

PVGCOE & SSDIOM

Student MANUAL

Computer network Lab

Course	317524 (2019)
Course Name	OOP and COmPuter graphics Lab
Class	t E AIDS –SEM I
Examination Scheme	TW:25 Practical:25
Credits	01

PVGCOE & SSDIOM

INDEX

Sr. No	Title of Experiment	CO
	GROUP – A	
1	Demonstrate the different types of topologies and types of transmission media by using a packet tracer tool.	CO1, CO4
2	Setup a WAN which contains wired as well as wireless LAN by using a packet tracer tool. Demonstrate transfer of a packet from LAN 1 (wired LAN) to LAN2 (Wireless LAN).	
3	Use packet Tracer tool for configuration of 3 router networks using one of the following protocols RIP/ OSPF/BGP.	CO1, CO4
4	Write a program to demonstrate Sub-netting and find subnet masks.	CO2
	GROUP – B	
5	Socket Programming using C/C++/Java. a. TCP Client, TCP Server b. UDP Client, UDP Server	CO2, CO3
6	Write a program using TCP socket for wired network for following a. Say Hello to Each other b. File transfer	CO2, CO3
7	. Write a program using UDP Sockets to enable file transfer (Script, Text, Audio and Video one file each) between two machines.	CO2, CO3
8	Study and analyze the performance of HTTP, HTTPS and FTP protocol using Packet tracer tool.	CO3, CO4
9	To study the IPsec (ESP and AH) protocol by capturing the packets using Wireshark tool.	CO1, CO4

10	. Illustrate the steps for implementation of S/MIME email security, POP3 through Microsoft® Office Outlook	CO3
	GROUP-C	
11	Installing and configuring DHCP server and assign IP addresses to client machines using DHCP server.	CO3
12	Write a program for DNS lookup. Given an IP address input, it should return URL and vice versa.	CO3

GROUP- A

Assignment No.: 1

TITLE: Implementation of different types of topologies and types of transmission media by using a packet tracer tool.

OBJECTIVES:

1. To learn different types of topologies and transmission media
2. To learn packet tracer tool

PROBLEM STATEMENT:

Demonstrate the different types of topologies and types of transmission media by using a packet tracer tool

OUTCOME:

Students will be able to,

1. get familiar with packet tracer tool
2. implement different topologies using packet tracer tool
3. understand different transmission media

THEORY-CONCEPT:

Network topology is the geometric representation of relationship of all the links connecting the devices or nodes. Network topology represent in two ways one is physical topology that define the way in which a network is physically laid out and other one is logical topology that defines how data actually flow through the network. Cisco Packet Tracer (CPT) is multi-tasking network simulation software to perform and analyze various network activities such as implementation of different topologies, select optimum path based on various routing algorithms, create DNS and DHCP server, sub netting, analyze various network configuration and troubleshooting commands. In order to start communication between end user devices and to design a network, we need to select appropriate networking devices like routers, switches, hubs and make physical Connection by connection cables to serial and fast Ethernet ports from the

component list of packet tracer. Networking devices are costly so it is better to perform first on packet tracer to understand the concept and behavior of networking.

DESIGNING OF TOPOLOGY

1. Bus Topology

In local area network, it is a single network cable runs in the building or campus and all nodes are connected along with this communication line with two end point scaled the bus or backbone. In other words, it is a multipoint data communication circuit that is easily control data flow between the computers because this configuration allows all stations to receive every transmission over the network. For bus topology we build network using three generic pc which are serially connected with three switches using copper straight through cable and switches are interconnected using copper cross over cable shown in fig1.

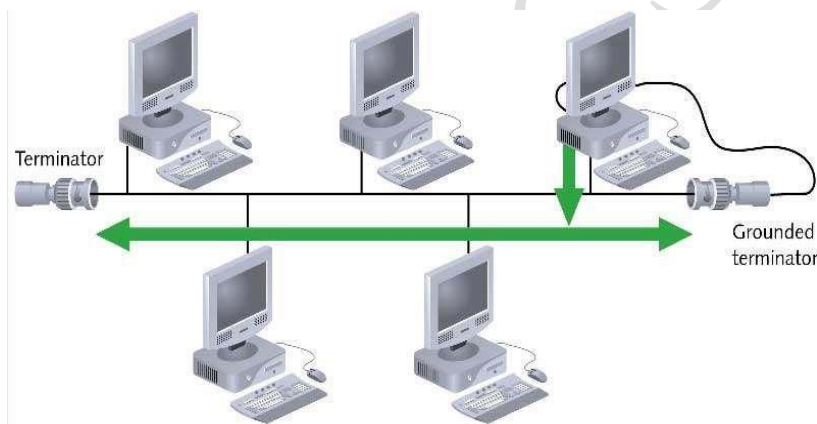


Fig:-- BUS TOPOLOGY

2. Star Topology

All the cables run from the computers to a central location where they are all connected by a device called a hub. It is a concentrated network, where the end points are directly reachable from a central location when network is expanded. Ethernet 10 base T is a popular network based on the star topology. For star topology we build network using five generic pc which are centrally connected to single switch 2950-24 using copper straight through cable shown in fig2.

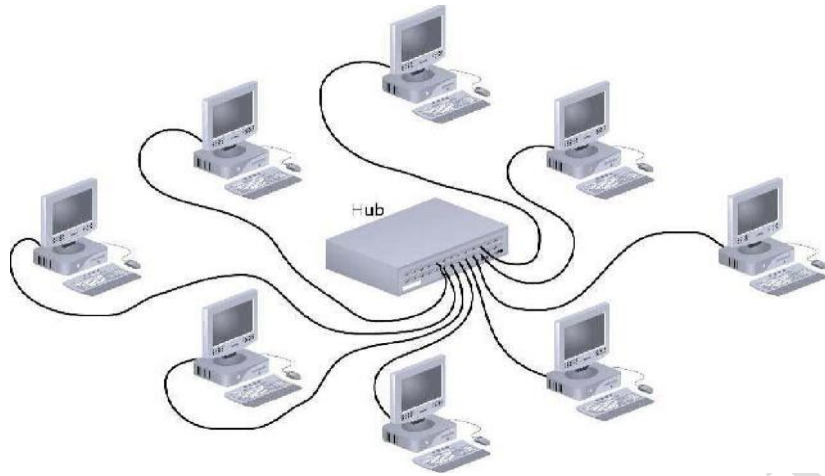


Fig 2: - Star Topology

3. Mesh Topology

Every device has a dedicated point to point link to every other device. The term dedicated stand for link carries traffic only between two devices it connects. It is a well connected topology; As shown in fig3. , in this every node has a connection to every other node in the network. The cable requirements are high and it can include multiple topologies. Failure in one of the computers does not cause the network to break down, as they have alternative paths to other computers star topology, all the cables run from the computers to a central location.

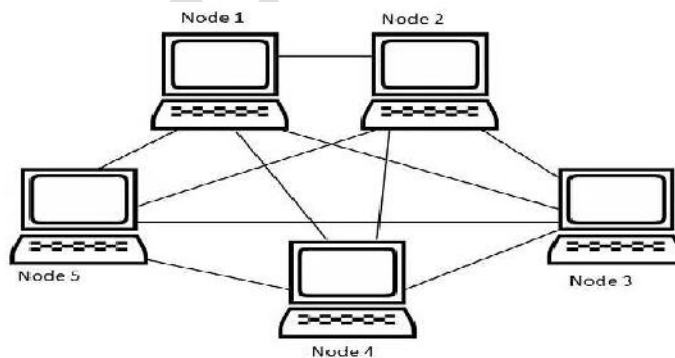


Fig 3:- Mesh Topology

TRANSMISSION MEDIA

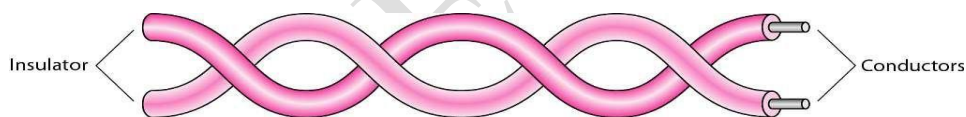
Transmission media is a communication channel that carries the information from the sender to the receiver. Data is transmitted through the electromagnetic signals. The main functionality of the transmission media is to carry the information in the form of bits through LAN (Local Area Network). It is a physical path between transmitter and receiver in data

communication. In a copper-based network, the bits in the form of electrical signals. In a fiber-based network, the bits in the form of light pulses. In OSI(Open System Interconnection) phase, transmission media supports the Layer 1. Therefore, it is considered to be as a Layer 1 component. The electrical signals can be sent through the copper wire, fiber optics, atmosphere, water, and vacuum. The characteristics and quality of data transmission are determined by the characteristics of medium and signal. Transmission media is of two types are wired media and wireless media. In wired media, medium characteristics are more important whereas, in wireless media, signal characteristics are more important. Different transmission media have different properties such as bandwidth, delay, cost and ease of installation and maintenance. The transmission media is available in the lowest layer of the OSI reference model, i.e., Physical layer. Transmission Media is broadly classified into the following types:

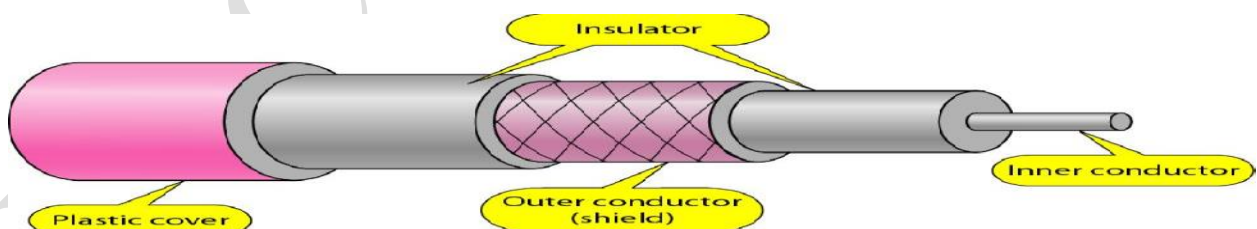
1. Guided
2. Unguided

1. Guided Transmission media: Transmission capacity depends on the distance and on whether the medium is point-to-point or multipoint. There are three types of guided transmission media.

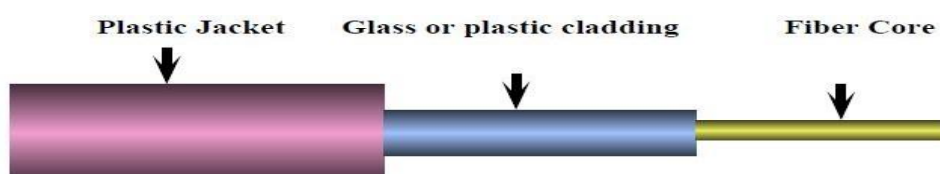
a) Twisted pair wires



b) Coaxial cables



c) Optical fiber



2. Unguided transmission Media

It is also referred to as Wireless or Unbounded transmission media. No physical medium is required for the transmission of electromagnetic signals.

Features:

- The signal is broadcasted through air
- Less Secure
- Used for larger distances

There are 3 types of Signals transmitted through unguided media:

Radio waves – These are easy to generate and can penetrate through buildings. The sending and receiving antennas need not be aligned. Frequency Range: 3KHz – 1GHz. AM and FM radios and cordless phones use Radio waves for transmission.

Microwaves – It is a line-of-sight transmission i.e., the sending and receiving antennas need to be properly aligned with each other. The distance covered by the signal is directly proportional to the height of the antenna. Frequency Range: 1GHz – 300GHz. These are majorly used for mobile phone communication and television distribution.

Infrared – Infrared waves are used for very short distance communication. They cannot penetrate through obstacles. This prevents interference between systems. Frequency Range: 300GHz – 400THz. It is used in TV remotes, wireless mouse, keyboard, printer, etc.

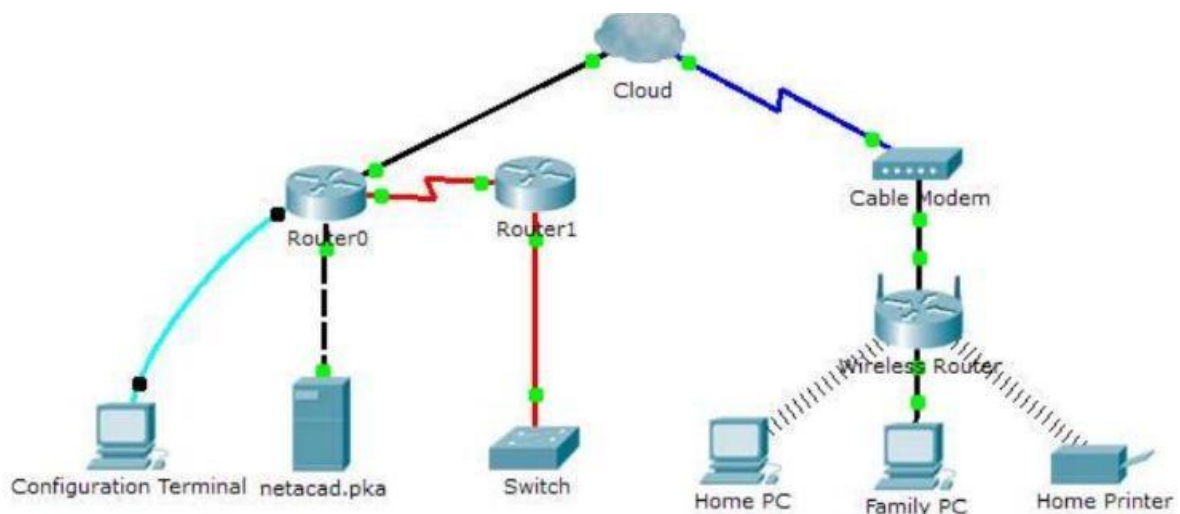
CONCLUSION: We have learned how to implement different topologies using different transmission media in packet tracer tool.

Assignment No.: 2

TITLE:-- Setup a WAN which contains wired as well as wireless LAN by using a packet tracer tool. Demonstrate transfer of a packet from LAN 1 (wired LAN) to LAN2 (Wireless LAN).

OBJECTIVES:-

- 1 Understand the setup of wired & wireless Network
2. Communication establishment between two different types of network.



