of managing the network. Additionally, since only one cable is utilized, it can be the single point of failure. If the network cable breaks, the entire network will be down.

## Star

A star topology is designed with each node (file server, workstations, and peripherals) connected directly to a central network hub, switch, or concentrator (See fig. 2).

Data on a star network passes through the hub, switch, or concentrator before continuing to its destination. The hub, switch, or concentrator manages and controls all functions of the network. It also acts as a repeater for the data flow. This configuration is common with twisted pair cable; however, it can also be used with coaxial cable or fiber optic cable.



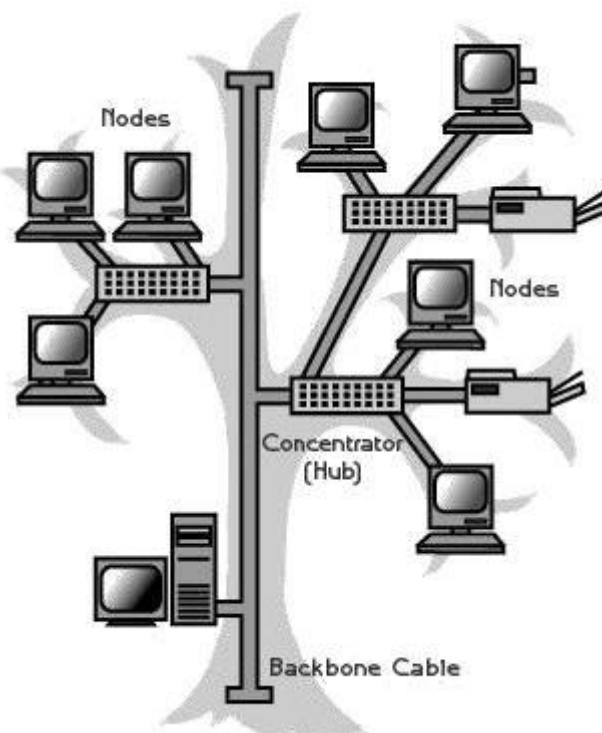
### Advantages of a Star Topology

- ❑ Easy to install and wire.
- ❑ No disruptions to the network when connecting or removing devices.
- ❑ Easy to detect faults and to remove parts.

### Disadvantages of a Star Topology

- ❑ Requires more cable length than a linear topology.
- ❑ If the hub, switch, or concentrator fails, nodes attached are disabled.
- ❑ More expensive than linear bus topologies because of the cost of the hubs, etc.

**Tree or Expanded Star:** A tree topology combines characteristics of linear bus and star topologies. It consists of groups of star-configured workstations connected to a linear bus backbone cable (See fig.). Tree topologies allow for the expansion of an existing network, and enable schools to configure a network to meet their needs.



## Advantages of a Tree Topology

- ❑ Point-to-point wiring for individual segments.
- ❑ Supported by several hardware and software vendors.

## Disadvantages of a Tree Topology

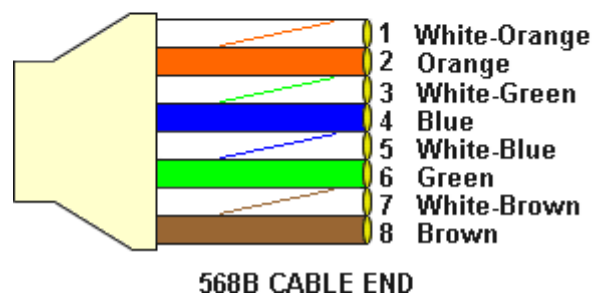
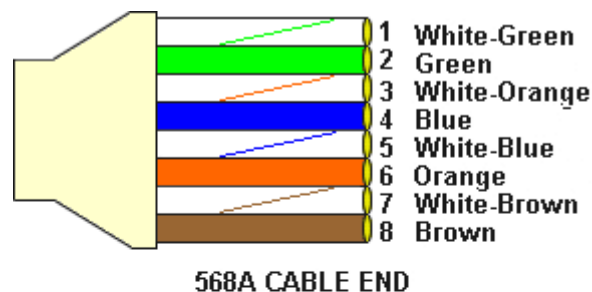
- ❑ Overall length of each segment is limited by the type of cabling used.
- ❑ If the backbone line breaks, the entire segment goes down.
- ❑ More difficult to configure and wire than other topologies.

## PROCEDURE

There are generally three main types of networking cables: straight-through, crossover, and rollover cables. Each cable type has a distinct use, and should not be used in place of another. So how do you know which cable to use for what you need?

### The Purpose of Straight-Through Cables

Straight-through cables get their name from how they are made. Out of the 8 pins that exist on both ends of an Ethernet cable, each pin connects to the same pin on the opposite side. Review the diagram below for a visual example:



Notice how each wire corresponds to the same pin. This kind of wiring diagram is part of the 568A standard. The 568B standard achieves the same thing, but through different wiring. It is generally accepted to use the 568A standard as pictured, since it allows compatibility with certain telephone hardware- while 568B doesn't.

Straight-through cables are primarily used for connecting unlike devices. A straight-through cable is typically used in the following situations:

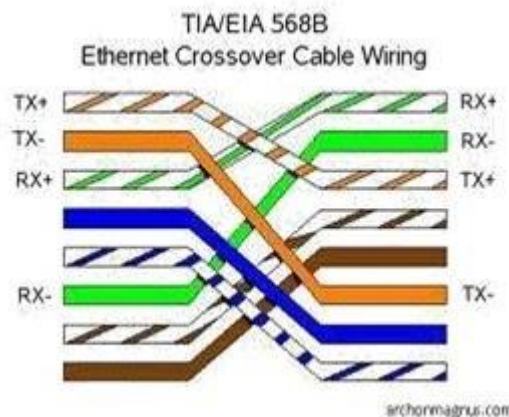
**Use a straight-through cable when:**

- Connecting a router to a hub
- Connecting a computer to a switch
- Connecting a LAN port to a switch, hub, or computer

Note that some devices such as routers will have advanced circuitry, which enables them to use both crossover and straight-through cables. In general, however, straight-through cables will not connect a **computer and router because they are not “unlike devices.”**

**The purpose of Crossover Cables**

Crossover cables are very similar to straight-through cables, except that they have pairs of wires that crisscross. This allows for two devices to communicate at the same time. Unlike straight-through cables, we use crossover cables to connect like devices. A visual example can be seen below:



Notice how all we did was switch the orange-white and green-white wires, and then the orange and green wires. This will enable like devices to communicate. Crossover cables are typically used in the following situations:

**Use a crossover cable when:**

- ❑ Connecting a computer to a router
- ❑ Connecting a computer to a computer
- ❑ Connecting a router to a router
- ❑ Connecting a switch to a switch
- ❑ Connecting a hub to a hub

While the rule of thumb is to use crossover cables with like devices, some devices do not follow standards. Others provide support for both types of cables. However, there is still something that both crossover and straight-through cables can't do.

**Conclusion: We have studied different types of topologies to form network and demonstrated these topology in packet tracer.**

## Practical No. 2

### Title: Study of Existing LAN

#### Objectives:

1. To understand the structure and working of various networks including the interconnecting devices used in them.
2. To get hands on experience of making and testing cables.

**Problem Statement:** Setup a wired LAN using Layer 2 Switch and then IP switch of minimum four computers. It includes preparation of cable, testing of cable using line tester, configuration machine using IP addresses, testing using PING utility and demonstrate the PING packets captured traces using Wireshark Packet Analyzer Tool.

#### Theory:

##### LAN - Local Area Network

A LAN connects network devices over a relatively short distance. A networked office building, school, or home usually contains a single LAN, though sometimes one building will contain a few small LANs (perhaps one per room), and occasionally a LAN will span a group of nearby buildings.

##### MAN-Metropolitan Area Network

A network spanning a physical area larger than a LAN but smaller than a WAN, such as a city. A MAN is typically owned and operated by a single entity such as a government body or large corporation.

##### WAN:

A **wide area network (WAN)** is a telecommunications network or computer network that extends over a large geographical distance. Wide area networks are often established with leased telecommunication circuits. Business, education and government entities use wide area networks to relay data to staff, students, clients, buyers, and suppliers from various locations across the world. In essence, this mode of telecommunication allows a business to effectively carry out its daily function regardless of location. The Internet may be considered a WAN

#### Network Devices:

##### Hubs

Hub is one of the basic icons of networking devices which works at physical layer and hence connect networking devices physically together. Hubs are fundamentally used in networks that use **twisted pair cabling** to connect devices.

They are designed to transmit the packets to the other appended devices without altering any of the transmitted packets received. They act as pathways to direct electrical signals to travel along.

They transmit the information regardless of the fact if data packet is destined for the device

connected or not.

Hub falls in two categories:

**Active Hub:** They are smarter than the passive hubs. They not only provide the path for the data signals infact they regenerate, concentrate and strengthen the signals before sending them to their destinations. Active hubs are also termed as '**repeaters**'.

**Passive Hub:** They are more like point contact for the wires to built in the physical network. They have nothing to do with modifying the signals.



## Switches

Switches are the linkage points of an Ethernet network. Just as in hub, devices in switches are connected to them through twisted pair cabling. But the difference shows up in the manner both the devices; hub and a switch treat the data they receive.

**Hub** works by sending the data to all the ports on the device whereas a **switch** transfers it only to that port which is connected to the destination device. A switch does so by havingan in-built learning of the MAC address of the devices connected to it.

Since the transmission of data signals are well defined in a **switch** hence the network performance is consequently enhanced. Switches operate in **full-duplex** mode where devices can send and receive data from the switch at the simultaneously unlike in half-duplex mode.

The transmission speed in switches is double than in Ethernet hub transferring a 20Mbps connection into 30Mbps and a 200Mbps connection to become 300Mbps. Performance improvements are observed in networking with the extensive usage of switches in the modern days.

The following method will elucidate further how data transmission takes place via switches:

- **Cut-through transmission:** It allows the packets to be forwarded as soon as they are received. The method is prompt and quick but the possibility of error checking gets overlooked in such kind of packet data transmission.
- **Store and forward:** In this switching environment the entire packet are received and 'checked' before being forwarded ahead. The errors are thus eliminated before being propagated further. The downside of this process is that error checking takes relatively longer time consequently making it a bit slower in processing and delivering.
- **Fragment Free:** In a fragment free switching environment, a greater part of the packet is examined so that the switch can determine whether the packet has been caught up in a collision. After the collision status is determined, the packet is forwarded.

## Bridges

A bridge is a computer networking device that builds the connection with the other bridge networks which use the same protocol. It works at the Data Link layer of the OSI Model and connects the different networks together and develops communication between them.

It connects two local-area networks; two physical LANs into larger logical LAN or two *segments* of the same LAN that use the same protocol.

Apart from building up larger networks, bridges are also used to segment larger networks into *smaller* portions.

## PING Command:

**ping** is a computer network administration software utility used to test the reachability of a host on an Internet Protocol (IP) network. It measures the round-trip time for messages sent from the originating host to a destination computer that are echoed back to the source.

The name comes from active sonar terminology that sends a pulse of sound and listens for the echo to detect objects under water, although it is sometimes interpreted as a backronym to *packet Internet groper*. Ping operates by sending Internet Control Message Protocol (ICMP/ICMP6) Echo Request packets to the target host and waiting for an ICMP Echo Reply.

The program reports errors, packet loss, and a statistical summary of the results, typically including the minimum, maximum, the mean round-trip times, and standard deviation of the mean. The command-line options of the ping utility and its output vary between the numerous implementations.

### **APPLICATION**

Most networks in the real world use the above-mentioned devices to create networks.

**Conclusion:** Thus we have studied wired LAN setup and connection.

## Practical No. 3

### PROBLEM STATEMENT:

**Write a program to demonstrate Sub-netting and find subnet masks.**

### Prerequisite:

1. IP Address Classes
2. Classless & Classful IP Addressing

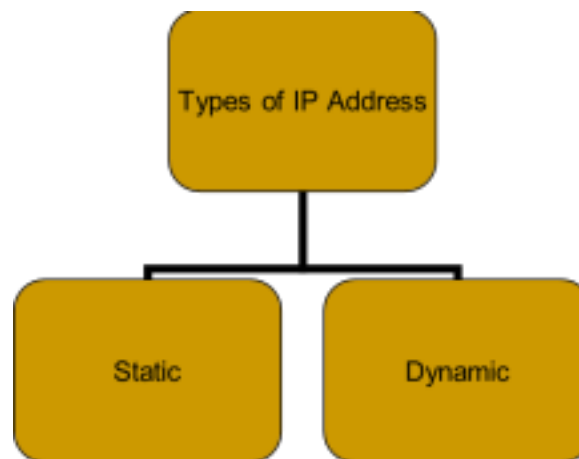
### Learning Objectives:

1. Understand the concept Subnetting.
2. Understand the Concept of Supernet.

### Theory

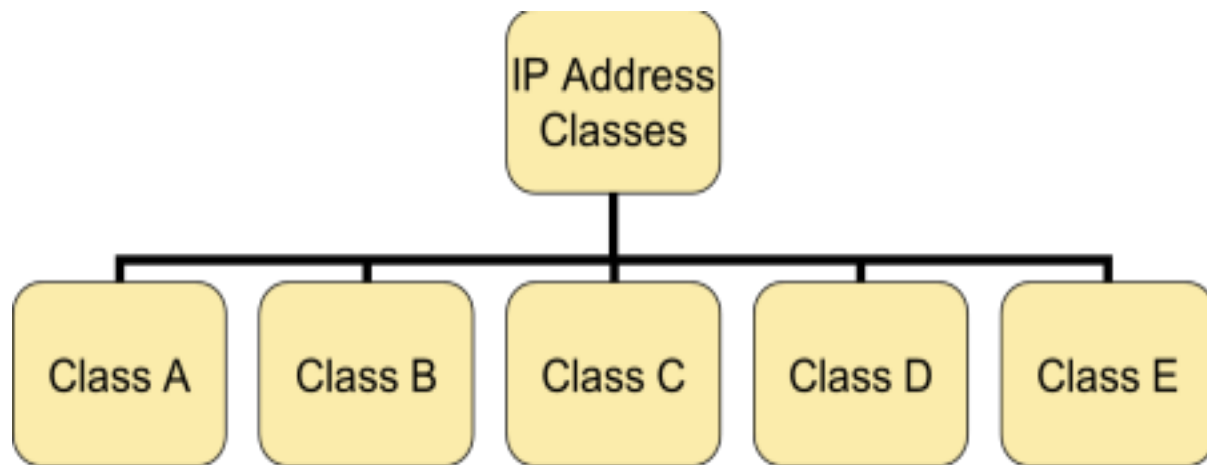
#### Introduction

- A Unique, 32-bit address used by computers to communicate over a computer network



#### Classes of Address

- IP address structure consists of two addresses, Network and Host
- IP address is divided into five classes



## IP Address Classes

The 5 IP classes are split up based on the value in the 1<sup>st</sup> octet: Class

A: 0-127

Class B: 128-191

Class C: 192-223

Class D: 224-239

Class E: 240-255

### IP Address Classes(Cont.)

	Byte 1	Byte 2	Byte 3	Byte 4
Class A	Network ID	Host ID		
Class B	Network ID		Host ID	
Class C	Network ID			Host ID
Class D	Multicast Address			
Class E	Reserved for future use			

## Examples of IP Address

- 14.23.120.8 - The first byte of the address represents 14 which lies between 0 and 127, hence Class A address.
- 134.11.78.56 - The first byte of address is 134 which lies between 128 and 191 hence the address belongs to Class B.
- 193.14.56.22 - As first byte is 193 which is between 192 and 223, hence the address belongs to Class C.