Addressing Table

Device	Interface	IP Address	Connects To
Cloud	Eth6	N/A	F0/0
	Coax7	N/A	Port0
Cable Modem	Port0	N/A	Coax7
	Port1	N/A	Internet
Router0	Console	N/A	RS232
	F0/0	192.168.2.1/24	Eth6
	F0/1	10.0.0.1/24	F0
	Ser0/0/0	172.31.0.1/24	Ser0/0

Device	Interface	IP Address	Connects To
Router1	Ser0/0	172.31.0.2/24	Ser0/0/0
	F1/0	172.16.0.1/24	F0/1
WirelessRouter	Internet	192.168.2.2/24	Port 1
	Eth1	192.168.1.1	F0
Family PC	F0	192.168.1.102	Eth1
Switch	F0/1	172.16.0.2	F1/0
Netacad.pka	F0	10.0.0.254	F0/1
Configuration Terminal	RS232	N/A	Console

- Part 1: Connect to the Cloud
- Part 2: Connect Router0
- Part 3: Connect Remaining Devices
- Part 4: Verify Connections
- Part 5: Examine the Physical Topology

Background

- When working in Packet Tracer (a lab environment or a corporate setting), you should know how to select the appropriate cable and how to properly connect devices. This activity will examine device configurations in Packet Tracer, selecting the proper cable based on the configuration, and connecting the devices. This activity will also explore the physical view of the network in Packet Tracer.

Instructions

Part 1: Connect to the Cloud

Step 1: Connect the cloud to Router0.

- At the bottom left, click the orange lightning icon to open the available Connections.
- Choose the correct cable to connect Router0 F0/0 to Cloud Eth6. Cloud is a type of switch, so use a Copper Straight-Through connection. If you attached the correct cable, the link lights on the cable turn green.

Step 2: Connect the cloud to Cable Modem.

- Choose the correct cable to connect Cloud Coax7 to Modem Port0.
- If you attached the correct cable, the link lights on the cable turn green.

Part 2: ConnectRouter0

Step 1: Connect Router0 to Router1.

- Choose the correct cable to connect Router0 Ser0/0/0 to Router1 Ser0/0. Use one of the available Serial cables.
- If you attached the correct cable, the link lights on the cable turn green.

Step 2: Connect Router0 to netacad.pka.

- Choose the correct cable to connect Router0 F0/1 to netacad.pkaF0. Routers and computers traditionally use the same wires to transmit (1 and 2) and receive (3 and 6). The correct cable to choose consists of these crossed wires. Although many NICs can now autosense which pair is used to transmit and receive, Router0 and netacad.pkado not have autosensing NICs.
- If you attached the correct cable, the link lights on the cable turn green.

Step 3: Connect Router0 to the Configuration Terminal.

- Choose the correct cable to connect Router0Console to Configuration Terminal RS232. This cable does not provide network access to Configuration Terminal, but allows you to configure Router0 through its terminal.
- If you attached the correct cable, the link lights on the cable turn black.

Part 3: Connect Remaining Devices

Step 1: Connect Router1 to Switch.

- Choose the correct cable to connect Router1 F1/0 to Switch F0/1.
- If you attached the correct cable, the link lights on the cable turn green. Allow a few seconds for the light to transition from amber to green.

Step 2: Connect Cable Modem to Wireless Router.

- Choose the correct cable to connect Cable ModemPort1 to Wireless RouterInternet port.

- If you attached the correct cable, the link lights on the cable will turn green.

Step 3: Connect Wireless Router to Family PC.

- Choose the correct cable to connect Wireless Router Ethernet 1 to Family PC.
- If you attached the correct cable, the link lights on the cable turn green.

Part 4: Verify Connections

Step 1: Test the connection from Family PC to netacad.pka.

- Open the Family PC command prompt and ping netacad.pka.
- Open the Web Browser and the web address <http://netacad.pka>.

Step 2: Ping the Switch from Home PC.

- Open the Home PC command prompt and ping the Switch IP address of to verify the connection.

Step 3: Open Router0 from Configuration Terminal.

- Open the Terminal of Configuration Terminal and accept the default settings.
- Press Enter to view the Router0 command prompt.
- Type show ip interface brief to view interface statuses.

Part 5: Examine the Physical Topology

Step 1: Examine the Cloud.

- Click the Physical Workspace tab or press Shift+P and Shift+L to toggle between the logical and physical workspaces.
- Click the Home City icon.
- Click the Cloud icon.

How many wires are connected to the switch in the blue rack?

- Click Back to return to Home City.

Step 2: Examine the Primary Network.

- Click the Primary Network icon. Hold the mouse pointer over the various cables.

What is located on the table to the right of the blue rack?

Configuration Terminal

- b. Click Back to return to Home City.

Step 3: Examine the Secondary Network.

- a. Click the Secondary Network icon. Hold the mouse pointer over the various cables.

Why are there two orange cables connected to each device?

Fiber cables come in pairs, one for transmit, the other for receive.

- b. Click Back to return to Home City.

Step 4: Examine the Home Network.

- a. Click the Home Network icon.

Why is there no rack to hold the equipment?

Home networks typically do not have racks.

- b. Click the Logical Workspace tab to return to the logical topology.

CONCLUSION: We have learned to configure wire and wireless network and its communication

Assignment No: 3

TITLE: Configuration of 3 router networks using RIP in Packet Tracer

OBJECTIVES:

1. To configure router.
2. To understand routing protocols.

PROBLEM STATEMENT: Use packet Tracer tool for configuration of 3 router networks using one of the following protocols RIP/OSPF/BGP.

OUTCOME: Students will be able to,

1. Configure router networks using Packet Tracer.
2. Understand routing protocols.

THEORY-CONCEPT:

Routing Information Protocol (RIP):

The Routing Information Protocol (RIP) is one of the oldest distance-vector routing protocols which employs the hop count as a routing metric. RIP prevents routing loops by implementing a limit on the number of hops allowed in a path from source to destination. The largest number of hops allowed for RIP is 15, which limits the size of networks that RIP can support.

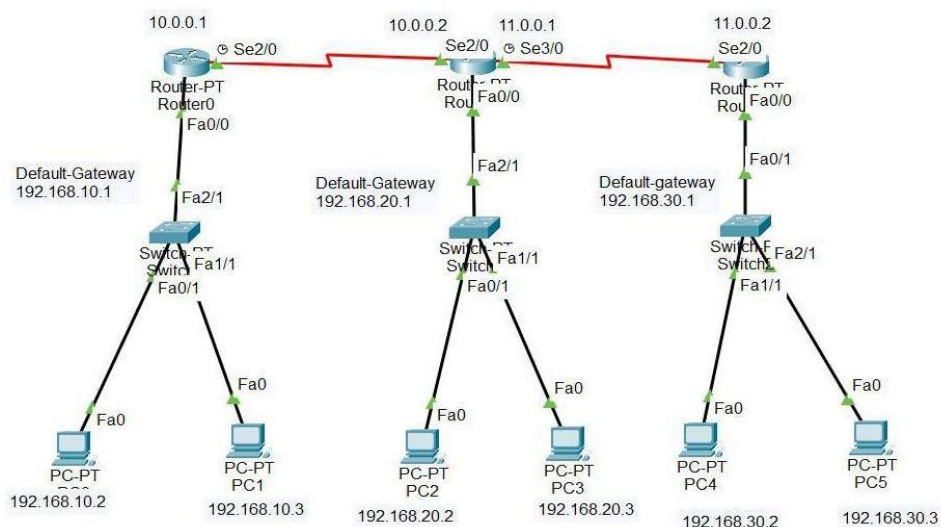
Step 1: First, open the Cisco packet tracer desktop and select the devices given below:

S.NO	Device	Model Name	Qty.
1.	PC	PC	6
2.	Switch	PT-Switch	3
3.	Router	PT-router	3

IP Addressing Table:

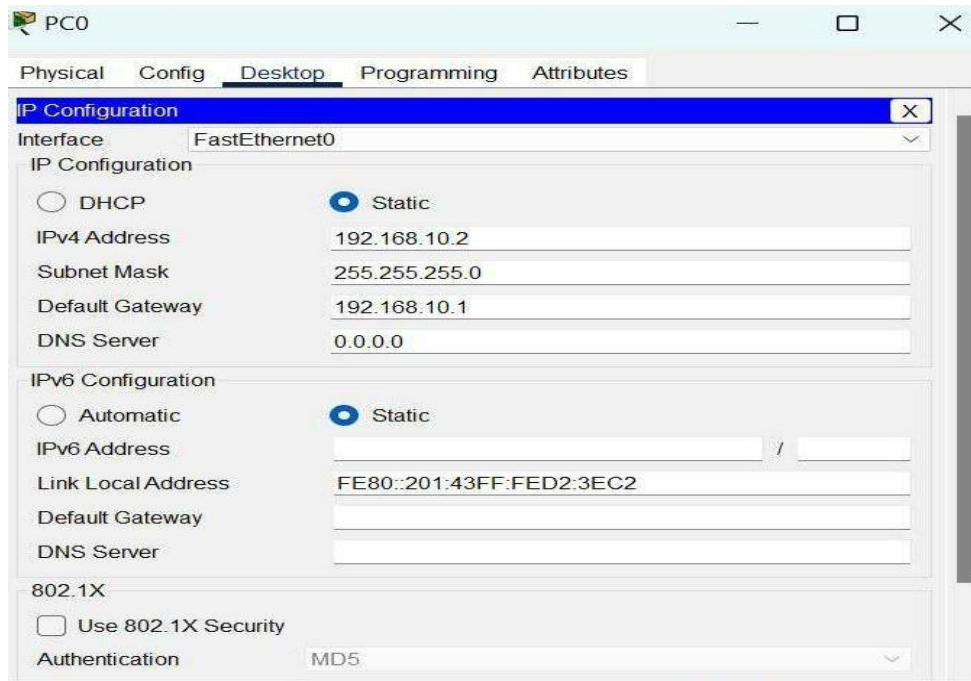
S.NO	Device	IPv4 Address	Subnet mask	Default Gateway
1.	PC0	192.168.10.2	255.255.255.0	192.168.10.1
2.	PC1	192.168.10.3	255.255.255.0	192.168.10.1
3.	PC2	192.168.20.2	255.255.255.0	192.168.20.1
4.	PC3	192.168.20.3	255.255.255.0	192.168.20.1
5.	PC4	192.168.30.2	255.255.255.0	192.168.30.1
6.	PC5	192.168.30.3	255.255.255.0	192.168.30.1

- Then, create a network topology as shown below the image.
- Use an Automatic connecting cable to connect the devices with others.

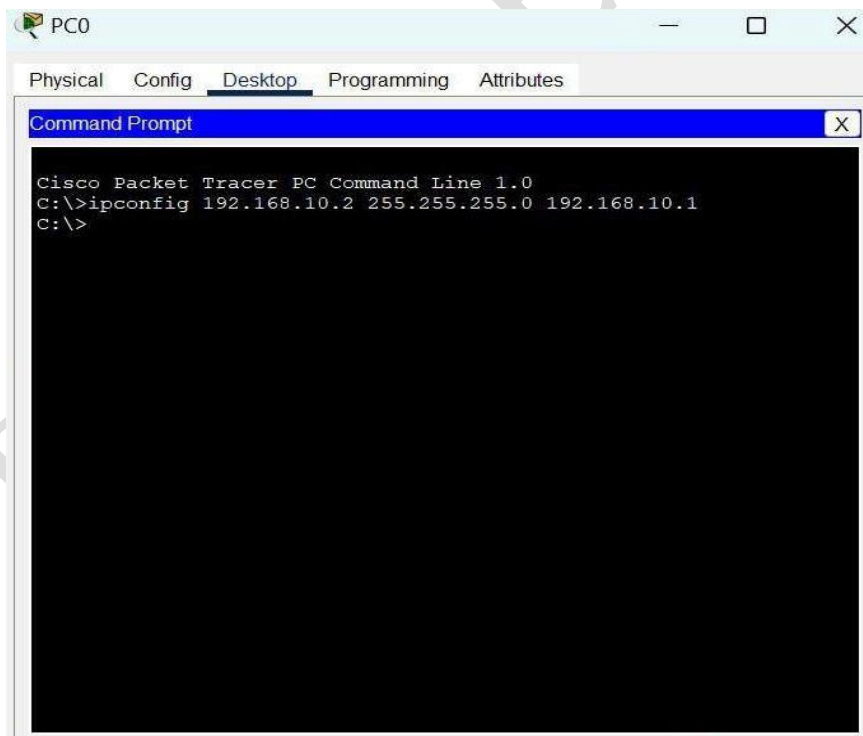


Step 2: Configure the PCs (hosts) with IPv4 address and Subnet Mask according to the IP

- addressing table given above.
- To assign an IP address in PC0, click on PC0.
- Then, go to desktop and then IP configuration and there you will IPv4 configuration.
- Fill IPv4 address and subnet mask.



- Assigning an IP address using the ipconfig command, or we can also assign an IP address with the help of a command.
- Go to the command terminal of the PC.
- Then, type ipconfig <IPv4 address><subnet mask><default gateway>(if needed)
- Example: ipconfig 192.168.10.2 255.255.255.0 192.168.10.1



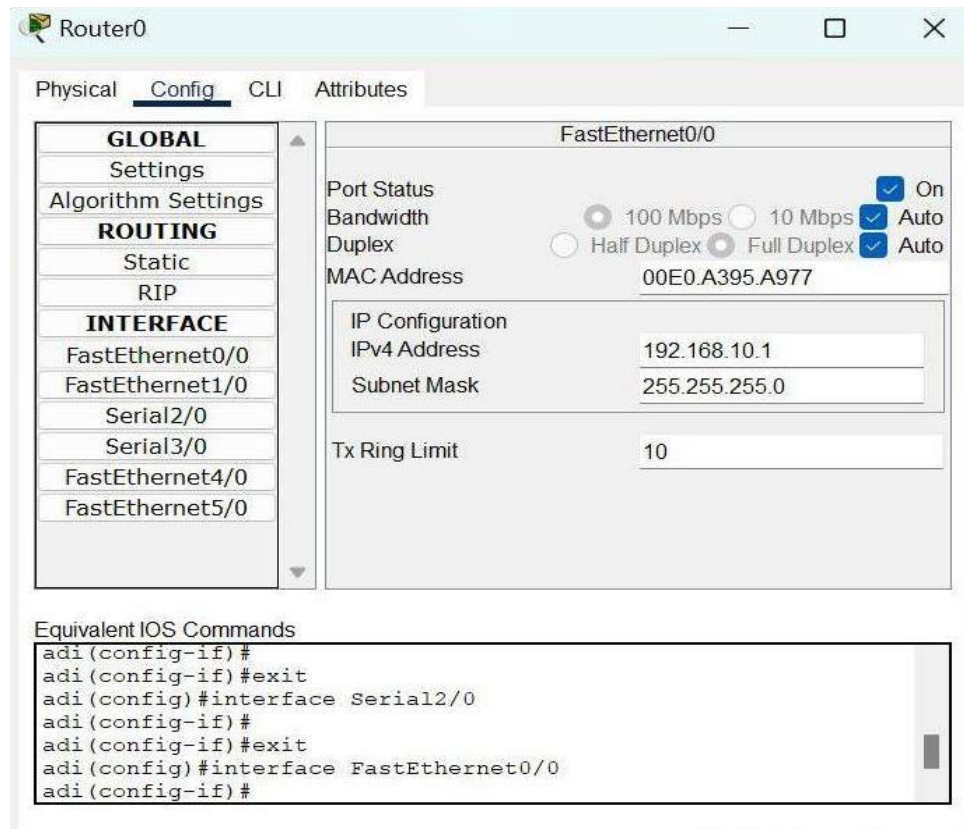
Repeat the same procedure with other PCs to configure them thoroughly.