## Subnet Mask

An IP address has 2 parts:

- The Network identification.
- The Host identification.

Frequently, the Network & Host portions of the address need to be separately extracted.

- In most cases, if you know the address class, it's easy to separate the 2 portions.
- Specifies part of IP address used to identify a subnetwork.
- Subnet mask when logically ANDed with IP address provides 32-bit network address

## Default Mask:

- Has predetermined number of 1s
- Class A, B and C contains 1s in network ID fields for default subnet mask.

Address Class	Default Mask (in Binary)
Class A	11111111.00000000.00000000.00000000
Class B	11111111.11111111.00000000.00000000
Class C	11111111.11111111.11111111.00000000

## Default Standard Subnet Masks

There are default standard subnet masks for Class A, B and C addresses:

Default Subnet Masks	
<i>Address Class</i>	<i>Subnet Mask</i>
Class A	255.0.0.0
Class B	255.255.0.0
Class C	255.255.255.0

### IP Subnetting:

- Allows you to divide a network into smaller sub-networks
- Each subnet has its own sub-network address
- Subnet can be created within Class A, B, or C based networks

### Subnetting:

- Division of a network into subnets
  - For example, division of a Class B address into several Class C addresses
  - Some of the host IDs are used for creating subnet IDs

### Need for Subnetting:

- Classes A and B have a large number of hosts corresponding to each network ID
  - It may be desirable to subdivide the hosts in Class C subnets
- Often, there is a limitation on the number of hosts that could be hosted on a single network segment
  - The limitation may be imposed by concerns related to the management of hardware
- Smaller broadcast domains are more efficient and easy to manage

## Subnetting Principle:

- Use parts of the host IDs for subnetting purpose
- A subnet mask is used to facilitate the flow of traffic between the different subnets and the outside network (hops)
- A hop is the distance a data packet travels from one node to the other .

## How to Calculate Subnets:

- To determine the number of subnets & hosts per subnet available for any of the available subnet masks, 2 simple formulas to calculate these numbers:
- Number of Subnets= $(2^n)$
- Number of Host per Subnets= $(2^{h-2})$
- Although the 2 formulas look identical, the key is to remember the number you're trying to calculate, hosts or subnets.
- Eg., suppose you are asked to determine the number of subnets available & the number of hosts available on each subnet on the network 192.168.1.0
- Using the subnet & hosts formulas, the answers are easily calculated. Of course, you must know your powers of 2 to calculate the answers

## Example:

- Host IP Address: 138.101.114.250
- Network Mask: 255.255.0.0 (or /16)
- Subnet Mask: 255.255.255.192 (or /26)

Given the following Host IP Address, Network Mask and Subnet mask find the Major Network Address, Network Broadcast Address, Range of Host if not subnetted, Subnet Address, range of host (First address and last address) ,Broadcast address,Total no of subnets and number of hosts per subnet.

### Major Informations:

- **Host IP Address:** 138.101.114.250
- **Network Mask:** 255.255.0.0
- **Subnet Mask:** 255.255.255.192
- **Major Network Address:** 138.101.0.0
- **Major Network Broadcast Address:** 138.101.255.255
- **Range of Hosts if not Subnetted:** 138.101.0.1 to 138.101.255.254

## Step 1: Convert to Binary

**138. 101. 114. 250**

**IP Address** 10001010 01100101 01110010 11111010 **Mask** 11111111 11111111  
11111111 11000000 **255. 255. 255. 192**

## Step 2: Find the Subnet Address

**138. 101. 114. 250**

**IP Address** 10001010 01100101 01110010 11111010 **Mask** 11111111 11111111  
11111111 11000000 **Network** 10001010 01100101 01110010 11000000 **138 101**  
**114 192**

### Determine the Network (or Subnet) where this Host address

**lives:** 1. Draw a line under the mask

2. Perform a bit-wise AND operation on the IP Address and the Subnet Mask

Note: 1 AND 1 results in a 1, 0 AND anything results in a 0

3. Express the result in Dotted Decimal Notation

4. The result is the **Subnet Address** of this Subnet or “Wire” which is 138.101.114.192

**138. 101. 114. 250**

**IP Address** 10001010 01100101 01110010 11111010 **Mask** 11111111 11111111  
11111111 11000000 **Network** 10001010 01100101 01110010 11000000 **138 101**  
**114 192**

Quick method:

1. Find the last (right-most) 1 bit in the subnet mask.
2. Copy all of the bits in the IP address to the Network Address
3. Add 0's for the rest of the bits in the Network Address

### Step 3: Subnet Range / Host Range

**IP Address** 10001010 01100101 01110010 11 111010 **Mask** 11111111 11111111  
11111111 11 000000 **Network** 10001010 01100101 01110010 11 000000 **subnet**  
**host**  
counting range counting  
range

**Determine which bits in the address contain Network (subnet) information and which contain Host information:**

- Use the **Network Mask**: 255.255.0.0 and divide (**Great Divide**) the from the rest of the address.
- Use **Subnet Mask**: 255.255.255.192 and divide (**Small Divide**) the subnet from the hosts between the last "1" and the first "0" in the subnet mask.

### Step 4: First Host / Last Host

**IP Address** 10001010 01100101 01110010 11 111010 **Mask** 11111111 11111111  
11111111 11 000000 **Network** 10001010 01100101 01110010 11 000000 **subnet**  
**host**  
counting range counting  
range

**First Host** 10001010 01100101 01110010 11 000001 138 101 114 193

**Last Host** 10001010 01100101 01110010 11 111110 138 101 114 254

**Broadcast** 10001010 01100101 01110010 11 111111 138 101 114 255

## Host Portion

- **Subnet Address:** all 0's
- **First Host:** all 0's and a 1
- **Last Host:** all 1's and a 0
- **Broadcast:** all 1's

## Step 5: Total Number of Subnets

- Total number of subnets
  - ☐ Number of subnet bits 10
  - ☐  $2^{10} = 1,024$
  - ☐ 1,024 total subnets
    - Subtract one "if" all-zeros subnet cannot be used
    - Subtract one "if" all-ones subnet cannot be used

## Step 6: Total Number of Hosts per Subnet

**IP Address** 10001010 01100101 01110010 11 111010

**Mask** 11111111 11111111 11111111 11 000000

**Network** 10001010 01100101 01110010 11 000000

### subnet host counting range

- Total number of hosts per subnet
  - ☐ Number of host bits 6
  - ☐  $2^6 = 64$
  - ☐ 64 host per subnets
    - Subtract one for the subnet address
    - Subtract one for the broadcast address
  - ☐ 62 hosts per subnet

## Conclusion:

Hence we have studied Subnetting and the importance of subnetting.

## Practical No. 4

### PROBLEM STATEMENT:

**Write a program to implement link state /Distance vector routing protocol to find suitable path for transmission**

### Prerequisite:

1. Shortest path finding
2. Classification of routing Algorithm

### Learning Objectives:

1. Understand the concept Distance vector routing
2. Understand the Concept of Routing Algorithms

### Theory

A distance-vector routing (DVR) protocol requires that a router inform its neighbors of topology changes periodically. Historically known as the old ARPANET routing algorithm (or known as Bellman-Ford algorithm).

Bellman Ford Basics – Each router maintains a Distance Vector table containing the distance between itself and ALL possible destination nodes. Distances, based on a chosen metric, are computed using information from the neighbors' distance vectors.

Information kept by DV router -

- Each router has an ID

Associated with each link connected to a router,

- there is a link cost (static or dynamic).
- Intermediate hops

Distance Vector Table Initialization -

- Distance to itself = 0
- Distance to ALL other routers = infinity number.

## Distance Vector Algorithm –

1. A router transmits its distance vector to each of its neighbors in a routing packet.
2. Each router receives and saves the most recently received distance vector from each of its neighbors.
3. A router recalculates its distance vector when:
  - It receives a distance vector from a neighbor containing different information than before.
  - It discovers that a link to a neighbor has gone down.

The DV calculation is based on minimizing the cost to each destination

$D_x(y)$  = Estimate of least cost from x to y

$C(x,v)$  = Node x knows cost to each neighbor v

$D_x = [D_x(y): y \in N]$  = Node x maintains distance vector

Node x also maintains its neighbors' distance vectors

– For each neighbor v, x maintains  $D_v = [D_v(y): y \in N]$

### 7.4.1 Introduction

Distance Vector Routing –

- It is a dynamic routing algorithm in which each router computes distance between itself and each possible destination i.e. its immediate neighbors.
- The router share its knowledge about the whole network to its neighbors and

accordingly updates table based on its neighbors.

- The sharing of information with the neighbors takes place at regular intervals.
- It makes use of Bellman Ford Algorithm for making routing tables.
- Problems – Count to infinity problem which can be solved by splitting horizon.
  - Good news spread fast and bad news spread slowly.
  - Persistent looping problem i.e. loop will be there forever.

#### Link State Routing –

- It is a dynamic routing algorithm in which each router shares knowledge of its neighbors with every other router in the network.
- A router sends its information about its neighbors only to all the routers through flooding.
- Information sharing takes place only whenever there is a change.
- It makes use of Dijkstra's Algorithm for making routing tables.
- Problems – Heavy traffic due to flooding of packets.
  - Flooding can result in infinite looping which can be solved by using Time to live (TTL) field.

Distance Vector Routing	Link State Routing
--> Bandwidth required is less due to local sharing, small packets and no flooding.	--> Bandwidth required is more due to flooding and sending of large link state packets.
--> Based on local knowledge since it updates table based on information from neighbors.	--> Based on global knowledge i.e. it have knowledge about entire network.
--> Make use of Bellman Ford algo	--> Make use of Dijkstra's algo
--> Traffic is less	--> Traffic is more
--> Converges slowly i.e. good news spread fast and bad news spread slowly.	--> Converges faster.
--> Count to infinity problem.	--> No count to infinity problem.
--> Persistent looping problem i.e. loop will there forever.	--> No persistent loops, only transient loops.
--> Practical implementation is RIP and IGRP.	--> Practical implementation is OSPF and ISIS.

**Conclusion:** Hence we have studied distance vector algorithm to find suitable path for transmission

## Practical No. 5

### Problem Definition:

Write a program using TCP socket for wired network for following

- a. Say Hello to Eachother
- b. File transfer
- c. Calculator

### Prerequisite:

- a) Socket Header
- b) Network Programming
- c) Ports

### Learning Objectives:

- 1. To understand Work of Socket
- 2. Different methods associated with Client & Server Socket

### New Concepts:

- 1. Client Server Communication
- 2. Port Address

### Theory:

#### Introduction

Theory: Socket Programming: The Berkeley socket interface, an API, allows communications between hosts or between processes on one computer, using the concept of a socket. It can work with many different I/O devices and drivers, although support for these depends on the operating system implementation. This interface implementation is implicit for TCP/IP, and it is therefore one of the fundamental technologies underlying the Internet. It was first developed at the University of California, Berkeley for use on Unix systems. All modern operating systems now have some implementation of the Berkeley socket interface, as it has become the standard interface for connecting to the Internet. Programmers can make the socket interfaces accessible at three different levels, most powerfully and fundamentally at the RAW socket level. Very few applications need the degree of control over outgoing communications that this provides, so RAW sockets support was intended to be available only on computers used for developing Internet related technologies. TCP provides the concept of a connection. A process creates a TCP socket by calling the `socket()` function with the parameters `PF_INET` or `PF_INET6` and `SOCK_STREAM`. Server Setting up a simple TCP server involves the following steps: Creating a TCP socket, with a call to `socket()`. Computer Networks Lab 2 Binding the socket to the listen port, with a call to `bind()`. Before calling `bind()`, a programmer must declare a `sockaddr_in` structure, clear it (with `bzero()`

ormemset()), and the sin\_family (AF\_INET or AF\_INET6), and fill its sin\_port (the listening port, in network byte order) fields. Converting a short int to network byte order can be done by calling the function htons() (host to network short). Preparing the socket to listen for connections (making it a listening socket), with a call to listen(). Accepting incoming connections, via a call to accept(). This blocks until an incoming connection is received, and then returns a socket descriptor for the accepted connection. The initial descriptor remains a listening descriptor, and accept() can be called again at any time with this socket, until it is closed. Communicating with the remote host, which can be done through send() and recv(). Eventually closing each socket that was opened, once it is no longer needed, using close(). Note that if there were any calls to fork(), each process must close the sockets it knew about (the kernel keeps track of how many processes have a descriptor open), and two processes should not use the same socket at once. Client: Setting up a TCP client involves the following steps:

1. Creating a TCP socket, with a call to socket().
2. Connecting to the server with the use of connect, passing a sockaddr\_in structure with the sin\_family set to AF\_INET or AF\_INET6, sin\_port set to the port the end point is listening (in network byte order), and sin\_addr set to the IPv4 or IPv6 address of the listening server (also in network byte order.)
3. Communicating with the server by send()ing and recv()ing. Terminating the connection and cleaning up with a call to close(). Again, if there were any calls to fork(), each process must close() the socket. Functions:
4. socket(): socket() creates an endpoint for communication and returns a descriptor. socket() takes three arguments: domain, which specifies the protocol family of the created socket.

For example: PF\_INET for network protocol IPv4 or PF\_INET6 for IPv6). type, one of: SOCK\_STREAM (reliable stream-oriented service) SOCK\_DGRAM (datagram service) SOCK\_SEQPACKET (reliable sequenced packet service), or SOCK\_RAW (raw protocols atop the network layer). protocol usually set to 0 to represent the default transport protocol for the specified domain and type values (TCP for PF\_INET or PF\_INET6 and SOCK\_STREAM, UDP for those PF\_ values and SOCK\_DGRAM), but which can also

explicitly specify a protocol. The function returns -1 if an error occurred. Otherwise, it returns an integer representing the newly-assigned descriptor. Prototype: `int socket(int domain, int type, int protocol); connect(): connect()` returns an integer representing the error code: 0 represents success, while -1 represents an error. Certain types of sockets are connectionless, most commonly user datagram protocol sockets. For these sockets, connect takes on a special meaning: the default target for sending and receiving data gets set to the given address, allowing the use of functions such as `send()` and `recv()` on connectionless sockets. Prototype: `int connect(int sockfd, const struct sockaddr *serv_addr, socklen_t addrlen); bind(): bind()` assigns a socket an address.

When a socket is created using `socket()`, it is given an address family, but not assigned an address. Before a socket may accept incoming connections, it must be bound. `bind()` takes three arguments: `sockfd`, a descriptor representing the socket to perform the bind on; `my_addr`, a pointer to a `sockaddr` structure representing the address to bind to. `addrlen`, a `socklen_t` field representing the length of the `sockaddr` structure. It returns 0 on success and -1 if an error occurs. Prototype: `int bind(int sockfd, struct sockaddr *my_addr, socklen_t addrlen); listen() listen()` prepares a bound socket to accept incoming connections. This function is only applicable to the `SOCK_STREAM` and `SOCK_SEQPACKET` socket types. It takes two arguments: `sockfd`, a valid socket descriptor.

Computer Networks Lab 4 backlog, an integer representing the number of pending connections that can be queued up at any one time. The operating system usually places a cap on this value. Once a connection is accepted, it is dequeued. On success, 0 is returned. If an error occurs, -1 is returned. Prototype: `int listen(int sockfd, int backlog); accept()` Programmers use `accept()` to accept a connection request from a remote host. It takes the following arguments: `sockfd`, the descriptor of the listening socket to accept the connection from. `cliaddr`, a pointer to the `sockaddr` structure that `accept()` should put the client's address information into. `addrlen`, a pointer to the `socklen_t` integer that will indicate to `accept()` how large the `sockaddr` structure pointed to by `cliaddr` is. When `accept()` returns, the `socklen_t` integer then indicates how many bytes of the `cliaddr` structure were actually used. The function returns a socket corresponding to the accepted connection, or -1 if an error occurs. Prototype: `int accept(int sockfd, struct sockaddr *cliaddr, socklen_t *addrlen); Blocking vs.`

nonblocking Berkeley sockets can operate in one of two modes: blocking or non-blocking. A blocking socket will not "return" until it has sent (or received) all the data specified for the operation. This may cause problems if a socket continues to listen: a program may hang as the socket waits for data that may never arrive. A socket is typically set to blocking or nonblocking mode using the `fcntl()` or `ioctl()` functions. Cleaning up The system will not release the resources allocated by the `socket()` call until a `close()` call occurs. This is especially important if the `connect()` call fails and may be retried. Each call to `socket()` must have a matching call to `close()` in all possible execution paths.

#### Algorithm: Server Program

1. Open the Server Socket: `ServerSocket server = new ServerSocket( PORT );`
2. Wait for the Client Request: `Socket client = server.accept();`
3. Create I/O streams for communicating to the client `InputStream is = new  
InputStream(client.getInputStream());      OutputStream os = new  
OutputStream(client.getOutputStream());`
4. Perform communication with client Receive from client: `String line = is.readLine();` Send to client: `os.writeBytes("Hello\n")`

#### 5. Close socket: `client.close();` Client Program

- 1) Create a Socket Object: `Socket client = new Socket(server, port_id);`
- 2) Create I/O streams for communicating with the server. `is = new  
InputStream(client.getInputStream());      os = new  
OutputStream(client.getOutputStream());`
- 3) Perform I/O or communication with the server: Receive data from the server: `String line = is.readLine();` Send data to the server: `os.writeBytes("Hello\n");`
- 4) Close the socket when done:
- 5) `client.close();`

## TYPES OF SOCKETS

### Socket Types

There are four types of sockets available to the users. The first two are most commonly used and the last two are rarely used.

Processes are presumed to communicate only between sockets of the same type but there is no restriction that prevents communication between sockets of different types.

- **Stream Sockets** – Delivery in a networked environment is guaranteed. If you send through the stream socket three items "A, B, C", they will arrive in the same order – "A, B, C". These sockets use TCP (Transmission Control Protocol) for data transmission. If delivery is impossible, the sender receives an error indicator. Data records do not have any boundaries.
- **Datagram Sockets** – Delivery in a networked environment is not guaranteed. They're connectionless because you don't need to have an open connection as in Stream Sockets – you build a packet with the destination information and send it out. They use UDP (User Datagram Protocol).
- **Raw Sockets** – These provide users access to the underlying communication protocols, which support socket abstractions. These sockets are normally datagram oriented, though their exact characteristics are dependent on the interface provided by the protocol. Raw sockets are not intended for the general user; they have been provided mainly for those interested in developing new communication protocols, or for gaining access to some of the more cryptic facilities of an existing protocol.

Here is the description of the parameters –

- **socket\_family** – This is either AF\_UNIX or AF\_INET, as explained earlier.
- **socket\_type** – This is either SOCK\_STREAM or SOCK\_DGRAM.
- **protocol** – This is usually left out, defaulting to 0.

Once you have socket object, then you can use required functions to create your client or server program. Following is the list of functions required –

### SERVER SOCKET METHODS

Sr.No.	Method & Description
1	<b>s.bind()</b> This method binds address (hostname, port number pair) to socket.
2	<b>s.listen()</b> This method sets up and start TCP listener.

3	<b>s.accept()</b> This passively accept TCP client connection, waiting until connection arrives (blocking).
---	---

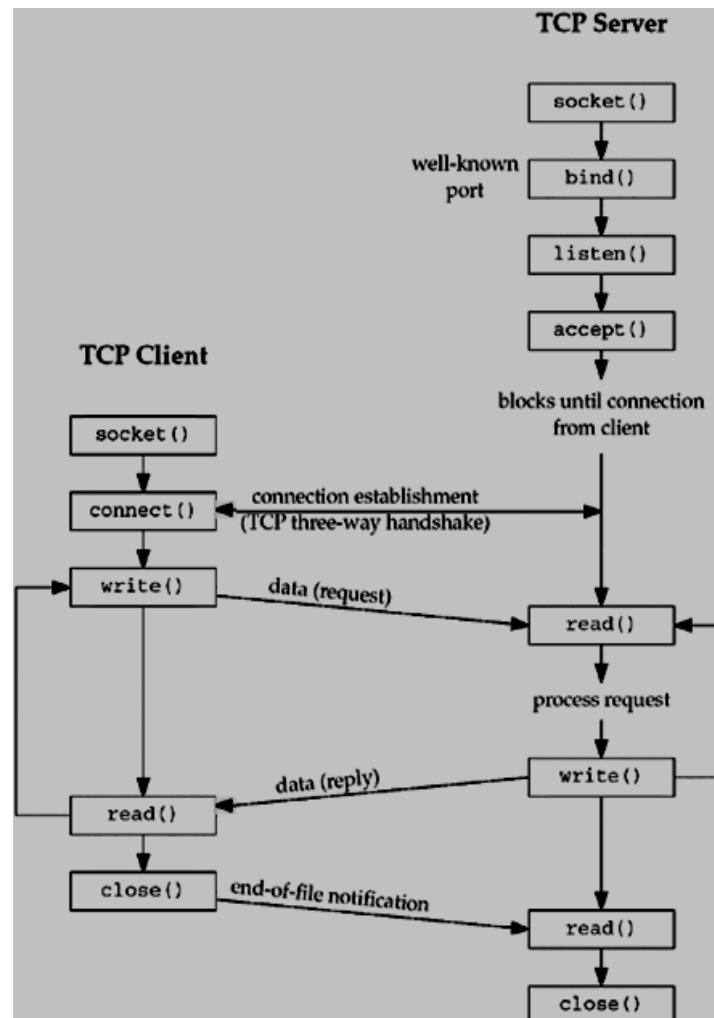
### CLIENT SOCKET METHODS

Sr.No.	Method & Description
1	<b>s.connect()</b> This method actively initiates TCP server connection.

### GENERAL SOCKET METHODS

Sr.No.	Method & Description
1	<b>s.recv()</b> This method receives TCP message
2	<b>s.send()</b> This method transmits TCP message
3	<b>s.recvfrom()</b> This method receives UDP message
4	<b>s.sendto()</b> This method transmits UDP message
5	<b>s.close()</b> This method closes socket
6	<b>socket.gethostname()</b> Returns the hostname.

**Methods Associated with Socket:**The following diagram shows the complete Client and Server interaction –



**CONCLUSION :** Thus we have successfully implemented the socket programming for TCP

## Practical No. 6

### Problem Definition:

**Write a program using UDP Sockets to enable file transfer (Script, Text, Audio and Video one file each) between two machines**

### Prerequisite:

a) Socket Header      b) Network Programming      c) Ports

### Learning Objectives:

1. To understand Work of Socket
2. Different methods associated with Client & Server Socket

### New Concepts:

1. Client Server Communication
2. Port Address

### Theory:

#### Introduction

What is UDP?

UDP is a connectionless and unreliable transport protocol. The two ports serve to identify the end points within the source and destination machines. User Datagram Protocol is used, in place of TCP, when a reliable delivery is not required. However, UDP is never used to send important data such as web-pages, database information, etc. Streaming media such as video, audio and others use UDP because it offers speed.

#### Why UDP is faster than TCP?

The reason UDP is faster than TCP is because there is no form of flow control. No error checking, error correction, or acknowledgment is done by UDP. UDP is only concerned with speed. So when, the data sent over the Internet is affected by collisions, and errors will be present. UDP packet's called as user datagrams with 8 bytes header. A format of user datagrams is shown in fig 3. In the user datagrams first 8 bytes contains header information and the remaining bytes contains data.

## **LINUX SOCKET PROGRAMMING:**

The Berkeley socket interface, an API, allows communications between hosts or between processes on one computer, using the concept of a socket. It can work with many different I/O devices and drivers, although support for these depends on the operating-system implementation. This interface implementation is implicit for TCP/IP, and it is therefore one of the fundamental technologies underlying the Internet. It was first developed at the University of California, Berkeley for use on Unix systems. All modern operating systems now have some implementation of the Berkeley socket interface, as it has become the standard interface for connecting to the Internet. Programmers can make the socket interfaces accessible at three different levels, most powerfully and fundamentally at the RAW socket level. Very few applications need the degree of control over outgoing communications that this provides, so RAW sockets support was intended to be available only on computers used for developing Internet-related technologies. In recent years, most operating systems have implemented support for it anyway, including Windows XP.

### **The header files:**

The Berkeley socket development library has many associated header files.

They include: **<sys/socket.h>**

Definitions for the most basic of socket structures with the BSD

**socket API <sys/socket.h>**

Basic data types associated with structures within the BSD socket API **<sys/types.h>**

**Socket API <sys/types.h>**

Definitions for the `socketaddr_in{}` and other base data structures

**<sys/un.h>**

Definitions and data type declarations for SOCK\_UNIX streams

UDP: UDP consists of a connectionless protocol with no guarantee of delivery. UDP packets may arrive out of order, become duplicated and arrive more than once, or even not arrive at all. Due to the minimal guarantees involved, UDP has considerably less overhead than TCP. Being connectionless means that there is no concept of a stream or connection between two hosts, instead, data arrives in datagrams. UDP address space, the space of UDP port numbers

(in ISO terminology, the TSAPs), is completely disjoint from that of TCP ports. Server: Code may set up a UDP server on port 7654 as follows:

```
sock = socket(PF_INET,SOCK_DGRAM,0);  
sa.sin_addr.s_addr = INADDR_ANY;  
sa.sin_port = htons(7654);  
bound = bind(sock,(struct sockaddr *)&sa, sizeof(struct sockaddr));  
if (bound < 0)  
fprintf(stderr, "bind(): %s\n",strerror(errno));  
listen(sock,3);
```

bind() binds the socket to an address/port pair. listen() sets the length of the new connections queue.

```
while (1)  
{  
printf("recv test...\n");  
recsize = recvfrom(sock, (void *)hz, 100, 0, (struct sockaddr *)&sa, fromlen);  
printf("recsize: %d\n",recsize);  
if (recsize < 0)  
fprintf(stderr, "%s\n", strerror(errno));  
sleep(1);  
printf("datagram: %s\n",hz);  
}
```

This infinite loop receives any UDP datagrams to port 7654 using recvfrom(). It uses the parameters: | socket | pointer to buffer for data | size of buffer | flags (same as in read or other receive socket function)

Client: A simple demo to send an UDP packet containing "Hello World!" to address 127.0.0.1, port 7654 might look like this:

```
#include #include #include #include #include  
#include int main(int argc, char *argv[])  
{  
int sock; struct sockaddr_in sa;
```

```

int bytes_sent, buffer_length;
char buffer[200];
sprintf(buffer, "Hello World!");
buffer_length = strlen(buffer) + 1;
sock = socket(PF_INET, SOCK_DGRAM, 0);
sa.sin_family = AF_INET;
sa.sin_addr.s_addr = htonl(0x7F000001);
sa.sin_port = htons(7654); bytes_sent = sendto(sock, buffer, buffer_length, 0, &sa,
sizeof(struct sockaddr_in) );
if(bytes_sent < 0) printf("Error sending packet: %s\n", strerror(errno) );
return 0;
}

```

In this code, buffer provides a pointer to the data to send, and buffer\_length specifies the size of the buffer contents. Typical UDP client code

- Create UDP socket to contact server (with a given hostname and service port number)
- Create UDP packet.
- Call send(packet), sending request to the server.
- Possibly call receive(packet) (if we need a reply).

Typical UDP Server code

- Create UDP socket listening to a well known port number.
- Create UDP packet buffer Call receive(packet) to get a request, noting the address of the client.
- Process request and send reply back with send(packet).