Step 3: Configure router with IP address and Subnet mask.

IP Addressing Table Router:

S.NO	Device	Interface	IPv4 Address	Subnet mask
1.	router0	FastEthernet0/0	192.168.10.1	255.255.255.0
		Serial2/0	10.0.0.1	255.0.0.0
2.	router1	FastEthernet0/0	192.168.20.1	255.255.255.0
		Serial2/0	10.0.0.2	255.0.0.0
		Serial3/0	11.0.0.1	255.0.0.0
3.	router2	FastEthernet0/0	192.168.30.1	255.255.255.0
		Serial2/0	11.0.0.2	255.0.0.0

- To assign an IP address in router0, click on router0.
- Then, go to config and then Interfaces.
- Make sure to turn on the ports.
- Then, configure the IP address in FastEthernet and serial ports according to IP addressing Table.
- Fill IPv4 address and subnet mask.



- Repeat the same procedure with other routers to configure them thoroughly.

Step 4: After configuring all of the devices we need to assign the routes to the routers.

To assign RIP routes to the particular router:

- First, click on router0 then Go to CLI.
- Then type the commands and IP information given below.

CLI command : network <network id>

RIP Routes for Router0 are given below:

- Router(config)#network 192.168.10.0
- Router(config)#network 10.0.0.0

RIP Routes for Router1 are given below:

Router(config)#network 192.168.20.0

Router(config)#network 10.0.0.0

Router(config)#network 11.0.0.0

RIP Routes for Router2 are given below:

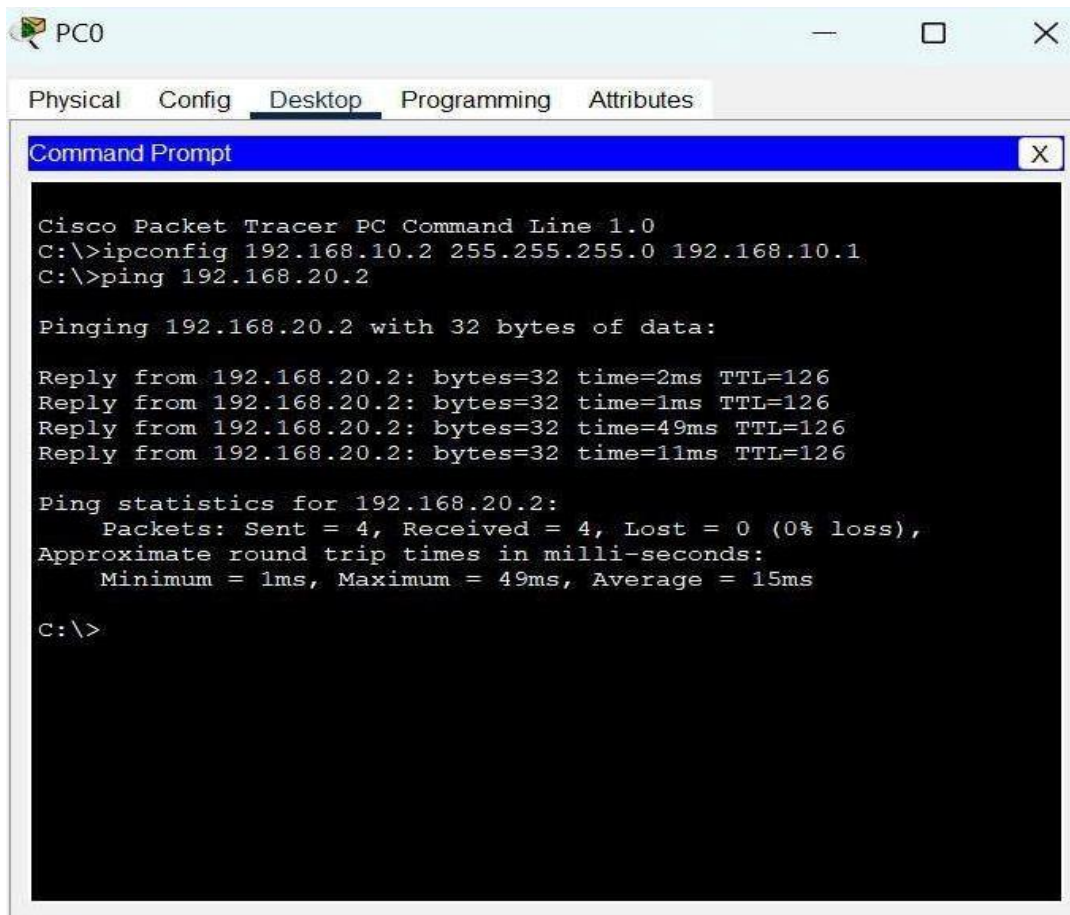
Router(config)#network 192.168.30.0

Router(config)#network 11.0.0.0

Step 5: Verifying the network by pinging the IP address of any PC.

- We will use the ping command to do so.
- First, click on PC0 then Go to the command prompt.
- Then type ping <IP address of targeted node>.
- As we can see in the below image we are getting replies which means the connection is
- working properly.

Example: ping 192.168.20.2



```
PC0
Physical Config Desktop Programming Attributes
Command Prompt
Cisco Packet Tracer PC Command Line 1.0
C:\>ipconfig 192.168.10.2 255.255.255.0 192.168.10.1
C:\>ping 192.168.20.2

Pinging 192.168.20.2 with 32 bytes of data:

Reply from 192.168.20.2: bytes=32 time=2ms TTL=126
Reply from 192.168.20.2: bytes=32 time=1ms TTL=126
Reply from 192.168.20.2: bytes=32 time=49ms TTL=126
Reply from 192.168.20.2: bytes=32 time=11ms TTL=126

Ping statistics for 192.168.20.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 1ms, Maximum = 49ms, Average = 15ms

C:\>
```

CONCLUSION: We have learned to configure 3 router networks using RIP in Cisco Packet Tracer tool.

Assignment No: 4

TITLE: Implement subnetting and find the subnet masks.

OBJECTIVES:

1. To learn network address classes
2. To understand the concept of subnetting

PROBLEM STATEMENT: Write a program to demonstrate Sub-netting and find subnet masks.

OUTCOME: Students will be able to

1. memorize network address classes
2. implement subnetting and subnet masks

THEORY-CONCEPT:

SUBNETTING

Subnetting is a process of dividing large network into the smaller networks based on layer 3 IP address. Every computer on network has an IP address that represents its location on network. Two versions of IP addresses are available IPv4 and IPv6. In this article we will perform subnetting on IPv4.

IPv4

IP addresses are displayed in dotted decimal notation, and appear as four numbers separated by dots. Each number of an IP address is made from eight individual bits known as an octet. Each octet can create a number value from 0 to 255. An IP address would be 32 bits long in binary, divided into the two components, network component and host component. Network component is used to identify the network that the packet is intended for, and host component is used to identify the individual host on network. IP addresses are broken into the two components:

Network component: - Defines network segment of device.

Host component: - Defines the specific device on a particular network segment

IP Classes in decimal notation

1. Class A addresses range from 1-126
2. Class B addresses range from 128-191
3. Class C addresses range from 192-223
4. Class D addresses range from 224-239
5. Class E addresses range from 240-254

- 0 [Zero] is reserved and represents all IP addresses.
- 127 is a reserved address and is used for testing, like a loop back on an interface.
- 255 is a reserved address and is used for broadcasting purposes

SUBNET MASK

Subnet mask is a 32 bits long address used to distinguish between network address and host address in IP address. Subnet mask is always used with IP address. Subnet mask has only one purpose, to identify which part of an IP address is network address and which part is host address.

For example, how will we figure out network partition and host partition from IP address 192.168.1.10. Here we need subnet mask to get details about network address and host address.

- In decimal notation subnet mask value 1 to 255 represent network address and value 0 [Zero] represent host address.
- In binary notation subnet mask **ON** bit [1] represent network address while **OFF** bit [0] represent host address.

In decimal notation

IP address 192.168.1.10

Subnet mask 255.255.255.0

Network address is 192.168.1 and host address is 10.

In binary notation

IP address 11000000.10101000.00000001.00001010

Subnet mask 11111111.11111111.11111111.00000000

Network address is 11000000.10101000.00000001 and host address is 00001010

The following table shows IP class associated with default subnet

IP	Default Subnet	Network bits	Host bits	Total hosts	Valid hosts
Class					
A	255.0.0.0	First 8 bits	Last 24 bits	16, 777, 216	16, 777, 214
B	255.255.0.0	First 16 bits	Last 16 bits	65,536	65,534
C	255.255.255.0	First 24 bits	Last 8 bits	256	254

Network ID

First address of subnet is called network ID. This address is used to identify one segment or broadcast domain from all the other segments in the network.

Block Size

Block size is the size of subnet including network address, hosts addresses and broadcast address.

Broadcast ID

There are two types of broadcast, direct broadcast and full broadcast.

Direct broadcast or local broadcast is the last address of subnet and can be heard by all hosts in subnet.

Full broadcast is the last address of IP classes and can be heard by all IP hosts in network. Full broadcast address is 255.255.255.255

The main difference between direct broadcast and full broadcast is that routers will not propagate local broadcasts between segments, but they will propagate directed broadcasts.

Host Addresses

All address between the network address and the directed broadcast address is called host address for the subnet. You can assign host addresses to any IP devices such as PCs, servers, routers, and switches.

Single class C IP range can fulfill this requirement, still you have to purchase 2 class C IP range, one for each network. Single class C range provides 256 total addresses and we need only 30 addresses, this will waste 226 addresses. These unused addresses would make additional route advertisements slowing down the network.

With subnetting you only need to purchase single range of class C. You can configure router to take first 26 bits instead of default 24 bits as network bits. In this case we would extend default boundary of subnet mask and borrow 2 host bits to create networks. By taking two bits from the host range and counting them as network bits, we can create two new subnets, and assign hosts them. As long as the two new network bits match in the address, they belong to the same network. You can change either of the two bits, and you would be in a new subnet.

Default subnet mask

Class	Subnet Mask	Format
A	255.0.0.0	Network.Host.Host.Host
B	255.255.0.0	Network.Network.Host.Host
C	255.255.255.0	Network.Network.Network.Host

ADVANTAGE OF SUBNETTING

- Subnetting breaks large network in smaller networks and smaller networks are easier to manage.
- Subnetting reduces network traffic by removing collision and broadcast traffic, that overall improve performance.
- Subnetting allows you to apply network security policies at the interconnection between

subnets.

- Subnetting allows you to save money by reducing requirement for IP range.

CONCLUSION: We have successfully implemented the subnetting and subnet mask program.

GROUP- B

Assignment No: 5

AIM: Socket Programming using C/C++/Java.

OBJECTIVE:

1. To learn socket programming.
2. To classify client server communication
3. To learn TCP and UDP.

PROBLEM STATEMENT: Socket Programming using C/C++/Java.

- a. TCP Client, TCP Server
- b. UDP Client, UDP Server

OUTCOME: Students will be able to

1. apply Socket programming for client server communication.
2. understand client server communication using TCP and UDP

THEORY-CONCEPT

USER DATAGRAM PROTOCOL (UDP)

UDP is a simple transport-layer protocol. The application writes a message to a UDP socket, which is then encapsulated in a UDP datagram, which is further encapsulated in an IP datagram, which is sent to the destination. There is no guarantee that a UDP will reach the destination, that the order of the datagrams will be preserved across the network or that datagrams arrive only once.

The problem of UDP is its lack of reliability: if a datagram reaches its final destination but the checksum detects an error, or if the datagram is dropped in the network, it is not automatically retransmitted. Each UDP datagram is characterized by a length. The length of a datagram is passed to the receiving application along with the data. No connection is established

between the client and the server and, for this reason, we say that UDP provides a connection-less service

UDP SOCKET API

There are some fundamental differences between TCP and UDP sockets. UDP is a connection-less, unreliable, datagram protocol (TCP is instead connection-oriented, reliable and stream based). There are some instances when it makes to use UDP instead of TCP. Some popular applications built around UDP are DNS, NFS, SNMP and for example, some Skype services and streaming media. Figure 1 shows the the interaction between a UDP client and server. First of all, the client does not establish a connection with the server. Instead, the client just sends a datagram to the server using the `sendto` function which requires the address of the destination as a parameter. Similarly, the server does not accept a connection from a client. Instead, the server just calls the `recvfrom` function, which waits until data arrives from some client. `recvfrom` returns the IP address of the client, along with the datagram, so the server can send a response to the client. As shown in the Figure 1, the steps of establishing a UDP socket communication on the client side are as follows:

- Create a socket using the `socket()` function;
- Send and receive data by means of the `recvfrom()` and `sendto()` functions.
- The steps of establishing a UDP socket communication on the server side are as follows:
 - Create a socket with the `socket()` function;
 - Bind the socket to an address using the `bind()` function;
 - Send and receive data by means of `recvfrom()` and `sendto()`.