## APPLICATION :

Socket programming is essential in developing any application over a network.

**Conclusion:** Thus we have studied Working of UDP Socket successfully.

## Practical No. 7

**Problem Definition:** Capture packets using Wireshark, write the exact packet capture filter expressions to accomplish the following and save the output in file:

1. Capture all TCP traffic to/from Facebook, during the time when you log in to your Facebook account
2. Capture all HTTP traffic to/from Facebook, when you log in to your Facebook account
3. Write a DISPLAY filter expression to count all TCP packets (captured under item #1) that have the flags SYN, PSH, and RST set. Show the fraction of packets that had each flag set.
4. Count how many TCP packets you received from / sent to Facebook, and how many of each were also HTTP packets.

### Prerequisite:

1. Knowledge about Packet tracer
2. TCP and UDP basics

### Learning Objectives:

1. Understand the concept and working of Wireshark

### Theory

**Ethernet:** Ethernet is a way of connecting computers together in a local area network. It has been the most widely used method of linking computers together in LANs since the 1990s. The basic idea of its design is that multiple computers have access to it and can send data at any time.

This is comparatively easy to engineer. If two computers send data at the same time, a collision will occur. When this happens, the data sent is not usable. In general, both computers will stop sending, and wait a random amount of time, before they try again.

A special protocol was developed to deal with such problems. It is called Carrier sense multiple access with collision detection or CSMA/CD. Different cable types there are different Ethernet standards. Today, Ethernet cables look like thick telephone cables. They connect to boxes called hubs or switches.

Each cable runs from a computer's network interface card (NIC) to such a box. This cable is called 10BaseT or 100BaseT, or 1000BaseT Cable. All cable types: 10Base2 and 10Base5: These coaxial cables are like those used in television, but they are a bit thinner. They are also called "thinnet" or "coax". Each computer has a "T" plugged into it, and cables plug into each side of the "T". Sometimes, instead of a "T", a vampire tap is used which goes through the skin of a cable. It supports 10Mbits per second transfer speed, and was the first to be adopted. 10BaseT: Cables look like thick phone cables, but with 8 copper wires instead of 2 or 4, and

they go from each computer' to a Hub or a Switch. Supported speed is 10 MBit/second.  
10BaseF: Same as 10BaseT, but cables transmit light pulses, instead of electrical signals.  
100BaseT: Cables look the same as 10BaseT, but can run at up to 100 MBits per second  
1000BaseT: Cables look the same as 10BaseT, but can run at up to 1GBit (1000MBit) per second.

IP has the task of delivering packets from the source host to the destination host solely based on the IP addresses in the packet headers. For this purpose, IP defines packet structures that encapsulate the data to be delivered. It also defines addressing methods that are used to label the datagram with source and destination information. Historically, IP was the connectionless datagram service in the original

**Transmission Control Program** introduced by Vint Cerf and Bob Kahn in 1974; the other being the connection-oriented Transmission Control Protocol (TCP). The Internet protocol suite is therefore often referred to as TCP/IP.

The first major version of IP, Internet Protocol Version 4 (IPv4), is the dominant protocol of the Internet. Its successor is Internet Protocol Version 6 (IPv6). Function: The Internet Protocol is responsible for addressing hosts, encapsulating data into datagrams (including fragmentation and reassembly) and routing datagrams from a source host to a destination host across one or more IP networks. For these purposes, the Internet Protocol defines the format of packets and provides an addressing system.

Each datagram has two components: a header and a payload.

The IP header includes source IP address, destination IP address, and other metadata needed to route and deliver the datagram. The payload is the data that is transported. This method of nesting the data payload in a packet with a header is called encapsulation.

IP addressing entails the assignment of IP addresses and associated parameters to host interfaces. The address space is divided into subnetworks, involving the designation of network prefixes. IP routing is performed by all hosts, as well as routers, whose main function is to transport packets across network boundaries. Routers communicate with one another via specially designed routing protocols, either interior gateway protocols or exterior gateway protocols, as needed for the topology of the network. TCP: After going through the various layers of the Model, it's time to have a look at the TCP protocol and to study its functionality. This section will help the reader to get to know about the concepts and characteristics of the TCP, and then gradually dive into the details of TCP like connection establishment/closing, communication in TCP and why the TCP protocol is called a reliable as well as an adaptive protocol.

## What is TCP?

In theory, a transport layer protocol could be a very simple software routine, but the TCP protocol cannot be called simple. Why use a transport layer which is as complex as TCP? The most important reason depends on IP's unreliability. In fact all the layers below TCP are unreliable and deliver the datagram hop-by-hop.

The IP layer delivers the datagram hop-by-hop and does not guarantee delivery of a datagram; it is a connectionless system. IP simply handles the routing of datagrams; and if problems occur, IP discards the packet without a second thought, generating an error message back to the sender in the process. The task of ascertaining the status of the datagrams sent over a network and handling the resending of information

if parts have been discarded falls to TCP. Most users think of TCP and IP as a tightly knit pair, but TCP can be, and frequently is, used with other transport protocols. For example, TCP or parts of it are used in the File Transfer Protocol (FTP) and the Simple Mail Transfer Protocol (SMTP), both of which do not use IP. The Transmission Control Protocol provides a considerable number of services to the IP layer and the upper layers. Most importantly, it provides a connection-oriented protocol to the upper layers that enable an application to be sure that a datagram sent out over the network was received in its entirety. In this role, TCP acts as a message-validation protocol providing reliable communications.

### **TCP Header Format Source Port:**

16 bits The source port number. Destination Port: 16 bits The destination port number. Sequence Number: 32 bit The sequence number of the first data octet in this segment (except when SYN is present). If SYN is present the sequence number is the initial sequence number (ISN) and the first data octet is ISN+1. Acknowledgment Number: 32 bits If the ACK control bit is set this field contains the value of the next sequence number the sender of the segment is expecting to receive.

Once a connection is established this is always sent. Data Offset: 4 bits The number of 32 bit words in the TCP Header. This indicates where the data begins. The TCP header (even one including options) is an integral number of 32 bits long. Reserved: 6 bits Reserved for future use. Must be zero. Control Bits: 6 bits (from left to right): URG: Urgent Pointer field significant ACK: Acknowledgment field significant PSH: Push Function RST: Reset the connection SYN: Synchronize sequence numbers FIN: No more data from sender Window: 16 bits The number of data octets beginning with the one indicated in the acknowledgment field which the sender of this segment is willing to accept. Checksum: 16 bits

The checksum field is the 16 bit one's complement of the one's complement sum of all 16 bit words in the header and text. If a segment contains an odd number of header and text octets to be checksummed, the last octet is padded on the right with zeros to form a 16 bit word for checksum purposes. The pad is not transmitted as part of the segment. While computing the checksum, the checksum field itself is replaced with zeros.

The checksum also covers a 96 bit pseudo header conceptually prefixed to the TCP header. This pseudo header contains the Source Address, the Destination Address, the Protocol, and TCP length. This gives the TCP protection against misrouted segments. This information is carried in the Internet Protocol and is transferred across the TCP/Network interface in the arguments or results of calls by the TCP on the IP. The TCP Length is the TCP header length plus the data length in octets (this is not an explicitly transmitted quantity, but is computed), and it does not count the 12 octets of the pseudo header. Urgent Pointer: 16 bits This field communicates the current value of the urgent pointer as a positive offset from the sequence number in this segment.

The urgent pointer points to the sequence number of the octet following the urgent data. This field is only be interpreted in segments with the URG control bit set. Options: variable Options may occupy space at the end of the TCP header and are a multiple of 8 bits in length. All options are included in the checksum. An option may begin on any octet boundary. There are two cases for the format of an option:

Case 1: A single octet of option-kind.

Case 2: An octet of option-kind, an octet of option-length, and the actual option-data octets. The option-length counts the two octets of option-kind and option-length as well as the option-data octets. Note that the list of options may be shorter than the data offset field might imply. The content of the header beyond the End-of-Option option must be header padding (i.e., zero).

### **What is UDP?**

'Figure 2:UDP UDP is a connectionless and unreliable transport protocol.The two ports serve to identify the end points within the source and destination machines. User Datagram Protocol is used, in place of TCP, when a reliable delivery is not required.However, UDP is never used to send important data such as web-pages, database information, etc. Streaming media such as video,audio and others use UDP because it offers speed.

### **Why UDP is faster than TCP?**

The reason UDP is faster than TCP is because there is no form of flow control. No error checking,error correction, or acknowledgment is done by UDP.UDP is only concerned with speed. So when, the data sent over the Internet is affected by collisions, and errors will be present. UDP packet is called as user datagrams with 8 bytes header. A format of user datagrams is shown in figure

3. In the user datagrams first 8 bytes contains header information and the remaining bytes contains data.

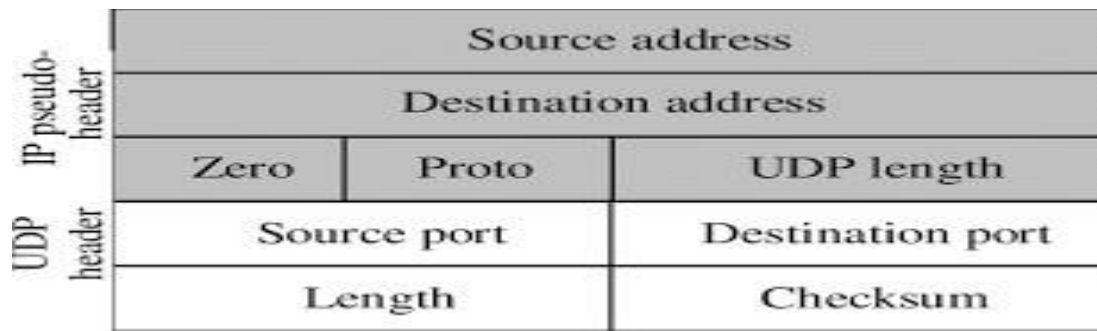


Figure 3:UDP datagrams Source port number: This is a port number used by source host,who is transferring data. It is 16 bit longs. So port numbers range between 0 to 65,535. Destination port number: This is a port number used by Destination host, who is getting data. It is also 16 bits long and also same number of port range like source host. length: Length field is a 16 bits field. It contains the total length of the user datagram, header and data. Checksum:

The UDP checksum is optional. It is used to detect error fro the data. If the field is zero then checksum is not calculated. And true calculated then field contains

1. Characteristics of UDP The characteristics of UDP are given below.

- End-to-end. UDP can identify a specific process running on a computer.
- Unreliable, connectionless delivery (e.g. USPS)::

**UDP uses a connectionless communication setup.**

In this UDP does not need to establish a connection before sending data. Communication consists only of the data segments themselves

- Same best effort semantics as IP
- No ack, no sequence, no flow control
- Subject to loss, duplication, delay, out-of-order, or loss of connection
- Fast, low overhead 1.Suit for reliable, local network 2.RTP(Real-Time Transport Protocol)

**Conclusion: Thus we have studied packet formats captured through Wireshark for wired network**

## Practical No. 8

**Problem Definition:** Study and Analyze the performance of HTTP, HTTPS and FTP protocol using Packet tracer tool.

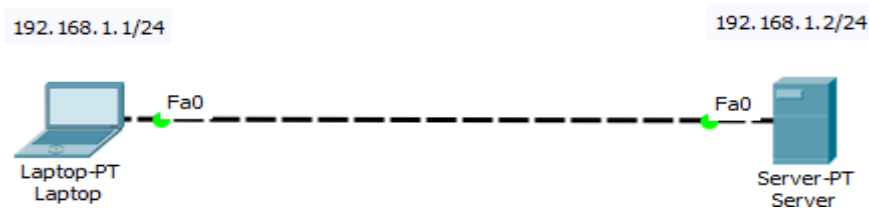
The File Transfer Protocol (FTP) is a standard network protocol used for the transfer of computer files between a client and server on a computer network.

FTP employs a client-server architecture whereby the client machine has an FTP client installed and establishes a connection to an FTP server running on a remote machine. After the connection has been established and the user is successfully authenticated, the data transfer phase can begin.

Worth noting: Although FTP does support user authentication, all data is sent in clear text, including usernames and passwords. For secure transmission that protects the username and password, and encrypts the content, FTP is often secured with SSL/TLS (FTPS) or replaced with SSH File Transfer Protocol (SFTP).

Let's now do FTP configuration in Packet Tracer:

1. Build the network topology.



FTP topology.PNG

2. Configure static IP addresses on the Laptop and the server.

Laptop: IP address: 192.168.1.1 Subnet Mask: 255.255.255.0

Server: IP address: 192.168.1.2 Subnet Mask: 255.255.255.0

3. Now try using an FTP client built in the Laptop to send files to an FTP server configured in the Server.

From the Laptop's command prompt, FTP the server using the server IP address by typing:  
ftp 192.168.1.2

Provide the username(cisco) and password(cisco) [which are the defaults] for ftp login.

```
C:\>
C:\>ftp 192.168.1.2
Trying to connect...192.168.1.2
Connected to 192.168.1.2
220- Welcome to FT Ftp server
Username:cisco
331- Username ok, need password
Password:
330- Logged in
(passive mode On)
ftp>
```

ftp from laptop.PNG

You are now in the FTP prompt .

PC0 has an FTP client which can be used to read, write, delete and rename files present in the FTP server.

The FTP server can be used to read and write configuration files as well as IOS images. Additionally, the FTP server also supports file operations such rename, delete and listing directory.

With that in mind, we can do something extra. So let's do this:

4. Create a file in the Laptop then upload it to the server using FTP.

To do this, open the Text Editor in the Laptop, create a file and give it your name of choice.

Type any text in the editor then save your file. e.g. myFile.txt.

5. Now upload the file from the Laptop to the server using FTP. (An FTP connection has to be started first. But this is what we've done in step 3)

So to do an FTP upload, we'll type:

```
put MyFile.txt
```

```

ftp>
ftp>put MyFile.txt

Writing file MyFile.txt to 192.168.1.2:
File transfer in progress...

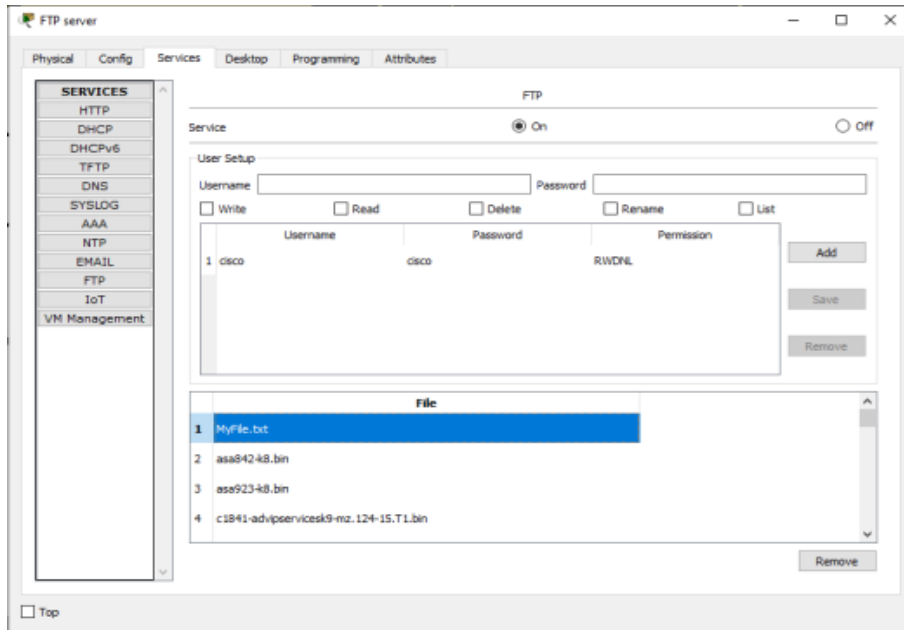
[Transfer complete - 47 bytes]

47 bytes copied in 0.023 secs (2043 bytes/sec)
ftp>

```

put MyFile to FTP directory.PNG

6. Once file upload is successful, go to the Server FTP directory to verify if the file sent has been received . To do this, go to Server-> Services->FTP. Here look for MyFile.txt sent from the laptop.



MyFile.txt really send to sever.PNG

Something extra: To check other FTP commands supported by the FTP client running on the Laptop(or PC), you can use a question mark (?) on the Laptop's command prompt as shown below:

All FTP commands supported

You can see the put command that we used to upload our file to the FTP server. Other commands listed include:

get-used to get(download) a file from the server.

For example: get MyFile.txt

delete- to delete a file in the FTP directory with the server

For example: delete MyFile.txt

Rename- used to Rename a file

cd - used to change directory.

For example, we can open an HTTP directory in the server by typing: cd /http. This will change the current directory from FTP directory to HTTP directory

Once the http directory is open, you can upload a file to the HTTP server. You're now uploading a file to an HTTP folder(directory) using FTP.

For example: put MyFile.txt

```
ftp>cd /http
ftp>
Working directory changed to /http successfully
ftp>put MyFile.txt

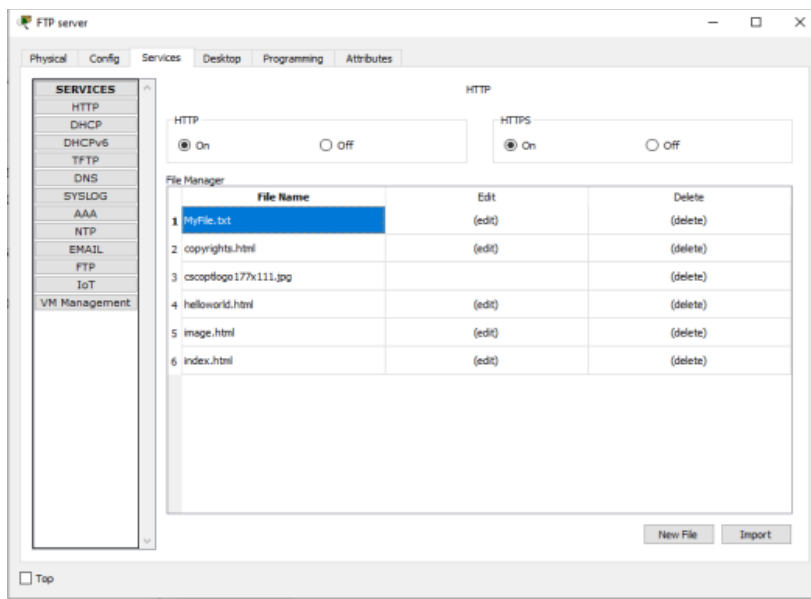
Writing file MyFile.txt to 192.168.1.2:
File transfer in progress...

[Transfer complete - 47 bytes]
47 bytes copied in 0.01 secs (4700 bytes/sec)
```

To see this working, let's open an HTTP directory and upload(put) a file to it using FTP:

changing directory then put files to HTTP directory using FTP

You can now check up in the HTTP directory in the server and verify that the file uploaded from the Laptop(MyFile.txt) is well received:



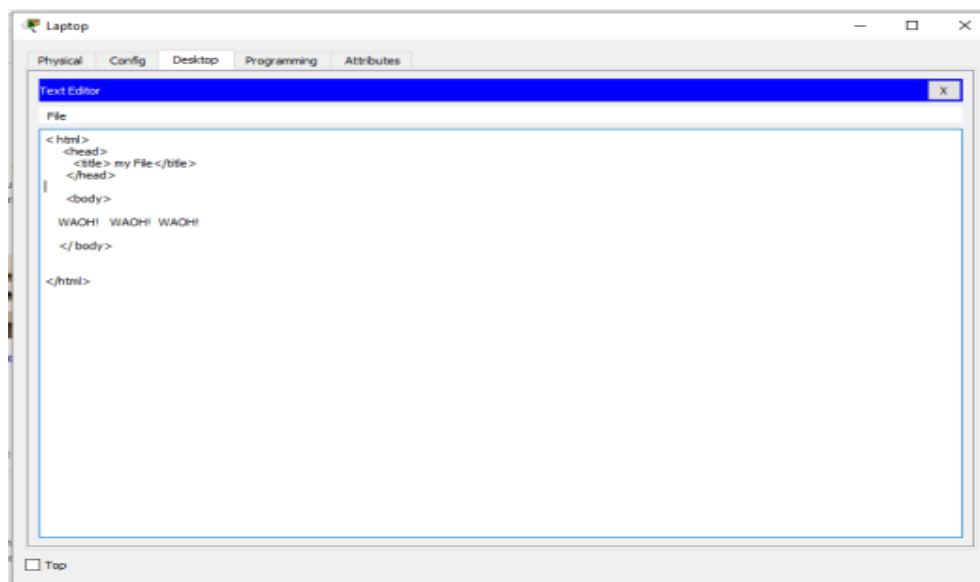
MyFile.txt really send to HTTP server

Notice that we are uploading files to an HTTP Server directory using File Transfer Protocol.(FTP). This is what actually happens when you use an FTP client such as FileZilla client to upload files to a website. In our case here, we are using an FTP client built-in the Laptop.

This may interest you: The first FTP client applications were command-line programs developed before operating systems had graphical user interfaces, and are still shipped with most Windows and Linux operating systems. (Actually this is what we have been using this far). Many FTP clients(e.g. FileZilla) and automation utilities have since been developed for desktops, servers, mobile devices, and hardware. FTP has also been incorporated into productivity applications, such as HTML editors.

We'll create an html file in our Laptop, upload it to HTTP server directory using FTP, then try to access the file from the Laptop's browser.

On the Laptop, open the text editor, then type some markup(html) and save the file with the extension .html. See all this below:



File2 HTML code

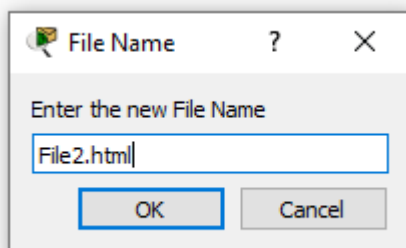
```
C:\>ftp 192.168.1.2
Trying to connect...192.168.1.2
Connected to 192.168.1.2
220- Welcome to PT Ftp server
Username:cisco
331- Username ok, need password
Password:
230- Logged in
(passive mode On)
ftp>cd /http
ftp>
Working directory changed to /http successfully
ftp>put File2.html

Writing file File2.html to 192.168.1.2:
File transfer in progress...

[Transfer complete - 136 bytes]

136 bytes copied in 0.041 secs (3317 bytes/sec)
ftp>
```

Save your file as an html file like this:



File2 html.PNG

Now upload the file( File2.html) to the HTTP server using FTP. This is easy. We've already done it previously!

If you're already in the HTTP directory, you just need to type: put File2.html. If no, first ftp the server(ftp 192.168.1.2), provide the login username(cisco) and password(cisco); change the current directory to HTTP(cd /http) , and finally upload the html file onto the HTTP directory(put File2.html)

```
C:\>ftp 192.168.1.2
Trying to connect...192.168.1.2
Connected to 192.168.1.2
220- Welcome to PT Ftp server
Username:cisco
331- Username ok, need password
Password:
230- Logged in
(passive mode On)
ftp>cd /http
ftp>
Working directory changed to /http successfully
ftp>put File2.html

Writing file File2.html to 192.168.1.2:
File transfer in progress...

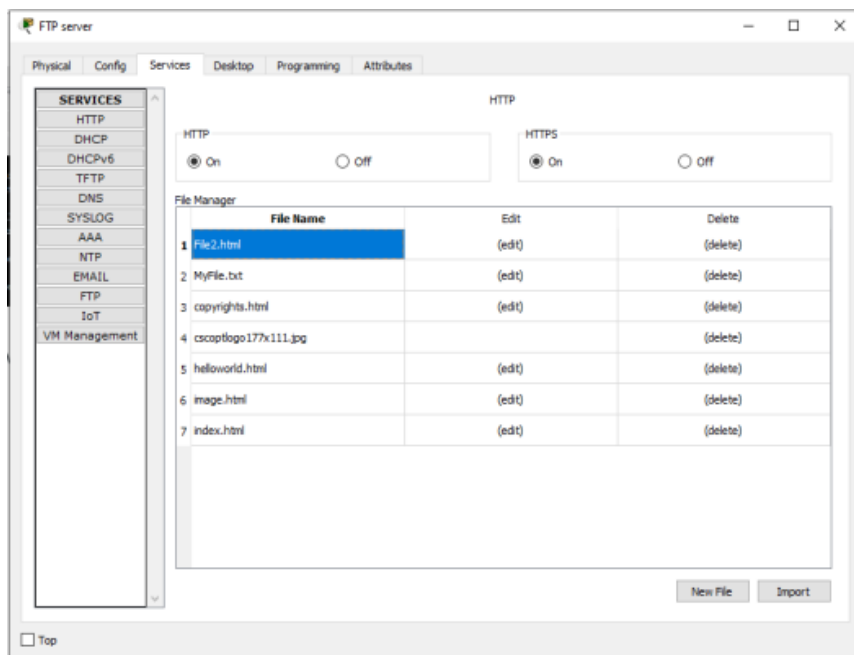
[Transfer complete - 136 bytes]

136 bytes copied in 0.041 secs (3317 bytes/sec)
ftp>
```

Sending File2. html to HTTP directory.PNG

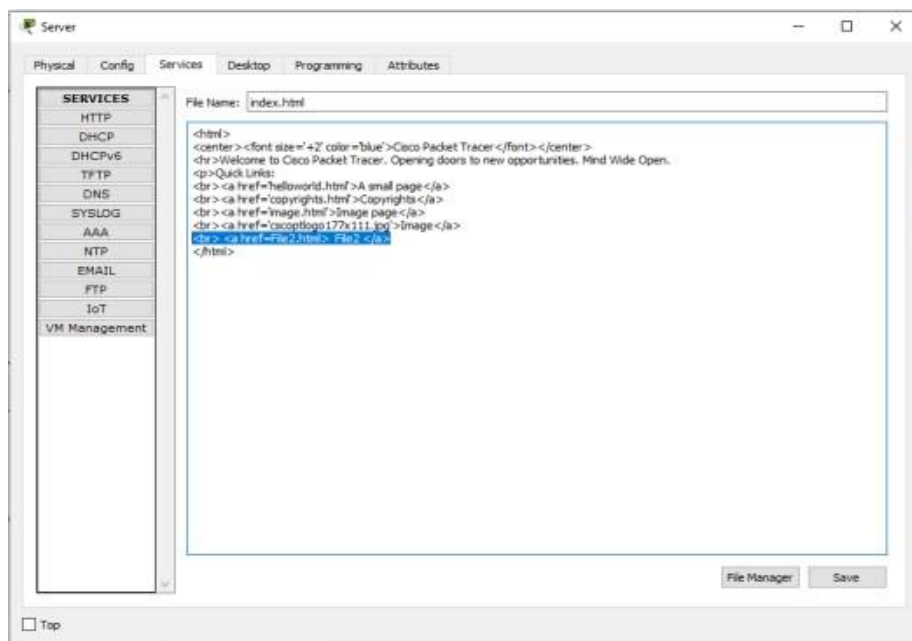
Check whether the html file uploaded has been received in the HTTP directory:

Go to Server->Services-> HTTP. Then look up for the file in the File Manager.



File2 HTML really uploaded into HTTP directory.PNG

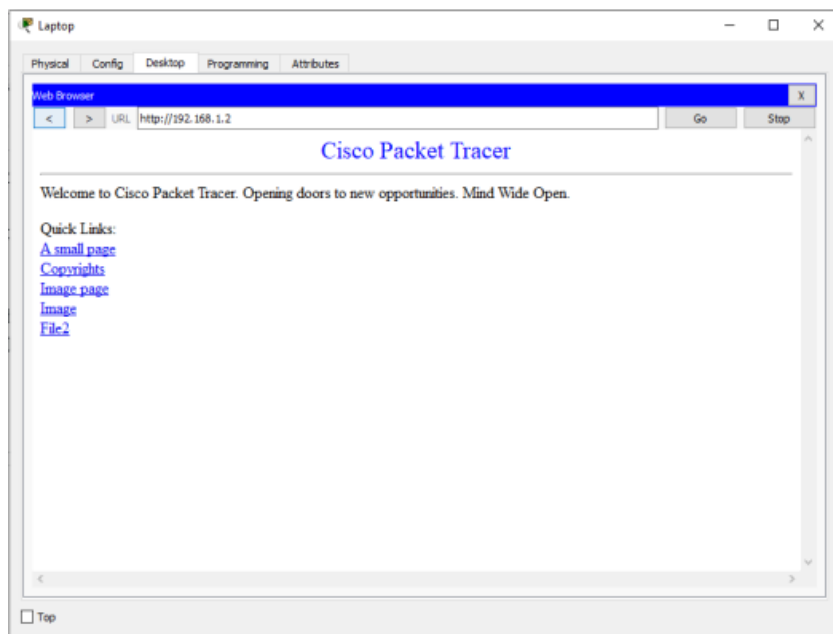
Now edit index.html file in the HTTP directory so as to include a link to File2 that we've just uploaded. This will make File2 accessible from the Laptop's browser. To do this, locate index.html then click edit. Proceed to edit it as shown below. Then save and accept overwrite. Index.html editing to include File2 html.PNG



Finally, try to access the newly uploaded file from the Laptop's browser.