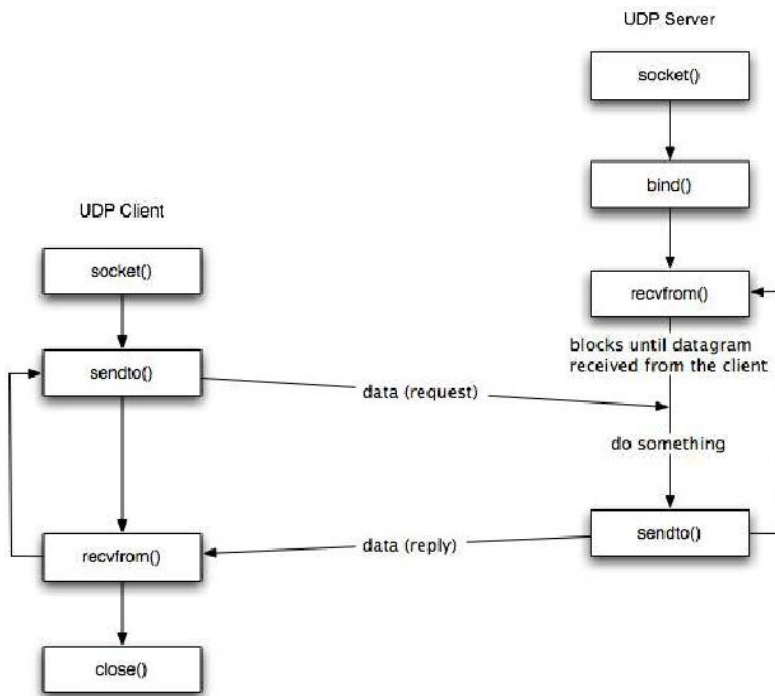


Fig 1 Client Server communication using UDP

TRANSMISSION CONTROL PROTOCOL (TCP)

TCP provides a connection-oriented service, since it is based on connections between clients and servers. TCP provides reliability. When a TCP client send data to the server, it requires an acknowledgement in return. If an acknowledgement is not received, TCP automatically retransmit the data and waits for a longer period of time. We have mentioned that UDP datagrams are characterized by a length. TCP is instead a byte-stream protocol, without any boundaries at all.

TCP SOCKET API

The sequence of function calls for the client and a server participating in a TCP connection is presented in Figure 2

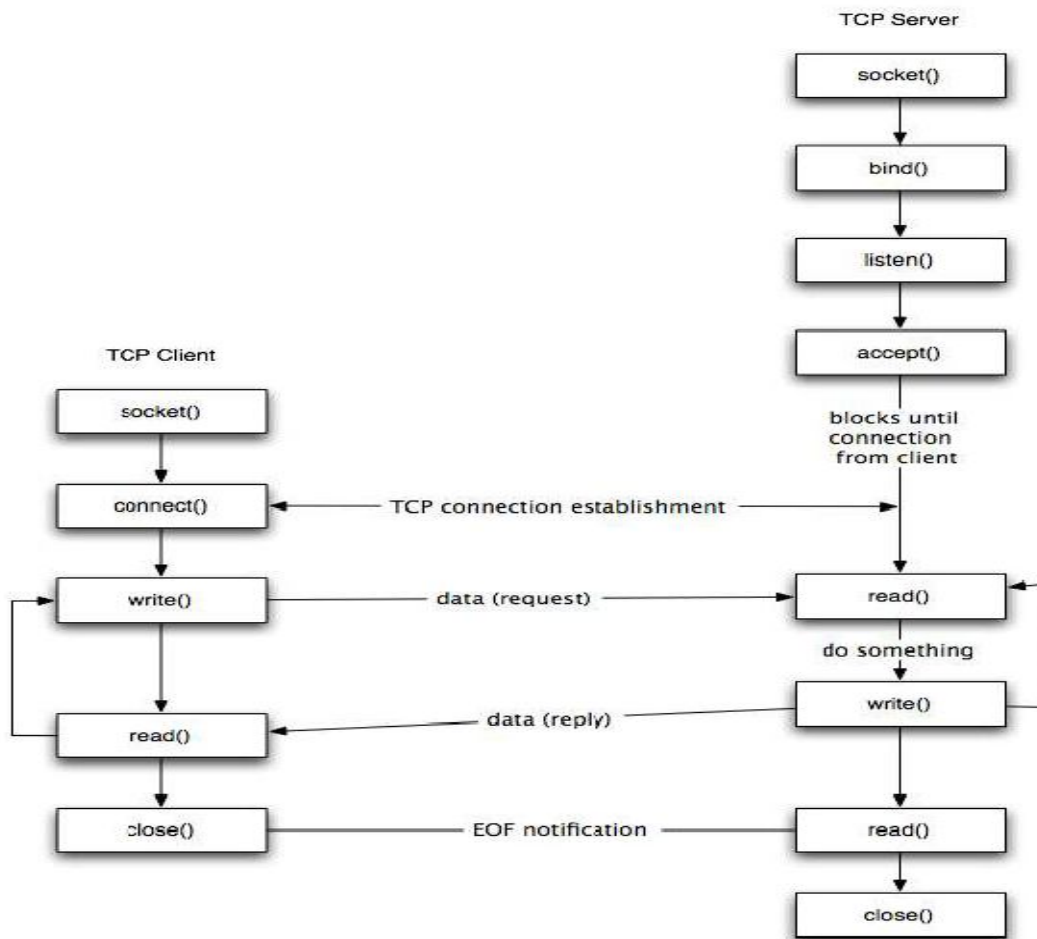


Fig 2 Client Server communication using TCP

CONCLUSION: In this experiment we implemented client server communication using TCP and UDP

Assignment No: 6

AIM: TCP socket for wired network.

OBJECTIVE: 1. To learn TCP socket for wired network

PROBLEM STATEMENT: Write a program using TCP socket for wired network for following

- a. Say Hello to Each other
- b. File transfer

OUTCOME: Students will able to

1. implement socket programming using TCP.

THEORY-CONCEPT:

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.

Socket: Sockets allow communication between two different processes on the same or different machines. To be more precise, it's a way to talk to other computers using standard Unix file descriptors. In Unix, every I/O action is done by writing or reading a file descriptor. A file descriptor is just an integer associated with an open file and it can be a network connection, a text file, a terminal, or something else.

Socket Types

There are four types 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
- **Sequenced Packet Sockets** – They are similar to a stream socket, with the exception that record boundaries are preserved. This interface is provided only as a part of the Network Systems (NS) socket abstraction, and is very important in most serious NS applications. Sequenced-packet sockets allow the user to manipulate the Sequence Packet Protocol

(SPP) or Internet Datagram Protocol (IDP) headers on a packet or a group of packets, either by writing a prototype header along with whatever data is to be sent, or by specifying a default header to be used with all outgoing data, and allows the user to receive the headers on incoming packets.

Stages for server

1. Socket creation:

```
int sockfd = socket(domain, type, protocol)
```

sockfd: socket descriptor, an integer (like a file-handle)

domain: integer, communication domain

e.g., AF_INET (IPv4 protocol) , AF_INET6 (IPv6 protocol)

type: communication type

SOCK_STREAM: TCP(reliable, connection oriented)

SOCK_DGRAM: UDP(unreliable, connectionless)

protocol: Protocol value for Internet Protocol(IP), which is 0. This is the same number which appears on protocol field in the IP header of a packet.(man protocols for more details)

2. Bind:

```
int bind(int sockfd, const struct sockaddr *addr, socklen_t addrlen);
```

After creation of the socket, bind function binds the socket to the address and port number

specified in addr(custom data structure). In the example code, we bind the server to the localhost, hence we use INADDR_ANY to specify the IP address

3. Listen:

```
int listen(int sockfd, int backlog);
```

It puts the server socket in a passive mode, where it waits for the client to approach the server to make a connection. The backlog, defines the maximum length to which the queue of pending connections for sockfd may grow. If a connection request arrives when the queue is full, the client may receive an error with an indication of ECONNREFUSED.

4. Accept:

```
int new_socket= accept(int sockfd, struct sockaddr *addr, socklen_t *addrlen);
```

It extracts the first connection request on the queue of pending connections for the listening socket, sockfd, creates a new connected socket, and returns a new file descriptor referring to that socket. At this point, connection is established between client and server, and they are ready to transfer data.

Stages for Client

1. Socket connection: Exactly same as that of server's socket creation
2. Connect:

```
int connect(int sockfd, const struct sockaddr *addr, socklen_t addrlen);
```

The connect() system call connects the socket referred to by the file descriptor sockfd to the address specified by addr. Server's address and port is specified in addr.

CONCLUSION:

We have successfully implemented the TCP socket programming

Assignment No:7

AIM: UDP Sockets to enable file transfer

OBJECTIVE: 1. To learn how to transfer file using socket programming.

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

OUTCOME: Students will able to

1. implement socket programming using UDP.

THEORY-CONCEPT:

A transport layer protocol usually has several responsibilities. One is to create a process-to-process communication; UDP uses port numbers to accomplish this. Another responsibility is to provide control mechanisms at the transport level. UDP does this task at a very minimal level. There is no flow control mechanism and there is no acknowledgment for received packets. UDP, however, does provide error control to some extent. If UDP detects an error in the received packet, it silently drops it. UDP is a connectionless, unreliable transport protocol. It does not add anything to the services of IP except for providing process-to-process communication instead of host-to-host communication. UDP packets, called user datagrams, have a fixed-size header of 8 bytes.

- Source port number. This is the port number used by the process running on the source host. It is 16 bits long, which means that the port number can range from 0 to 65,535. If the source host is the client (a client sending a request), the port number, in most cases, is an ephemeral port number requested by the process and chosen by the UDP software running on the source host. If the source host is the server (a server sending a response), the port number, in most cases, is a well-known port number
- Destination port number. This is the port number used by the process running on the destination host. It is also 16 bits long. If the destination host is the server (a client sending a request), the port number, in most cases, is a well-known port number. If the

destination host is the client (a server sending a response), the port number, in most cases, is an ephemeral port number. In this case, the server copies the ephemeral port number it has received in the request packet.

- **Length.** This is a 16-bit field that defines the total length of the user datagram, header plus data. The 16 bits can define a total length of 0 to 65,535 bytes. However, the total length needs to be much less because a UDP user datagram is stored in an IP datagram with the total length of 65,535 bytes. The length field in a UDP user datagram is actually not necessary. A user datagram is encapsulated in an IP datagram. There is a field in the IP datagram that defines the total length. There is another field in the IP datagram that defines the length of the header. So if we subtract the value of the second field from the first, we can deduce the length of the UDP datagram that is encapsulated in an IP datagram. $\text{UDP length} = \text{IP length} - \text{IP header's length}$ However, the designers of the UDP protocol felt that it was more efficient for the destination UDP to calculate the length of the data from the information provided in the UDP user datagram rather than ask the IP software to supply this information. We should remember that when the IP software delivers the UDP user datagram to the UDP layer, it has already dropped the IP header.
- **Checksum.** This field is used to detect errors over the entire user datagram (header plus data). The checksum is discussed in the next section.

UDP SERVICES

UDP provides process-to-process communication using sockets, a combination of IP addresses and port numbers. Several port numbers used by UDP are shown in Table 1.

Table 1: Port numbers used by UDP

Port	Protocol	Description
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
11	Users	Active users
13	Daytime	Returns the date and the time
17	Quote	Returns a quote of the day
19	Chargen	Returns a string of characters
53	Domain	Domain Name Service (DNS)
67	Bootps	Server port to download bootstrap information
68	Bootpc	Client port to download bootstrap information
69	TFTP	Trivial File Transfer Protocol
111	RPC	Remote Procedure Call
123	NTP	Network Time Protocol
161	SNMP	Simple Network Management Protocol
162	SNMP	Simple Network Management Protocol (trap)

Connectionless Services: As mentioned previously, UDP provides a connectionless service.

This means that each user datagram sent by UDP is an independent datagram. There is no relationship between the different user datagrams even if they are coming from the same source process and going to the same destination program. The user datagrams are not numbered. Also, there is no connection establishment and no connection termination as is the case for TCP. This means that each user datagram can travel on a different path.

Flow Control: UDP is a very simple protocol. There is no flow control, and hence no window mechanism. The receiver may overflow with incoming messages. The lack of flow control means that the process using UDP should provide for this service, if needed.

Error Control: There is no error control mechanism in UDP except for the checksum. This means that the sender does not know if a message has been lost or duplicated. When the receiver detects an error through the checksum, the user datagram is silently discarded. The lack of error control means that the process using UDP should provide for this service if needed.

Checksum: UDP checksum calculation is different from the one for IP. Here the checksum includes three sections: a pseudo header, the UDP header, and the data coming from the application layer. The pseudo header is the part of the header of the IP packet in which the user datagram is to be encapsulated with some fields filled with 0s. If the checksum does not include the pseudoheader, a user datagram may arrive safe and sound. However, if the IP header is corrupted, it may be delivered to the wrong host. The protocol field is added to ensure that the packet belongs to UDP, and not to TCP.

Encapsulation: When a process has a message to send through UDP, it passes the message to UDP along with a pair of socket addresses and the length of data. UDP receives the data and adds the UDP header. UDP then passes the user datagram to IP with the socket addresses. IP adds its own header, using the value 17 in the protocol field, indicating that the data has come from the UDP protocol. The IP datagram is then passed to the data link layer. The data link layer receives the IP datagram, adds its own header (and possibly a trailer), and passes it to the physical layer. The physical layer encodes the bits into electrical or optical signals and sends it to the remote machine.

De-encapsulation: When the message arrives at the destination host, the physical layer decodes the signals into bits and passes it to the data link layer. The data link layer uses the header (and the trailer) to check the data. If there is no error, the header and trailer are dropped and the datagram is passed to IP. The IP software does its own checking. If there is no error, the header is dropped and the user datagram is passed to UDP with the sender and receiver IP addresses. UDP uses the checksum to check the entire user datagram. If there is no error, the header is dropped and the application data along with the sender socket address is passed to the process. The sender socket address is passed to the process in case it needs to respond to the message received. When a process has a message to send through UDP, it passes the message to UDP along with a pair of socket addresses and the length of data. UDP receives the data and adds the UDP header. UDP then passes the user datagram to IP with the socket addresses. IP adds its own header, using the value 17 in the protocol field, indicating that the data has come from the UDP protocol. The IP datagram is then passed to the data link layer. The data link layer receives the IP datagram, adds its own header (and possibly a trailer), and passes it to the physical layer. The physical layer encodes the bits into electrical or optical signals and sends it to the remote machine.

Multiplexing:

At the sender site, there may be several processes that need to send user datagrams. However, there is only one UDP. This is a many-to-one relationship and requires multiplexing. UDP accepts messages from different processes, differentiated by their assigned port numbers. After adding the header, UDP passes the user datagram to IP. De-multiplexing: At the receiver site, there is only one UDP. However, we may have many processes that can receive user datagrams. This is a one-to-many relationship and requires demultiplexing. UDP receives user datagrams

from IP. After error checking and dropping of the header, UDP delivers each message to the appropriate process based on the port numbers.

UDP APPLICATIONS

The following shows some typical applications that can benefit more from the services of UDP than from those of TCP.

- UDP is suitable for a process that requires simple request-response communication with little concern for flow and error control. It is not usually used for a process such as FTP that needs to send bulk data.
- UDP is suitable for a process with internal flow and error-control mechanisms. For example, the Trivial File Transfer Protocol (TFTP) process includes flow and error control. It can easily use UDP.