So go to the Laptop's browser and access the server using the server's IP address. By doing this, the browser is making an http request to the server. The server will respond to the Laptop with the index.html file containing a link to File2 which we've uploaded from the Laptop using FTP.

Http response with File2.PNG



Click File2 link to view the contents of the file in the browser.

**Conclusion:** Thus we have studied the performance of HTTP, HTTPS and FTP protocol using Packet tracer tool.

**Problem Definition:** Illustrate the steps for implementation of S/MIME email security, POP3 through Microsoft® Office Outlook.

Secure/Multipurpose Internet Mail Extensions (S/MIME) is an internet standard that allows the sender of an email to protect the confidentiality of the message by encrypting its content with the public key contained in the recipient's S/MIME certificate.

Outlook on the recipient's computer can then decrypt the message with the private key installed on their device. S/MIME is supported natively by Microsoft Windows and Outlook, enabling end-to-end encryption and sender authentication through digital signatures.

### 1. Getting an S/MIME Certificate

You can purchase an S/MIME certificate from a Certificate Authority (CA) or retailer, or also obtain one free of charge from Actalis, an Italian CA.

Certificate Authorities provide S/MIME certificate bundles either as a PKCS #12 file (.p12 or .pfx) if they generated the certificate for you, or as a PKCS #7 (.p7b) file if you created the private key on your own computer and submitted a Certificate Signing Request (CSR) to the CA.

### 2. Installing the S/MIME Certificate in Outlook

**Note:** The instructions provided here were tested with Microsoft Outlook on Windows 10 in March 2021.

#### Steps for Installation:

1. Download and unzip the certificate bundle if needed
2. Launch Outlook and select *File > Options from the main menu*
3. Select *Trust Center > Trust Center Settings*
4. Select *Email Security* and click the *Import/Export* button to import the S/MIME certificate
5. Browse to the S/MIME certificate file location on your computer
6. Locate the *Security Profile* (i.e., S/MIME certificate) to import into Outlook
7. Enter the password associated with the S/MIME certificate (also known as *Digital ID* or *Security Profile*)  
*Note: If you obtained the certificate as a .p12 or .pfx file from a certificate authority, they must have also given you the password*
8. Unless you need enhanced security, leave the security level set to Medium and click OK on the pop-up dialog box
9. Allow protected access to your S/MIME certificate's private key

### 3. Turning On S/MIME Signing and Encryption

We will now set up Outlook to digitally sign outgoing mail with the new S/MIME identity. This will allow our email contacts with S/MIME-compatible email software to:

- Authenticate the messages we send.
- Automatically import our public key so they can use it in the future to send us encrypted emails.

#### Steps for Activation:

- Click the Settings button under Encrypted email
- Name your security settings and make sure *Cryptography* format is set to *S/MIME*
- Check *Default Security Settings* for this cryptographic format
- Check *Security Settings* for all cryptographic messages

#### Signing Certificate

- Click *Choose...* to browse to the S/MIME certificate file and click OK to confirm.

#### Encryption Certificate

- Click *Choose...* to browse to the S/MIME certificate file and confirm.
- Finally, check Send these certificates with signed messages.

### 4. Encrypted Email Settings

Go back to the Email Security tab and set the default options for S/MIME email as below:

- Check Encrypt contents and attachments for outgoing messages.
- Check Add digital signature to outgoing messages.

You should now be able to send emails digitally signed with your S/MIME identity and receive encrypted emails. Outlook should automatically import the S/MIME public key of anyone who sends you a signed email, so you should also be able to send encrypted messages to anyone who has sent you a signed message before.

Setting up S/MIME in Microsoft Outlook is a detailed but essential process to ensure the security and confidentiality of your emails. By following these steps, you protect your communications through encryption and digital authentication, enhancing security and preventing unauthorized access and phishing attacks.

**Conclusion:** Thus we have studied the steps for implementation of S/MIME email security, POP3 through Microsoft® Office Outlook.

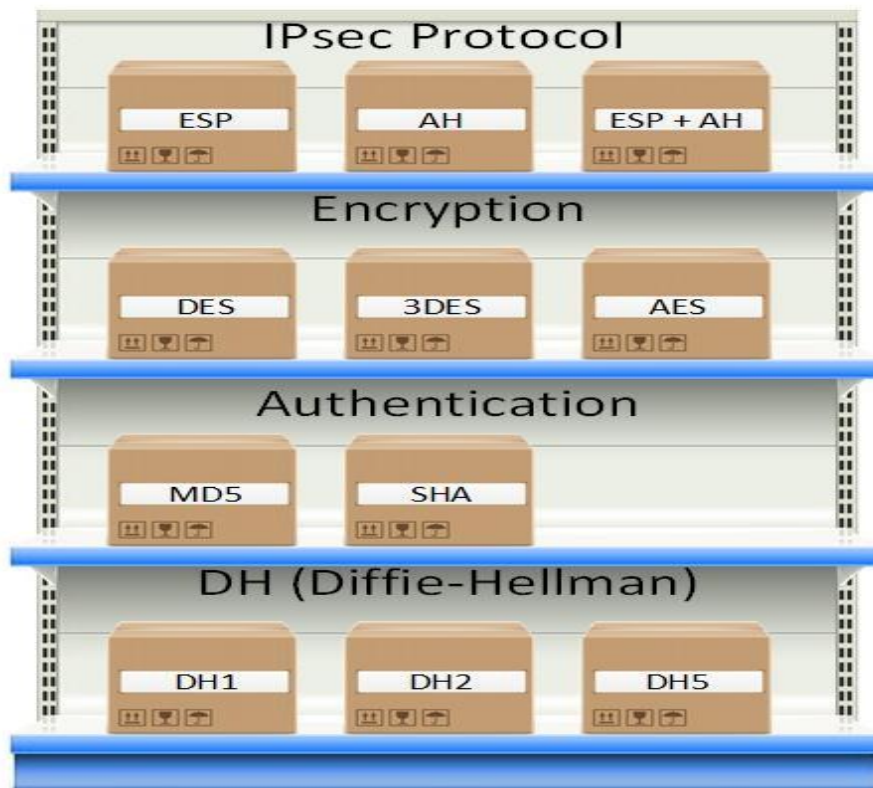
## Practical No. 10

**Problem Definition:** To study the IPsec (ESP and AH) protocol by capturing the packets using Wireshark tool.

IPsec (Internet Protocol Security) is a framework that helps us to protect IP traffic on the network layer. Why? Because the IP protocol itself doesn't have any security features at all. IPsec can protect our traffic with the following features:

- **Confidentiality:** by encrypting our data, nobody except the sender and receiver will be able to read our data.
- **Integrity:** we want to make sure that nobody changes the data in our packets. By calculating a hash value, the sender and receiver will be able to check if changes have been made to the packet.
- **Authentication:** the sender and receiver will authenticate each other to make sure that we are really talking with the device we intend to.
- **Anti-replay:** even if a packet is encrypted and authenticated, an attacker could try to capture these packets and send them again. By using sequence numbers, IPsec will not transmit any duplicate packets.

As a framework, IPsec uses a variety of protocols to implement the features. Here's an overview:



IPsec can be used on many different devices, it's used on routers, firewalls, hosts and servers. Here are some examples how you can use it:

- Between two routers to create a site-to-site VPN that “bridges” two LANs together.
- Between a firewall and windows host for remote access VPN.
- Between two linux servers to protect an insecure protocol like telnet.

IPsec is pretty complex and there are a lot of different ways to implement it. In this lesson I will start with an overview and then we will take a closer look at each of the components.

Before we can protect any IP packets, we need two IPsec peers that build the IPsec tunnel.

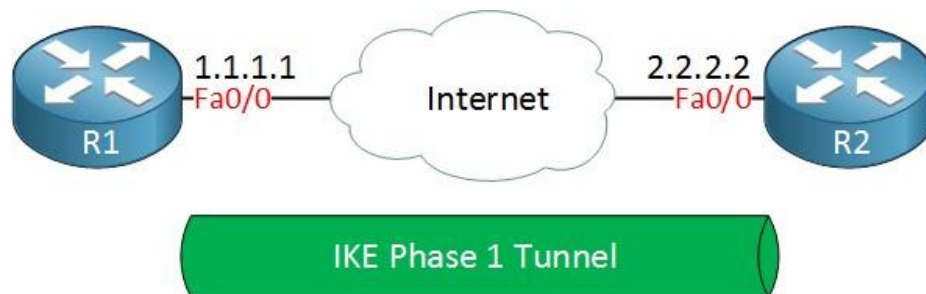
To establish an IPsec tunnel, we use a protocol called **IKE (Internet Key Exchange)**.

There are **two phases** to build an IPsec tunnel:

- **IKE phase 1**
- **IKE phase 2**

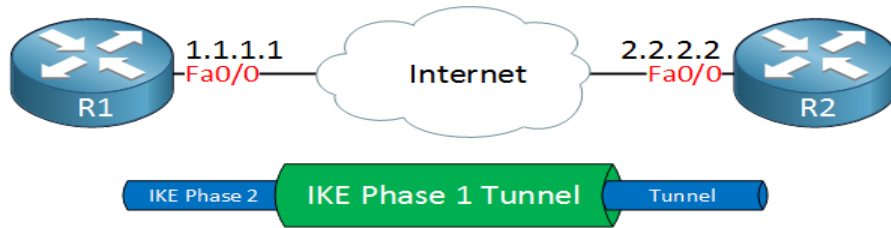
In IKE phase 1, two peers will negotiate about the encryption, authentication, hashing and other protocols that they want to use and some other parameters that are required. In this phase, an **ISAKMP (Internet Security Association and Key Management Protocol)** session is established. This is also called the **ISAKMP tunnel** or **IKE phase 1 tunnel**.

The collection of parameters that the two devices will use is called a **SA (Security Association)**. Here's an example of two routers that have established the IKE phase 1 tunnel:

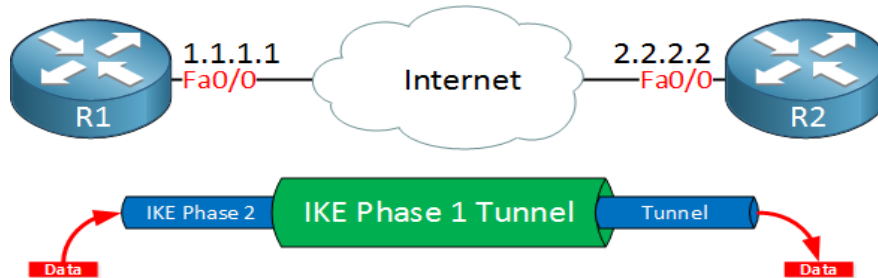


The IKE phase 1 tunnel is only used for **management traffic**. We use this tunnel as a *secure method to establish the second tunnel* called the **IKE phase 2 tunnel** or **IPsec tunnel** and for management traffic like keepalives.

Below is the fig of our two routers that completed IKE phase 2:



Once IKE phase 2 is completed, we have an IKE phase 2 tunnel (or IPsec tunnel) that we can use to protect our user data. This user data will be sent through the IKE phase 2 tunnel:



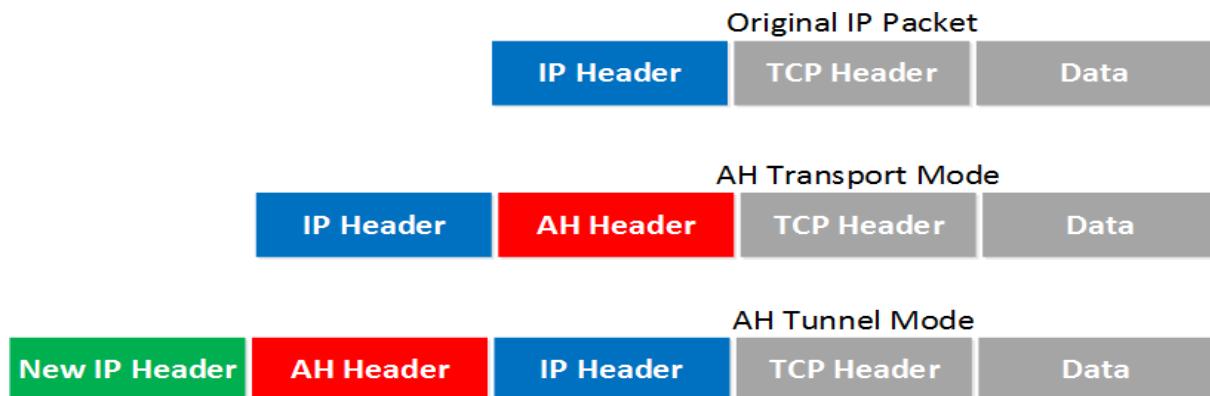
IKE builds the tunnels for us but it doesn't authenticate or encrypt user data. We use two other protocols for this:

- **AH (Authentication Header)**
- **ESP (Encapsulating Security Payload)**

Both protocols support two different modes:

- **Transport mode**
- **Tunnel mode**

The main difference between the two is that with transport mode we will use the **original IP header** while in tunnel mode, we use a new **IP header**. Here's an example to help you visualize this:



Transport mode is often between two devices that want to protect some insecure traffic (example: telnet traffic). Tunnel mode is typically used for site-to-site VPNs where we need to encapsulate the original IP packet since these are mostly private IP addresses and can't be routed on the Internet. I will explain these two modes in detail later in this lesson.

The entire process of IPsec consists of five steps:

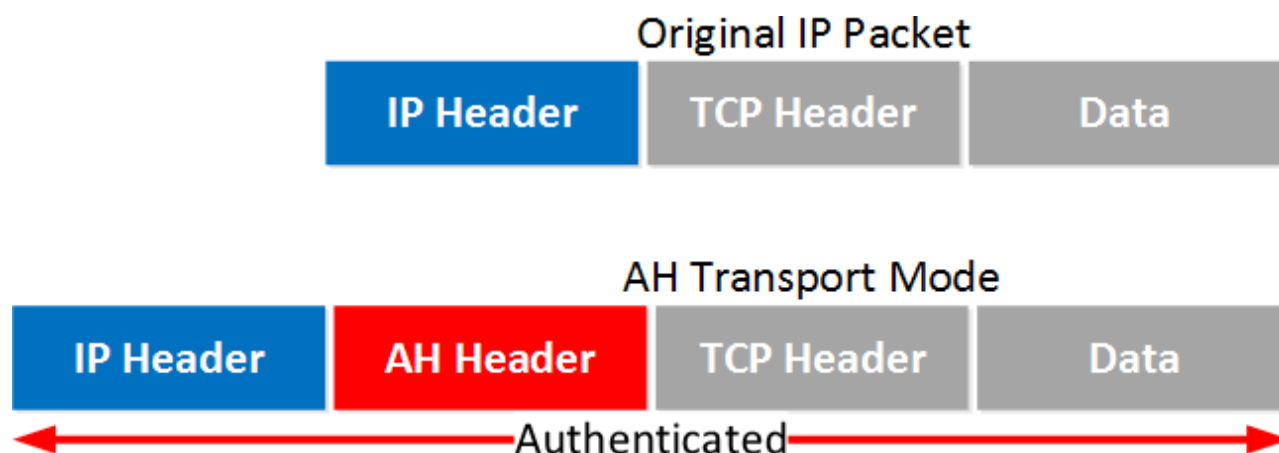
- **Initiation:** something has to trigger the creation of our tunnels. For example when you configure IPsec on a router, you use an access-list to tell the router what data to protect. When the router receives something that matches the access-list, it will start the IKE process. It's also possible to manually initiate the tunnel.
- **IKE phase 1:** we negotiate a security association to build the IKE phase 1 tunnel (ISAKMP tunnel).
- **IKE phase 2:** within the IKE phase 1 tunnel, we build the IKE phase 2 tunnel (IPsec tunnel).
- **Data transfer:** we protect user data by sending it through the IKE phase 2 tunnel.
- **Termination:** when there is no user data to protect then the IPsec tunnel will be terminated after a while.

## ➤ Authentication Header Protocol

AH offers authentication and integrity but it doesn't offer any encryption. It protects the IP packet by calculating a hash value over almost all fields in the IP header. The fields it excludes are the ones that can be changed in transit (TTL and header checksum). Let's start with transport mode...

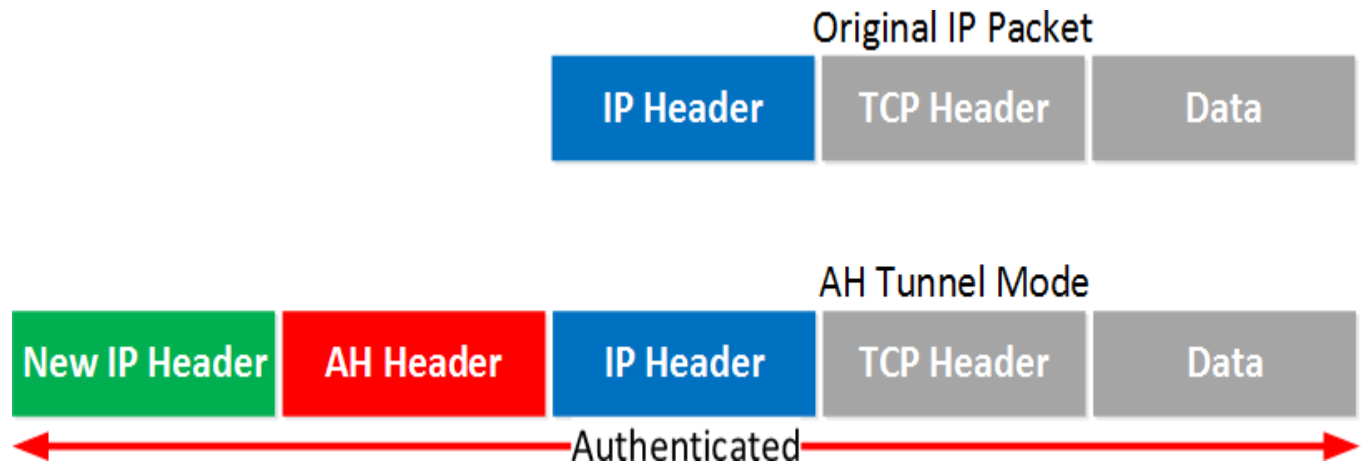
### ➤ Transport Mode

Transport mode is simple, it just adds an AH header after the IP header. Here's an example of an IP packet that carries some TCP traffic:



### ➤ Tunnel Mode

With tunnel mode we add a new IP header on top of the original IP packet. This could be useful when you are using private IP addresses and you need to tunnel your traffic over the Internet. It's possible with AH but it doesn't offer encryption:

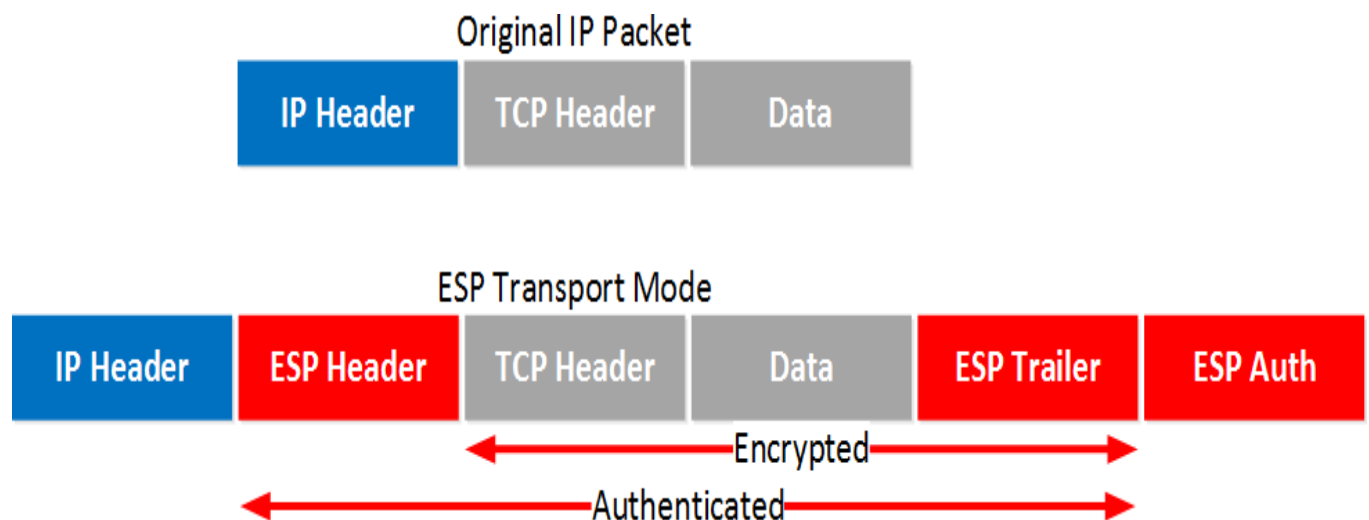


### ➤ ESP (Encapsulating Security Payload) Protocol

ESP is the more popular choice of the two since it allows you to encrypt IP traffic. We can use it in transport or tunnel mode, let's look at both.

#### ➤ Transport Mode

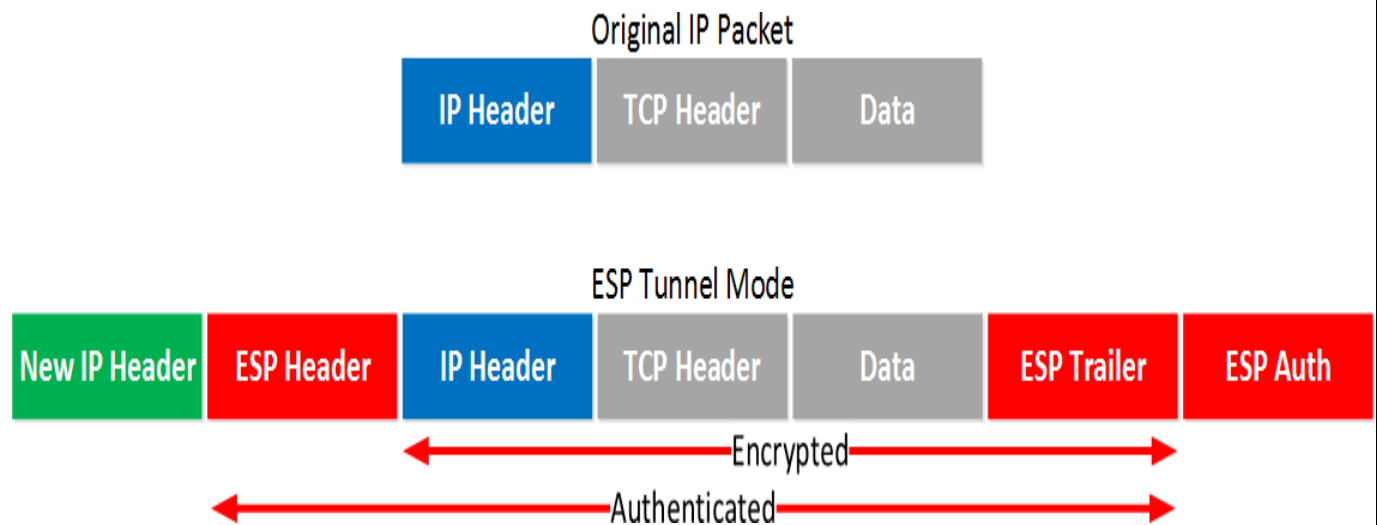
When we use transport mode, we use the original IP header and insert an ESP header. Here's what it looks like:



Above you can see that we add an ESP header and trailer. Our transport layer (TCP for example) and payload will be encrypted. It also offers authentication but unlike AH, it's not for the entire IP packet.

### ➤ Tunnel Mode

How about ESP in tunnel mode? This is where we use a new IP header which is useful for site-to-site VPNs:



It's similar to transport mode but we add a new header. The original IP header is now also encrypted.

**Conclusion:** Thus we have studied IPsec (ESP and AH) protocol by capturing the packets using Wireshark tool

## Practical No. 11

### Problem Definition:

**Installing and configure DHCP server and write a program to install the software on remote machine.**

### Prerequisite:

1. Knowledge about IP and Subnets.
2. Linux basic commands.

### Learning Objectives:

1. Understand the concept of DHCP.
2. Configuring DHCP and installation of software.

### New Concepts:

1. Crimping
2. Access Point Configuration

### Theory

#### Introduction

- DHCP (Dynamic Host Configuration Protocol) is a protocol that lets network administrators manage centrally and automate the assignment of IP (Internet Protocol) configurations on a computer network.
- When using the Internet's set of protocols (TCP/IP), in order for a computer system to communicate to another computer system it needs a unique IP address.
- Without DHCP, the IP address must be entered manually at each computer system. DHCP lets a network administrator supervise and distribute IP addresses from a central point.
- The purpose of DHCP is to provide the automatic (dynamic) allocation of IP client configurations for a specific time period (called a lease period) and to eliminate the work necessary to administer a large IP network.
- When connected to a network, every computer must be assigned a unique address.
- However, when adding a machine to a network, the assignment and configuration of network (IP) addresses has required human action.
- The computer user had to request an address, and then the administrator would manually configure the machine. Mistakes in the configuration process are easy for novices to make, and can cause difficulties for both the administrator making the

- error as well as neighbors on the network. Also, when mobile computer users travel between sites, they have had to relive this process for each different site from which they connected to a network.
- In order to simplify the process of adding machines to a network and assigning unique IP addresses manually, there is a need to automate the task.
- The introduction of DHCP alleviated the problems associated with manually assigning TCP/IP client addresses. Network administrators have quickly appreciated the importance, flexibility and ease-of-use offered in DHCP.

#### **Advantages of DHCP:-**

DHCP has several major advantages over manual configurations.

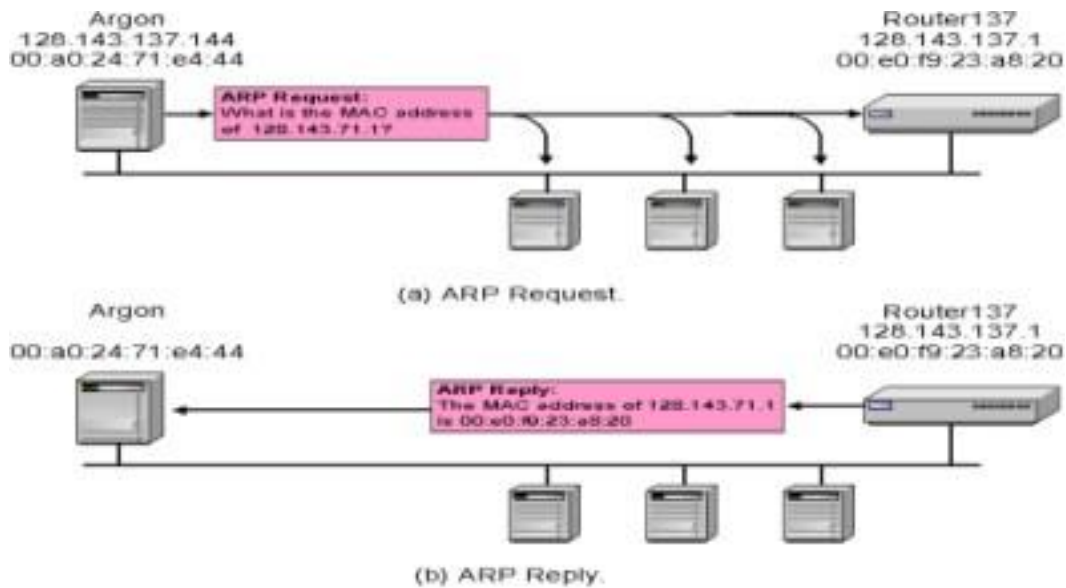
- Each computer gets its configuration from a "pool" of available numbers automatically for a specific time period (called a leasing period), meaning no wasted numbers.
- When a computer has finished with the address, it is released for another computer to use. Configuration information can be administered from a single point.
- Major network resource changes (e.g. a router changing address), requires only the DHCP server be updated with the new information, rather than every system.