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
`DataInputStream is = new DataInputStream (client.getInputStream());`
`DataOutputStream os = new DataOutputStream(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 DataInputStream (client.getInputStream());`


```
os = new DataOutputStream(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()

CONCLUSION

Thus, we have successfully implemented UDP socket for file transfer between two machines .

Assignment No:8

AIM: Study and Analyze the performance of HTTP, HTTPS and FTP

OBJECTIVE: 1. Learn HTTP, HTTPS and FTP

PROBLEM STATEMENT: Study and Analyze the performance of HTTP, HTTPS and FTP protocol using Packet tracer tool.

OUTCOME: Students will able to

1. analyze the performance of HTTP, HTTPS and FTP using packet tracer tool.

THEORY-CONCEPT:

HTTP:

The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. This is the foundation for data communication for the World Wide Web (i.e. internet) since 1990. HTTP is a generic and stateless protocol which can be used for other purposes as well using extensions of its request methods, error codes, and headers. Basically, HTTP is a TCP/IP based communication protocol, that is used to deliver data (HTML files, image files, query results, etc.) on the World Wide Web. The default port is TCP 80, but other ports can be used as well. It provides a standardized way for computers to communicate with each other. HTTP specification specifies how clients' request data will be constructed and sent to the server, and how the servers respond to these requests.

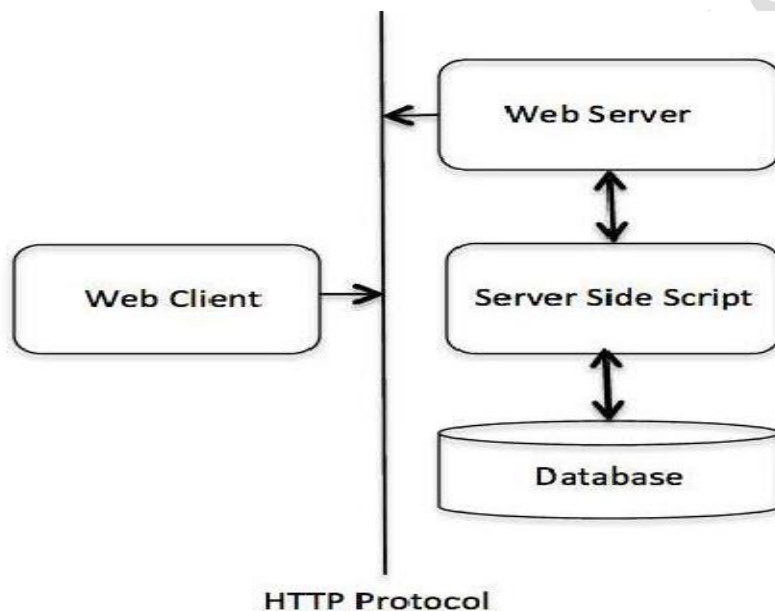
There are three basic features that make HTTP a simple but powerful protocol:

1. **HTTP is connectionless:** The HTTP client, i.e., a browser initiates an HTTP request and after a request is made, the client waits for the response. The server processes the request and sends a response back after which client disconnect the connection. So client and server knows about each other during current request and response only. Further requests are made on new connection like client and server are new to each other.

2. **HTTP is media independent:** It means, any type of data can be sent by HTTP as long as both the client and the server know how to handle the data content. It is required for the client as well as the server to specify the content type using appropriate MIME-type.

3. **HTTP is stateless:** As mentioned above, HTTP is connectionless and it is a direct result of HTTP being a stateless protocol. The server and client are aware of each other only during a current request. Afterwards, both of them forget about each other. Due to this nature of the protocol, neither the client nor the browser can retain information between different requests across the web pages.

The following diagram shows a very basic architecture of a web application and depicts where HTTP sits:



The HTTP protocol is a request/response protocol based on the client/server based architecture where web browsers, robots and search engines, etc. act like HTTP clients, and the Web server acts as a server.

Client

The HTTP client sends a request to the server in the form of a request method, URI, and protocol version, followed by a MIME-like message containing request modifiers, client information, and possible body content over a TCP/IP connection.

Server

The HTTP server responds with a status line, including the message's protocol version and a success or error code, followed by a MIME-like message containing server information, entity meta information, and possible entity-body content.

HTTPS:

HTTPS is an abbreviation of Hypertext Transfer Protocol Secure. It is a secure extension or version of HTTP. This protocol is mainly used for providing security to the data sent between a website and the web browser. It is widely used on the internet and used for secure communications. This protocol uses the 443 port number for communicating the data. This protocol is also called HTTP over SSL because the HTTPS communication protocols are encrypted using the SSL (Secure Socket Layer). By default, it is supported by various web browsers. Those websites which need login credentials should use the HTTPS protocol for sending the data.

Advantages of HTTPS

- The main advantage of HTTPS is that it provides high security to users.
- Data and information are protected. So, it ensures data protection.
- SSL technology in HTTPS protects the data from third-party or hackers. And this technology builds trust for the users who are using it.
- It helps users by performing banking transactions.

Disadvantages of HTTPS

- The big disadvantage of HTTPS is that users need to purchase the SSL certificate.
- The speed of accessing the website is slow because there are various complexities in communication.
- Users need to update all their internal links.

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.

Let's now do FTP configuration in Packet Tracer:

1. Build the network topology.



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 PT Ftp server
Username:cisco
331- Username ok, need password
Password:
230- Logged in
(passive mode On)
ftp>
```

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:

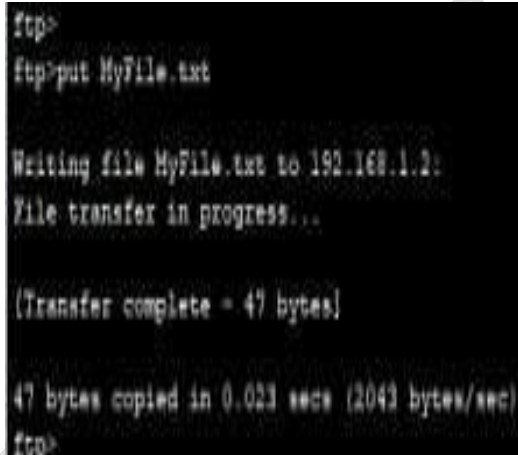
Create a file in the Laptop then **upload** it to the server using **FTP**.

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**.

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



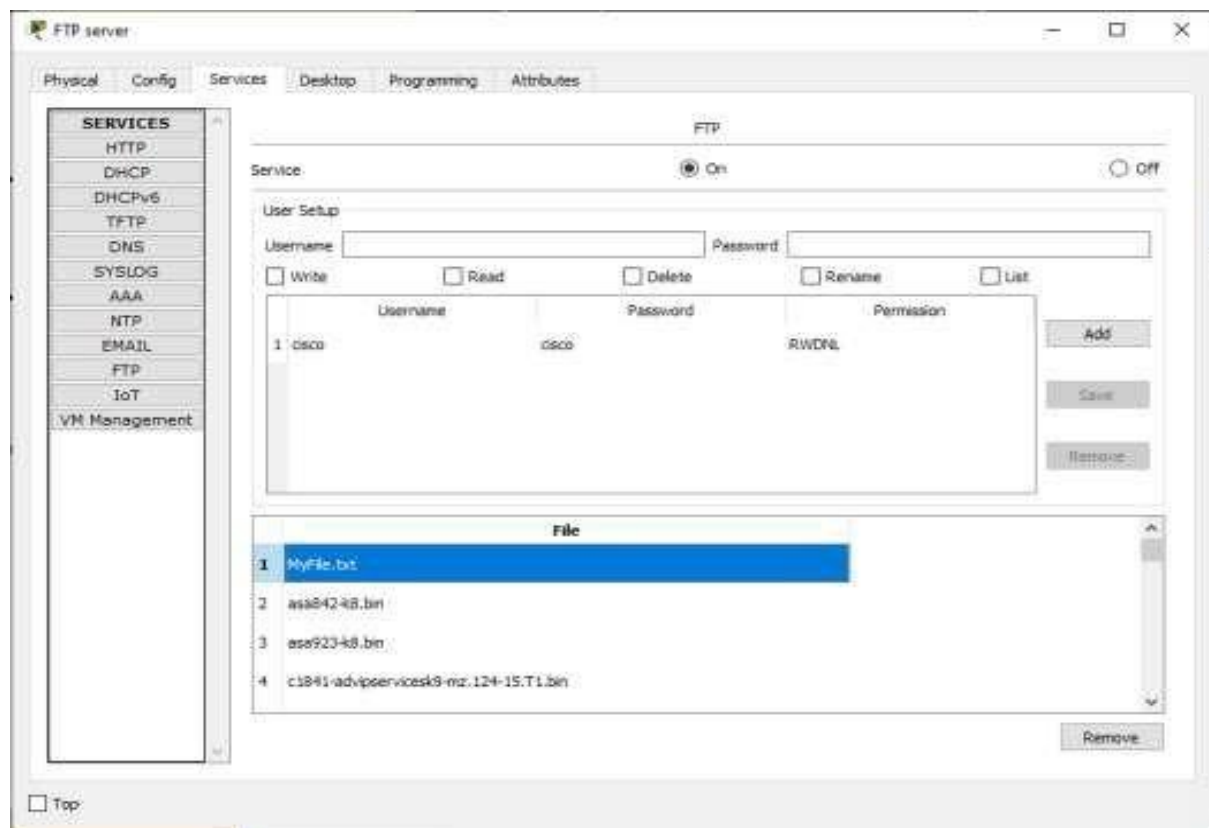
```
ftp>
ftp>put MyFile.txt

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>
```

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



CONCLUSION :

We have successfully analyzed the performance of HTTP, HTTPS and FTP

Assignment No:9

AIM: To study IPsec protocol

OBJECTIVE:

1. To learn IPsec protocol.
2. To learn ESP and AH protocol.
3. To get familiar with Wireshark tool.

PROBLEM STATEMENT: To study the IPsec (ESP and AH) protocol by capturing the packets using Wireshark tool.

OUTCOME: Students will able to

1. compare IPsec protocol.
2. Implement IPsec protocol using Wireshark tool.

THEORY-CONCEPT:

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. IPsec uses two distinct protocols, Authentication Header (AH) and Encapsulating Security Payload (ESP), which are defined by the IETF.

AUTHENTICATION HEADER (AH)

The AH protocol provides a mechanism for authentication only. AH provides data integrity, data origin authentication, and an optional replay protection service. Data integrity is ensured by using a message digest that is generated by an algorithm such as HMAC-MD5 or HMAC-SHA. Data origin authentication is ensured by using a shared secret key to create the message digest. Replay protection is provided by using a sequence number field with the AH header. AH authenticates IP headers and their payloads, with the exception of certain header fields that can be legitimately changed in transit, such as the Time To Live (TTL) field.

ENCAPSULATING SECURITY PAYLOAD (ESP)

The ESP protocol provides data confidentiality (encryption) and authentication (data integrity, data origin authentication, and replay protection). ESP can be used with confidentiality only, authentication only, or both confidentiality and authentication. When ESP provides authentication functions, it uses the same algorithms as AH, but the coverage is different. AH-style authentication authenticates the entire IP packet, including the outer IP header, while the ESP authentication mechanism authenticates only the IP datagram portion of the IP packet.

CONCLUSION:

We have successfully implemented IPsec protocol using Wireshark tool.

Assignment No:10

AIM: Steps for implementation of S/MIME email security, POP3 through Microsoft® Office Outlook

OBJECTIVE: 1. To learn email protocols using Microsoft® Office Outlook

PROBLEM STATEMENT: Illustrate the steps for implementation of S/MIME email security, POP3 through Microsoft® Office Outlook

OUTCOME: Students will able to

1. implement S/MIME, POP3 using Microsoft® Office Outlook

THEORY-CONCEPT:

S/MIME:

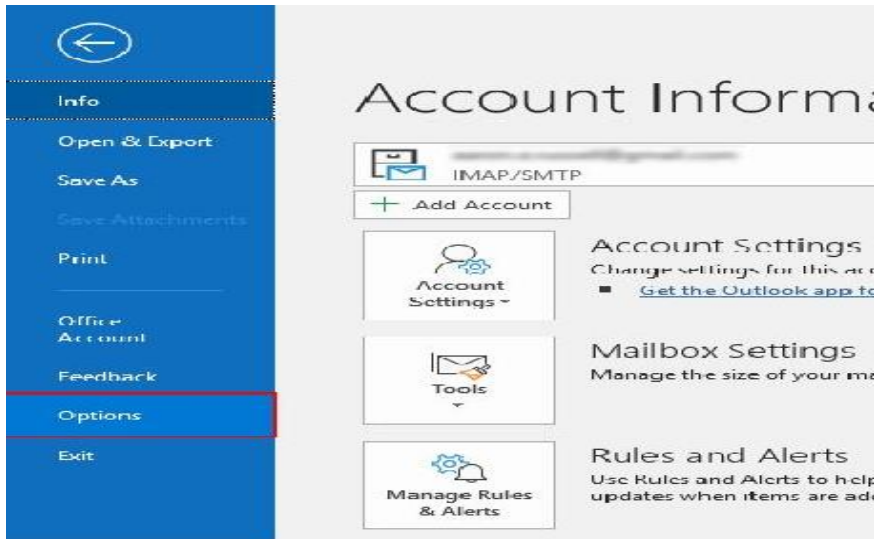
Step 1: Download your certificate.

Download a PKCS#12 file with your certificate from your SSL.com account by clicking the link supplied in your Certificate Activation Link email and following the on-screen instructions in your web browser. You will be prompted to create a password before downloading the file. (Keep this password secure – you will need it later.) Make sure to keep track of where you saved your PKCS#12 file, and do not lose it. If you lose your private key, you will be unable to read messages encrypted with your public key.

Note: when downloading your certificate it is possible to choose between the RSA and ECDSA algorithms via the Algorithm drop-down menu. However, ECDSA keys cannot be used for email encryption, so it's best to leave this set to RSA.