#### **DHCP message types:**

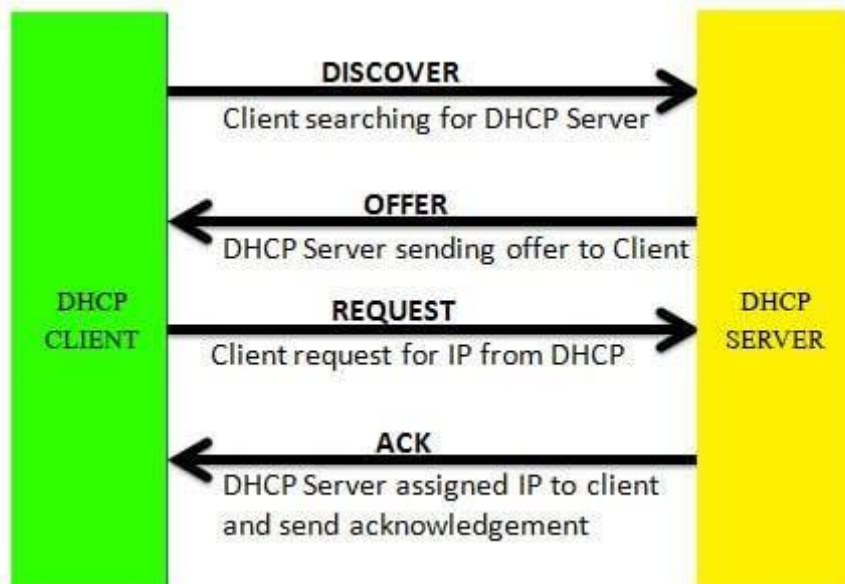
Value	Message Type
1	DHCPDISCOVER
2	DHCPOFFER
3	DHCPREQUEST
4	DHCPDECLINE
5	DHCPACK
6	DHCPNAK
7	DHCPRELEASE
8	DHCPINFORM

## DHCP Operations:-

### 1. DHCP Discover



### 2. DHCP Offer



3. DHCP Discover: At this time, the DHCP client can start to use the IP address

4. DHCP Release: At this time, the DHCP client has released the IP address

#### **5.4.4 Installing DHCP in Ubuntu:**

**Open terminal and type following commands:-**

1. `sudo apt-get install isc-dhcp-server`

2. `sudo gedit /etc/dhcp/dhcpd.conf` then make changes in file....

`default-lease-time 600; max-lease-time 7200;`

`option subnet-mask 255.255.255.0;`

`option broadcast-address 10.1.32.255;`

`subnet 192.168 1.0 netmask 255.255.255.0`

`range 10.1.32.10 10.1.32.20; }`

3. save file and close

4. again on terminal give following commands....

`sudo service isc-dhcp-server restart`

`sudo service isc-dhcp-server start`

5. On another PC in Internet properties change to Obtain IP address automatically and then check the IP address.

#### **Conclusion:**

Hence we Installed and Configured DHCP and studied Installation of Software on remote Machine.

## Practical No. 12

### Problem Definition:

Write a program for DNS lookup. Given an IP address input, it should return URL and vice versa.

### Prerequisite:

1. IP Address and OSI & TCP/IP Model.
2. Role of different servers.
3. Working of internet.

### Learning Objectives:

1. Understand what is Domain Name System and DNS lookup working.
2. Understand what is DNS Structure and Hierarchy.

### New Concepts:

1. Name Server and Domain Name System.
2. DNS lookup, Zone

### Theory

#### Introduction

The Domain Name System (DNS) is a hierarchical decentralized naming system for computers, services, or other resources connected to the Internet or a private network. It associates various information with domain names assigned to each of the participating entities.

It translates more readily memorized domain names to the numerical IP addresses needed for locating and identifying computer services and devices with the underlying network protocols. By providing a worldwide, distributed directory service, the Domain Name System is an essential component of the functionality on the Internet that has been in use since 1985.

#### HOST.TXT files:

The ARPANET, the predecessor of the Internet, had no distributed host name database. Each network node maintained its own map of the network nodes as needed and assigned those names that were memorable to the users of the system.

The hosts file contains lines of text consisting of an IP address in the first text field followed by one or more host names. Each field is separated by white space – tabs are often preferred for historical reasons, but spaces are also used. Comment lines may be included; they are indicated by an octothorpe (#) in the first position of such lines. Entirely blank lines in the file are ignored. For example, a typical hosts file may contain the following:

```
127.0.0.1 localhost loopback
::1 localhost
```

#### Domain Name Space

The domain name space refers to a hierarchy in the internet naming structure. This hierarchy

has multiple levels (from 0 to 127), with a root at the top. The following diagram shows the domain name space hierarchy:

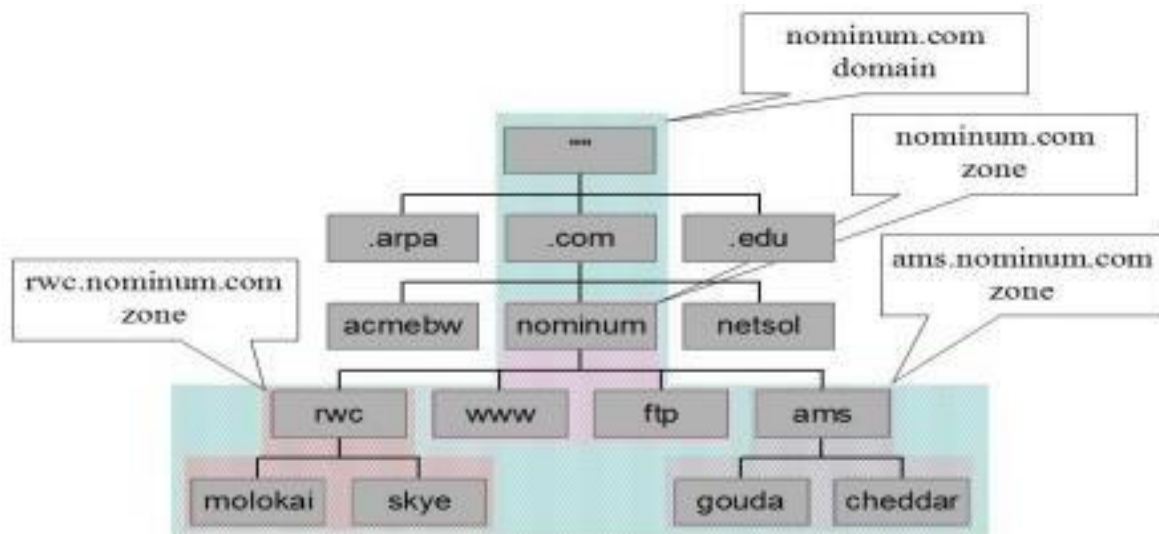
### Name Server

Name server contains the DNS database. This database comprises of various names and their corresponding IP addresses. Since it is not possible for a single server to maintain entire DNS database, therefore, the information is distributed among many DNS servers.

- Hierarchy of server is same as hierarchy of names.
- The entire name space is divided into the zones

### Zones

Zone is collection of nodes (sub domains) under the main domain. The server maintains a database called zone file for every zone.

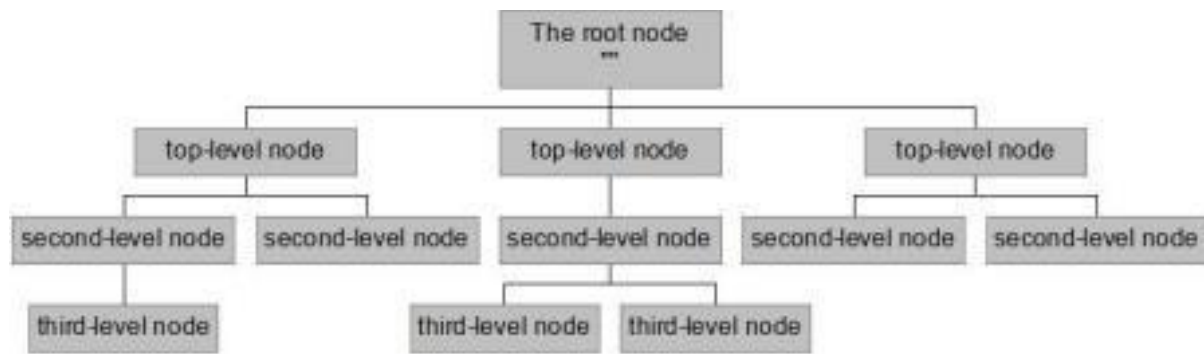


If the domain is not further divided into sub domains then domain and zone refers to the same thing. The information about the nodes in the sub domain is stored in the servers at the lower levels however; the original server keeps reference to these lower levels of servers.

### Types of Name Servers

Following are the three categories of Name Servers that manages the entire Domain Name System:

2. Root Server
3. Primary Server
4. Secondary Server



### Root Server

Root Server is the top level server which consists of the entire DNS tree. It does not contain the information about domains but delegates the authority to the other server

### Primary Servers

Primary Server stores a file about its zone. It has authority to create, maintain, and update the zone file.

### Secondary Server

Secondary Server transfers complete information about a zone from another server which may be primary or secondary server. The secondary server does not have authority to create or update a zone file.

### How does DNS work?

DNS servers answer questions from both inside and outside their own domains. When a server receives a request from outside the domain for information about a name or address inside the domain, it provides the authoritative answer. When a server receives a request from inside its own domain for information about a name or address outside that domain, it passes the request out to another server -- usually one managed by its internet service provider. If that server does not know the answer or the authoritative source for the answer, it will reach out to the DNS servers for the top-level domain -- e.g., for all of .com or .edu. Then, it will pass the request down to the authoritative server for the specific domain -- e.g., techtarget.com or stkate.edu; the answer flows back along the same path.

### How DNS Lookup Works

By now, you know that there are different servers hosting databases that contain the IP addresses of different domains and their sub-domains. You also know that there are Root Servers that hold the IP address of servers hosting Top Level Domains. These Root Servers help in reaching the servers containing databases that hold IP address of the main domain name. If there are sub-domains, their address can be on the same servers as of the main domain name or on a different server. All these servers are accessible for finding out the IP address of the exact URL that you need to use.

- **Forward Lookup:** When a name query is send to the DNS server against to IP address, it is generally said a forward lookup.

**Reverse Lookup:** DNS also provides a reverse lookup process, enabling clients to use a known IP address during a name query and look up a computer name based on its address.

**The process of finding out the IP address of any URL on the Internet is known as DNS lookup.**

To find out how DNS Lookup works, take the following example.

**Example:** Consider a network of ten computers. Each computer has its own address so that data packets travelling in the network know where to go. There is a 11th computer that hosts a database containing the alias names of each of these ten computers and their IP addresses. While the computer users can refer to the computers using their names, the data packets need the IP addresses of the computers so that they can reach the intended recipient. If computer A needs to use the printer attached to computer B, A will check the database on 11th computer to know the IP address of B and then find out the address of printer attached to B. Only after obtaining the address of the printer, A will route the print command to printer attached to B.

In this case, the following iterations happen:

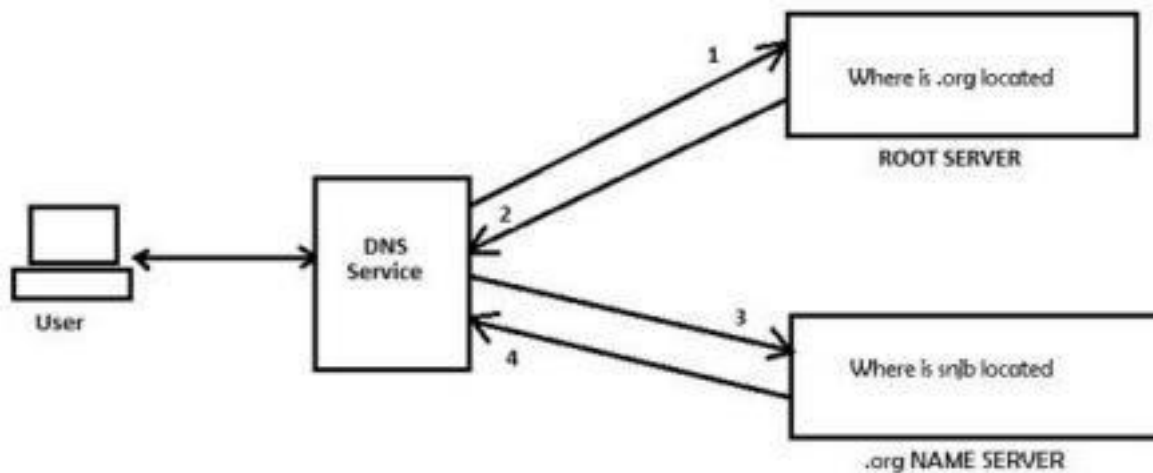
A contacts Computer 11

A contacts B

A contacts printer attached to B

A similar method is used to lookup DNS records. For example, when you click on <http://snjb.org>, your router will contact your default DNS Service for DNS resolution. The DNS service will contact Root Servers and ask for the IP address of server containing **.org** records. This address is sent back to your DNS service. The DNS service again reaches the Name Server containing addresses of **.org** domains and asks it for the address of <http://snjb.org>. Upon obtaining the IP address of the servers that host snjb.org, your DNS service will return the IP address to your computer which then fires up your browser to download the main webpage. This means your DNS service is sending at least two requests to receive the IP address of a simple domain name.

**Following is an image that explains how DNS lookup works:**



### Understanding How DNS Lookup Works

In the above case, if you were to look for <http://snjb.academiaerp.com/snjb/Login>, your DNS service had to run a request extra to know its IP address.

Since resolving DNS from scratch every time takes up time, many ISPs and DNS Service Providers create local caches that contain already resolved addresses. These are primarily the addresses they already fetched from Root Servers and other Name Servers at some point of time. In this case, when you send a request for a URL, instead of contacting the Root server directly, the DNS service would look up for the resolved address of the URL in its local DNS cache. If found, it would send the resolution back to your computer instantly else would go ahead and resolve the DNS using the above method of contacting Root Servers and other Name Servers.

Some operating systems too contain a local cached copy of addresses that you commonly use on your computer. This too, helps in saving time while using the Internet. We will talk about DNS caches in a different article at some later point of time.

**Conclusion:** Hence we conclude that we have lookup the URL which we want to visit the request is travels to local router to DNS server and it resolve the query as possible otherwise it forwards the query to next DNS hop.