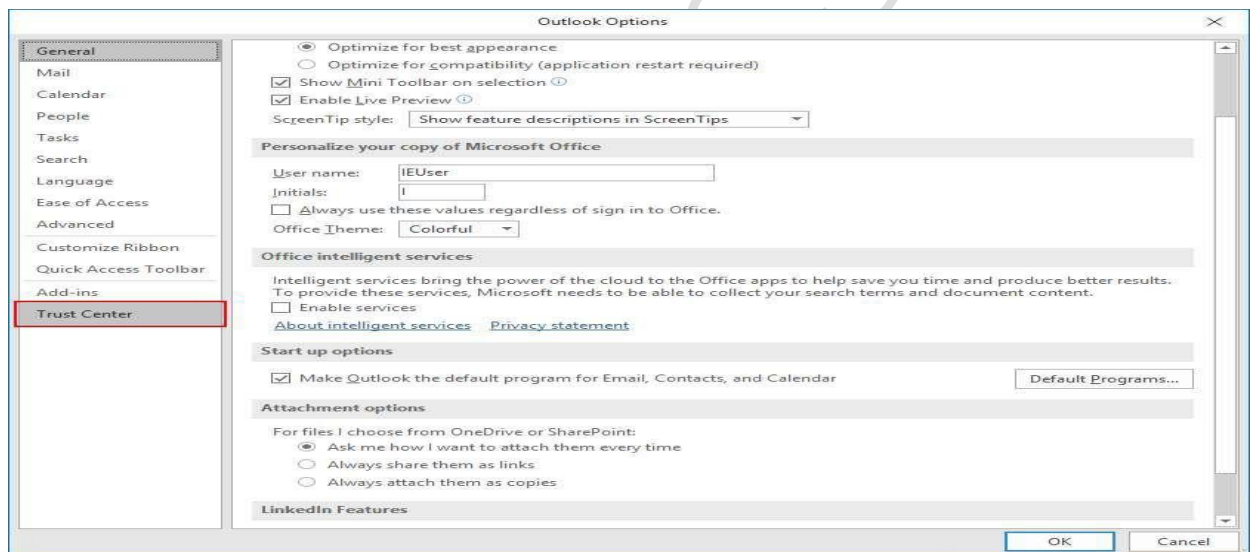
Step 2: Open Outlook Options.

In Outlook, select File from the main menu, then click Options.



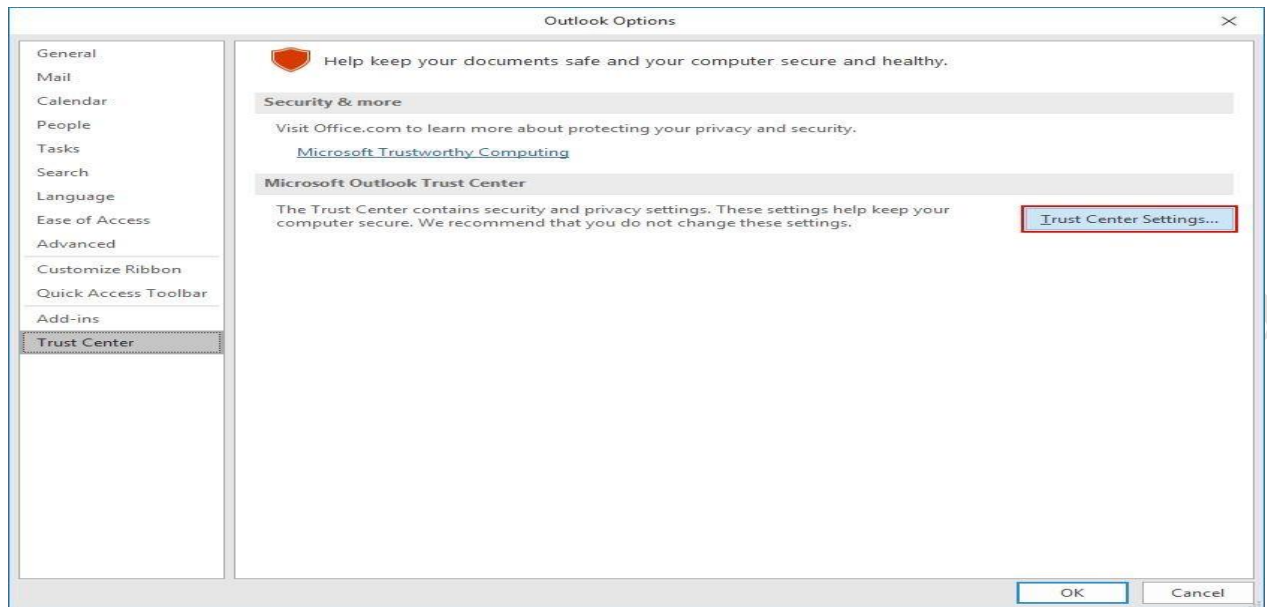
Step 3: Open Trust Center.

Select Trust Center at the bottom of the menu on the left side of the Outlook Options window



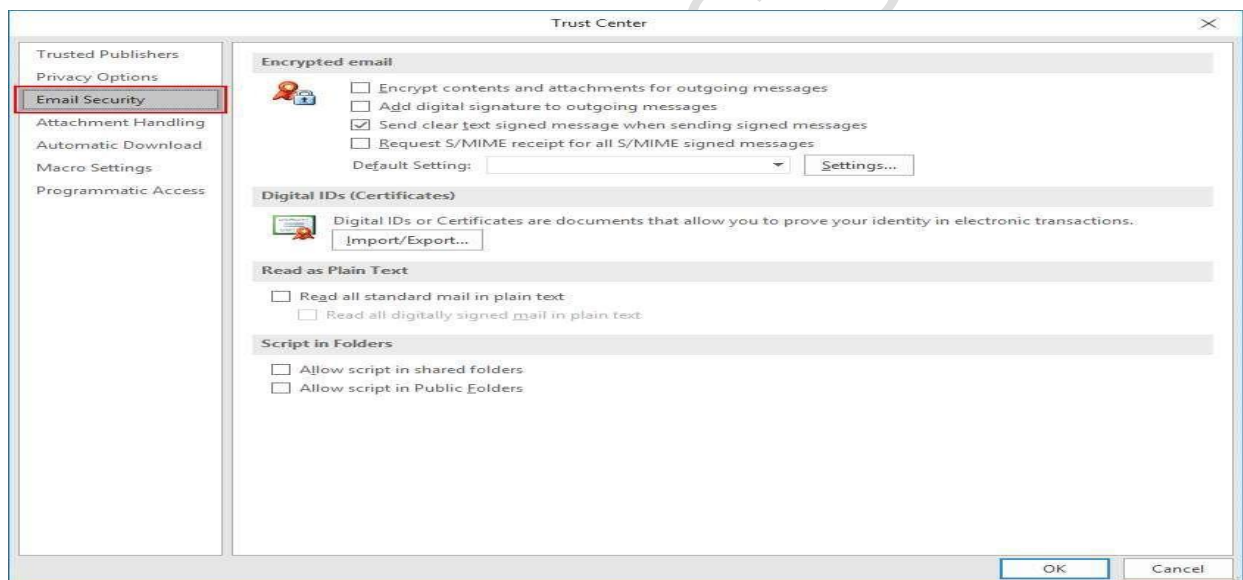
Step 4: Open Trust Center Settings.

Click the Trust Center Settings button.



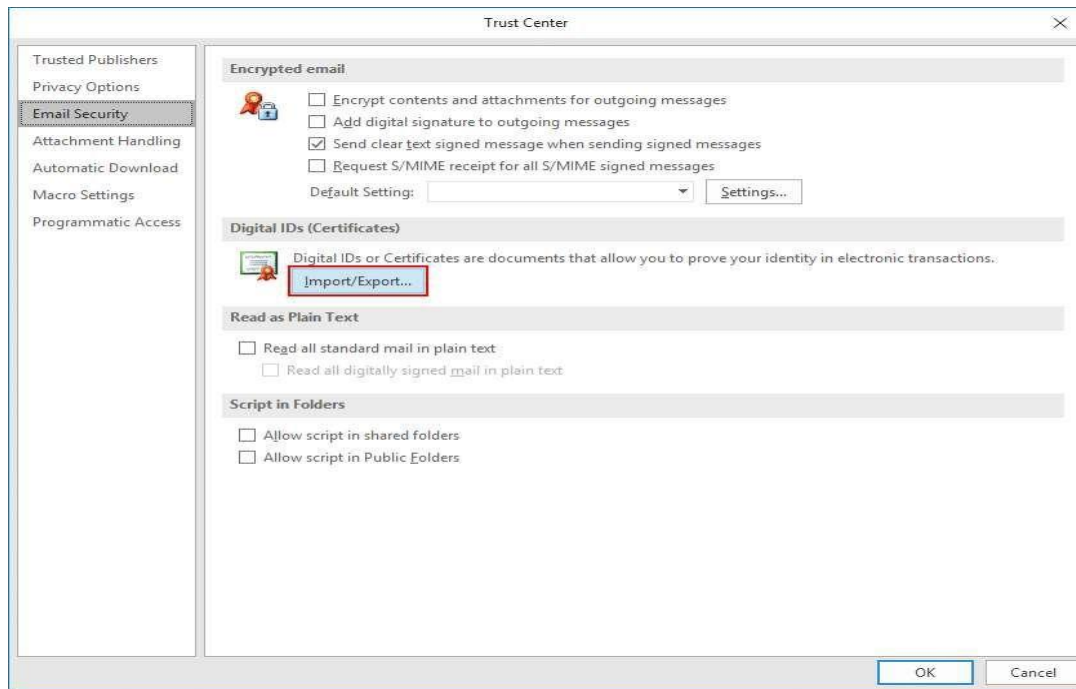
Step 5: Select Email Security.

Select **Email Security** from the left-hand menu of the **Trust Center** window.



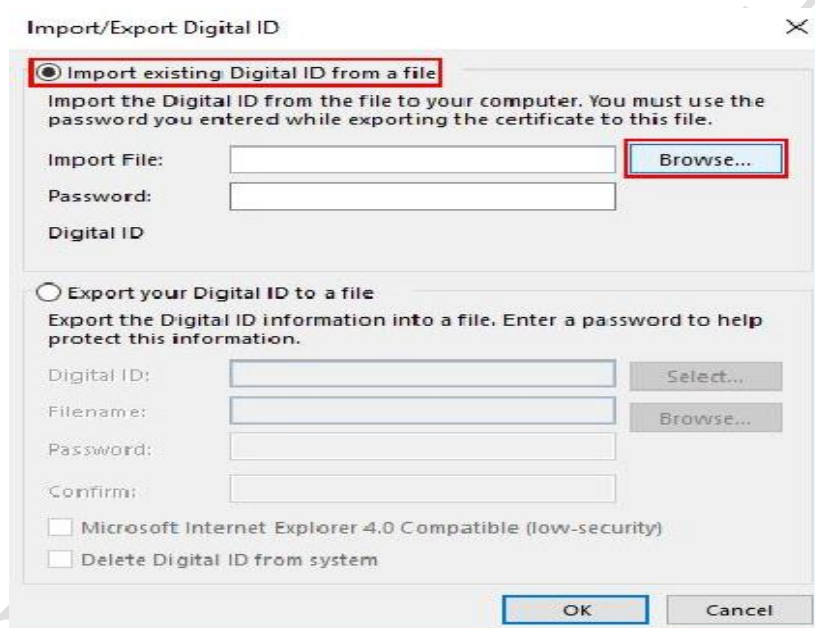
Step 6: Click Import/Export.

Click the Import/Export button, under Digital IDs (Certificates).



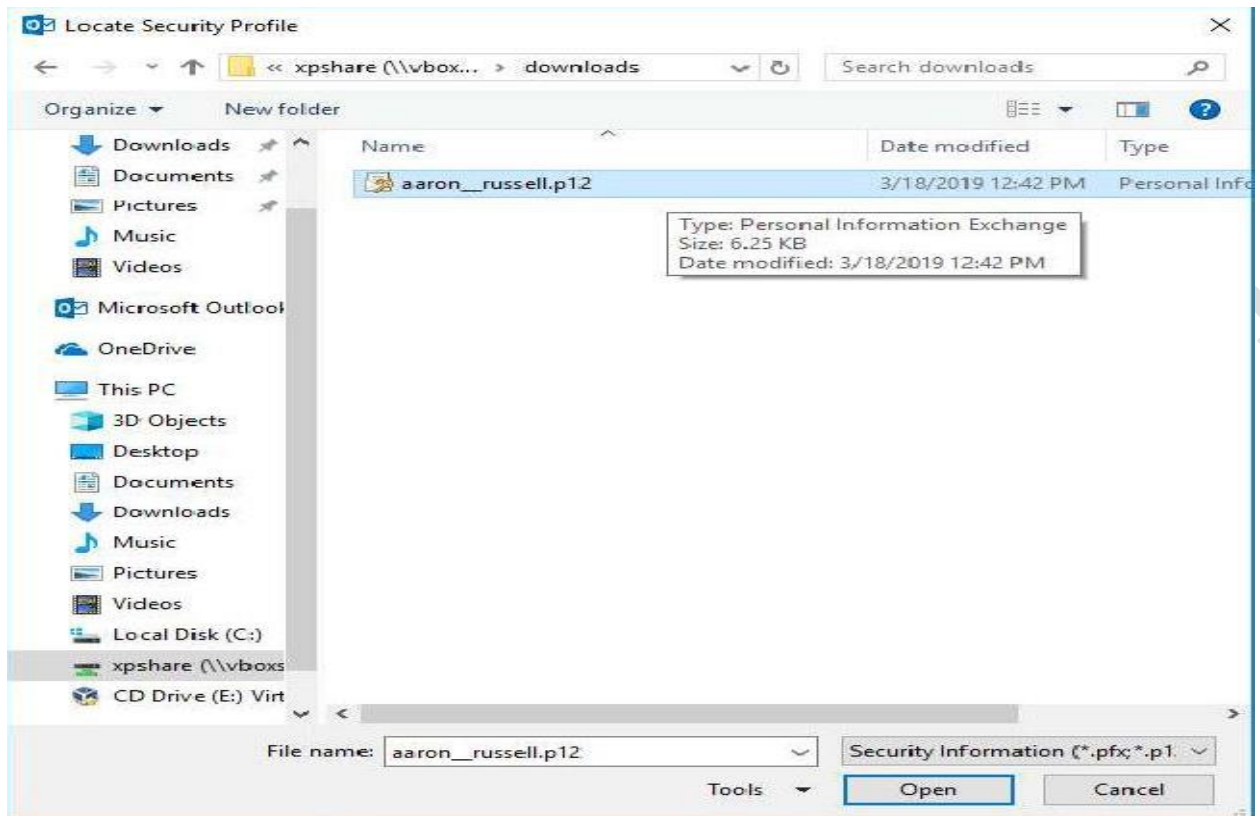
Step 7: Browse for file.

Make sure Import existing Digital ID from a file is checked, then click Browse...



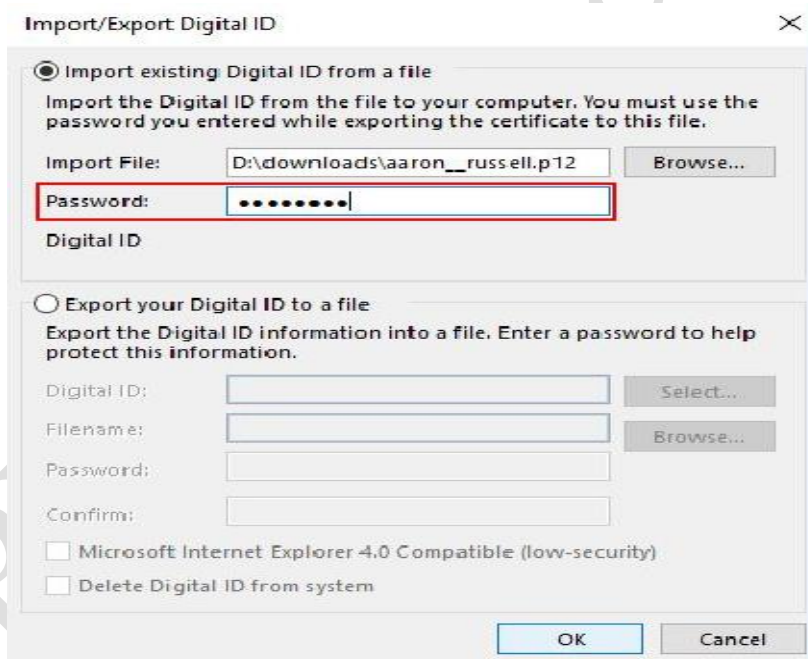
Step 8: Open file.

Navigate to the PKCS#12 file, then click Open. The filename extension should be .p12.



Step 9: Enter PKCS#12 password.

Enter the password you used when downloading the PKCS#12 file, then click **OK**.



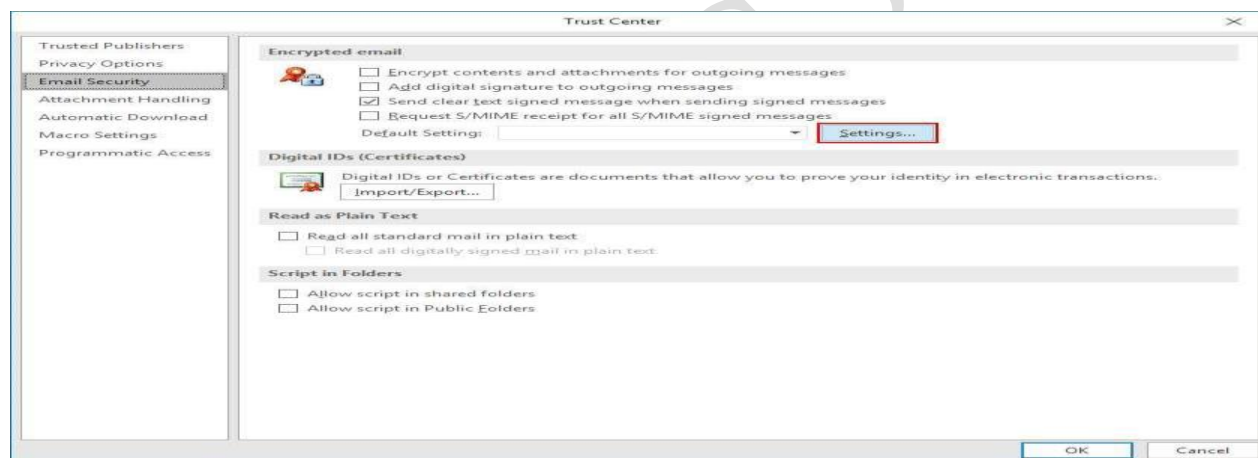
Step 10: Click OK.

Click **OK** on the security dialog box that pops up.



Step 11: Open encrypted email settings

Click the **Settings** button, under **Encrypted email**.

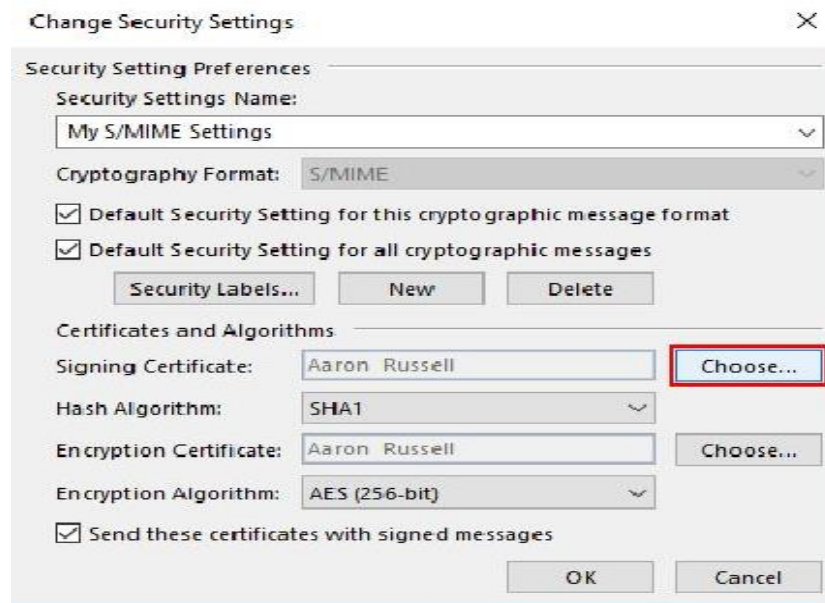


Step 12: Name security settings.

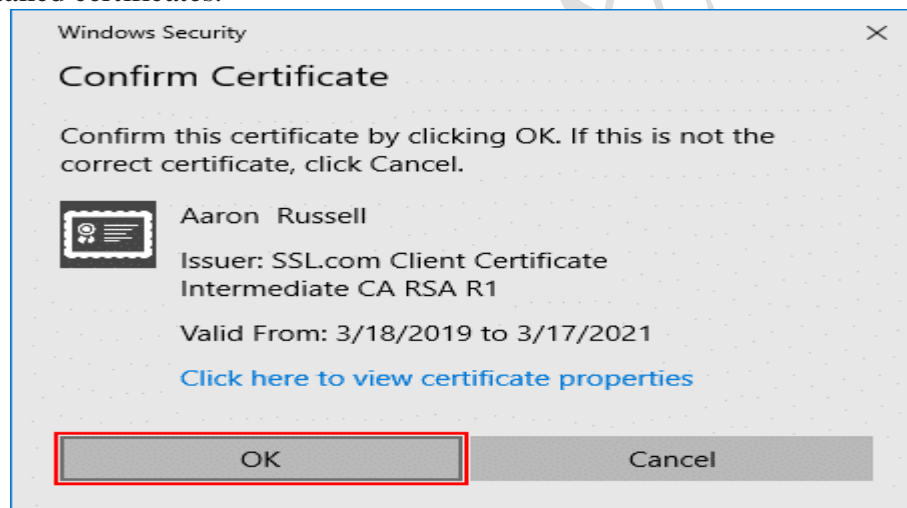
Enter a name for your security settings



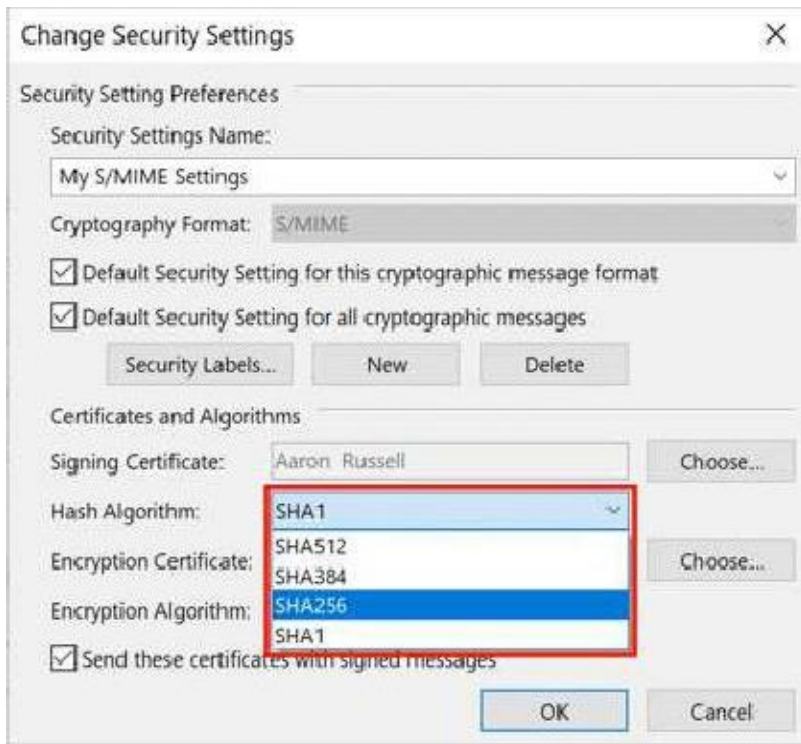
Step 13: Choose signing certificate.
Click **Choose**, next to **Signing Certificate**.



Step 14: Confirm or select certificate.
If you have only installed one certificate (as shown here), you can click **OK** on the **Confirm Certificate** dialog box that pops up. Otherwise, you will have to choose one from a list of installed certificates.



Step 15: Set hash algorithm.
Set the **Hash Algorithm** to **SHA256**.



Step 16: Choose encryption certificate.

Click **Choose**, next to **Encryption Certificate**, and click **OK** on the **Confirm Certificate** dialog box. Again, if you have more than one certificate, select the same one you chose for **Signing Certificate**.



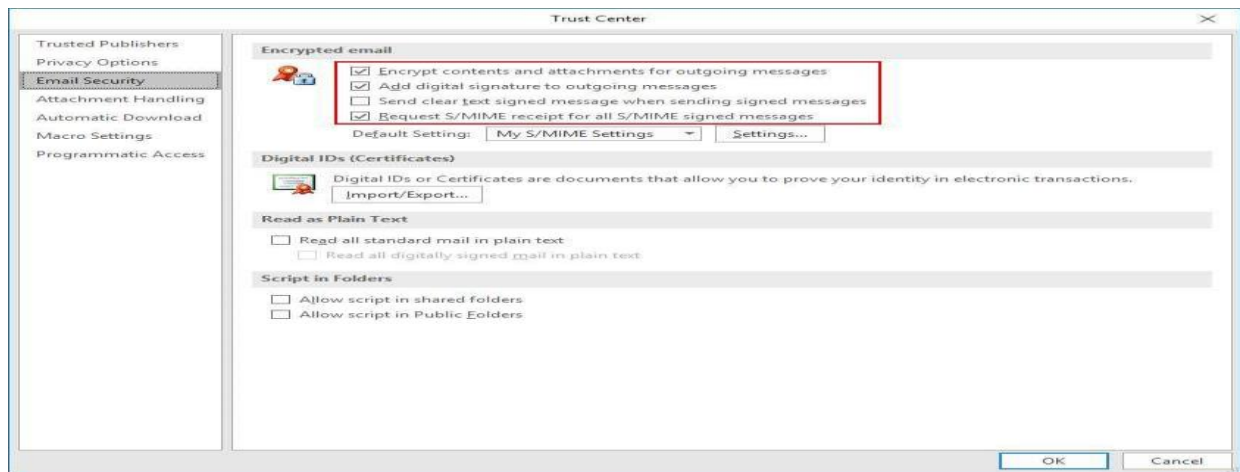
Step 17: Close window.

Click **OK** to close the **Change Security Settings** window.

Step 18: Set S/MIME defaults.

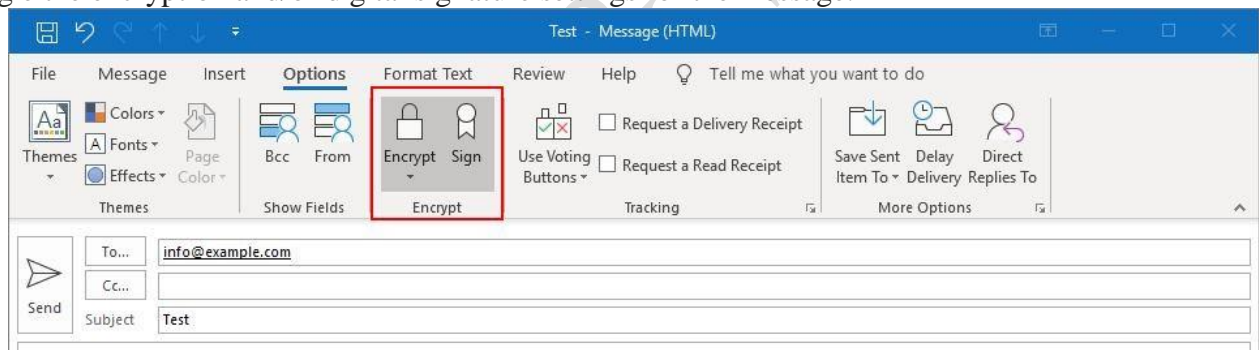
Set your desired default options for S/MIME email via the four checkboxes under **Encrypted email**,

then click **OK** to close the **Trust Center** Window.



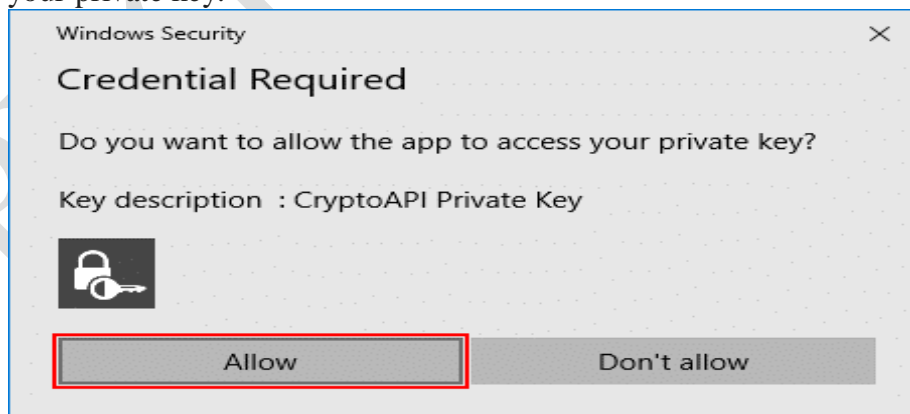
Step 19: Set S/MIME options in a new message.

Now that your S/MIME certificate is installed and configured, you can start sending signed and encrypted messages. Begin by creating a new email message in Outlook. Under Options, you can toggle the encryption and/or digital signature settings for the message.



Step 20: Allow Outlook to use your private key.

After sending, click Allow in the Windows Security dialog box that appears, allowing Outlook to use your private key.



Step 21: Potential problem with encryption.

Note that if you attempt to send encrypted email and do not have your recipient's public key, you will get an error message giving the option to send the message unencrypted. You can solve this issue by having them send you a signed email message, then adding them as a contact in Outlook.



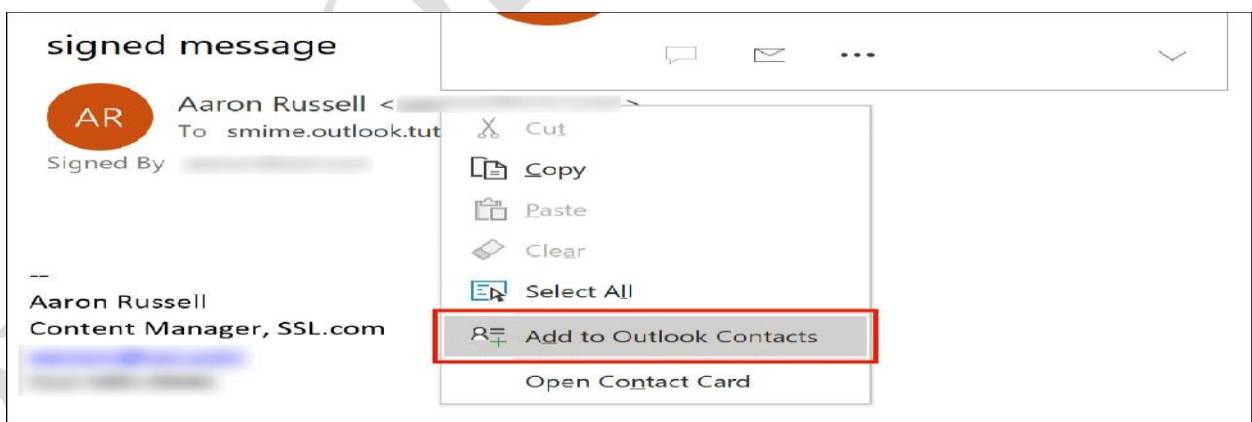
Step 22: Confirm signature.

When your contact sends you a signed email, you should see a small ribbon icon in the upper right corner of the message. You can confirm the certificate's details by clicking the icon.



Step 23: Add contact (step 1).

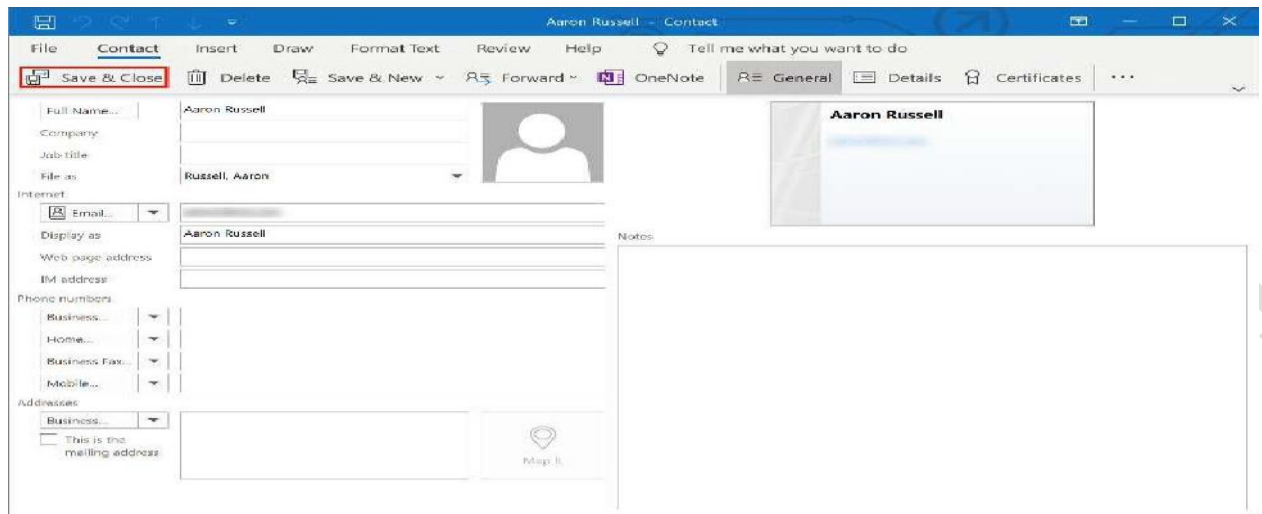
Right-click the sender's name and select **Add to Outlook Contacts**.



Step 24: Add contact (step 2).

Click **Save and Close** to save your contact.

You will now be able to send encrypted email to this Recipient



POP3:

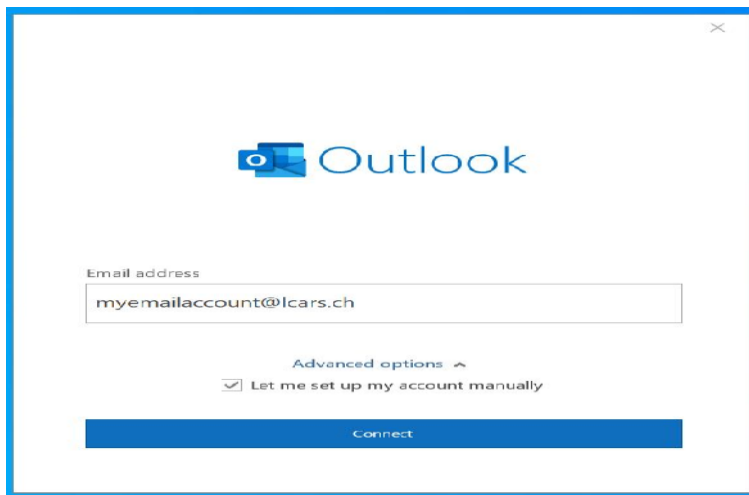
Step 1: Open Outlook. This can be done by clicking the logo in the bottom left corner of your taskbar on your keyboard type outlook, then select **Outlook** from the search results.

Step 2: When Outlook opens for the first time, you'll see a screen that looks like this:



- Enter your e-mail address.
- Click **Advanced Options**.
- Check the box next to **Let me setup my account manually**.
- Click **Connect**.

Your screen should look like the one below, except your email address should be filled in:



Step 3: Outlook will display a list of email services and options for you to choose from. We're setting up POP, so click **POP**.

After you select **POP**, the setup wizard will automatically proceed to the next screen.

Step 4: On the next screen, you will be able to enter your POP Account & Server Details. If the email account you're adding is hosted by ChemiCloud, your server configuration will be similar to the one below, you would just replace **yourdomain.tld** with your domain name.

Your **Incoming Mail Server** should be **mail.yourdomain.tld**

Under **Port**, use **995** for **POP**

For **Encryption Method**, select **SSL/TLS**

Check the box next to **Require login using Secure Password Authentication(SPA)**

Your **Outgoing Mail Server** should be **mail.yourdomain.tld**

Under **Port**, use **465**

For **Encryption Method**, select **SSL/TLS**

Check the box next to **Require login using Secure Password Authentication(SPA)**

Your configuration should be similar to the one below:

POP Account Settings
myemailaccount@lcars.ch (Not you?)

Incoming mail
Server mail.lcars.ch Port 995
☒ This server requires an encrypted connection (SSL/TLS)
☒ Require logon using Secure Password Authentication (SPA)

Outgoing mail
Server mail.lcars.ch Port 465
Encryption method SSL/TLS
☒ Require logon using Secure Password Authentication (SPA)

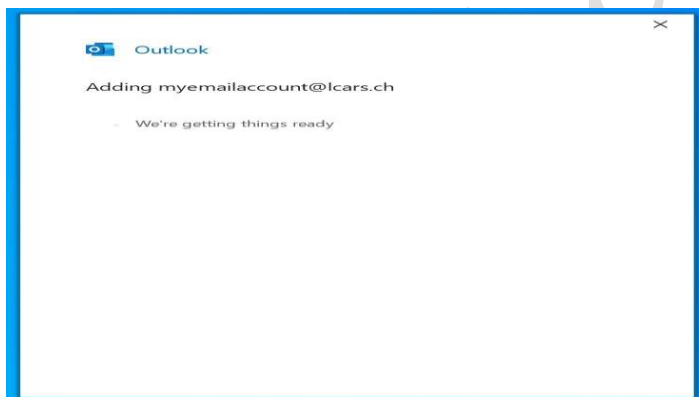
Message delivery
☐ Use an existing data file
Browse...

Go back Next

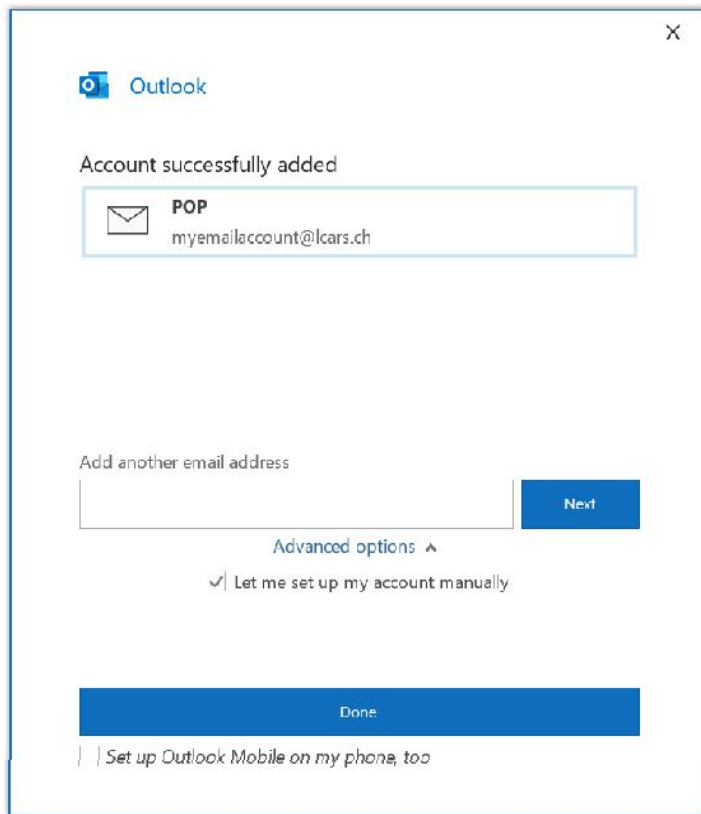
Click the blue **Next** button to proceed.

Step 5: Enter the password for this email account, and click the blue **Next** button.

Step 6: Outlook will connect to the mail server to verify your account configuration and subsequently add the account to Outlook.



Step 7: If you entered your account details correctly, you'll see a screen that looks like this:



CONCLUSION:

We have successfully implemented S/MIME email security and POP3 through Microsoft® Office Outlook

GROUP C

Assignment No:11

AIM: Installation and configuration of DHCP sever.

OBJECTIVE: 1. To learn DHCP server installation and configuration.

PROBLEM STATEMENT: Installing and configuring DHCP server and assign IP addresses to client machines using DHCP server

OUTCOME: Students will able to
1. install and configure DHCP server.

THEORY-CONCEPT:

A DHCP Server is a network server that automatically provides and assigns IP addresses, default gateways and other network parameters to client devices. It relies on the standard protocol known as Dynamic Host Configuration Protocol or DHCP to respond to broadcast queries by clients. A DHCP server automatically sends the required network parameters for clients to properly communicate on the network. Without it, the network administrator has to manually set up every client that joins the network, which can be cumbersome, especially in large networks. DHCP servers usually assign each client with a unique dynamic IP address, which changes when the client's lease for that IP address has expired.

How to Install the DHCP Service

Before you configure the DHCP service, you must install it on the server. DHCP is not installed by default during a typical installation of Windows Standard Server 2003 or Windows Enterprise Server 2003. You can install DHCP during the initial installation of Windows Server 2003, or after the initial installation is completed.

How to Install the DHCP Service on an Existing Server

1. Click Start, point to Control Panel, and then click **Add or Remove Programs**.
2. In the **Add or Remove Programs** dialog box, click Add/Remove Windows Components.
3. In the Windows Components Wizard, click Networking Services in the Components list, and then click Details.
4. In the Networking Services dialog box, click to select the **Dynamic Host Configuration Protocol (DHCP)** check box, and then click OK.
5. In the Windows Components Wizard, click Next to start Setup. Insert the Windows Server 2003 CD-ROM into the computer's CD-ROM or DVD-ROM drive if it is prompted to do so. Setup copies the DHCP server and tool files to your computer.
6. When Setup is completed, click Finish.

How to Configure the DHCP Service

After you have installed the DHCP service and started it, you must create a scope. The scope is a range of valid IP addresses available for lease to the DHCP client computers on the network. Microsoft recommends that, each DHCP server in your environment has at least one scope that does not overlap with any other DHCP server scope in your environment. In Windows Server 2003, DHCP servers in an Active Directory-based domain must be authorized to prevent *rogue* DHCP servers from coming online. Any Windows Server 2003 DHCP Server that determines itself to be unauthorized will not manage clients

How to Create a New Scope

1. Click Start, point to Programs, point to Administrative Tools, and then click DHCP.

2. In the console tree, right-click the DHCP server on which you want to create the new DHCP scope, and then click New Scope.

3. In the New Scope Wizard, click Next, and then type a name and description for the scope. The name can be anyone that you want, but it should be descriptive enough so that you can identify the purpose of the scope on your network (for example, you can use a name such as "Administration Building Client Addresses"). Click Next.

4. Type the range of addresses that can be leased as part of this scope. For example, use a range of IP addresses from a starting IP address of 192.168.100.1 to an ending address of 192.168.100.100. Because these addresses are given to clients, they must all be valid addresses for your network and not currently in use. If you want to use a different subnet mask, type the new subnet mask. Click Next.

5. Type any IP addresses that you want to exclude from the range that you entered. These addresses include any one in the range described in step 4 that may have already been statically assigned to various computers in your organization. Typically, domain controllers, Web servers, DHCP servers, Domain Name System (DNS) servers, and other servers, have statically assigned IP addresses. Click Next.

6. Type the number of days, hours, and minutes before an IP address lease from this scope expires. It determines how long a client can hold a leased address without renewing it. Click Next, and then click **Yes, I want to configure these options now** to extend the wizard to include settings for the most common DHCP options. Click Next.

7. Type the IP address for the default gateway that should be used by clients that obtain an IP address from this scope. Click Add to add the default gateway address in the list, and then click Next.

8. If you are using DNS servers on your network, type your organization's domain name in the **Parent domain** box. Type the name of your DNS server, and then click Resolve to make sure that your DHCP server can contact the DNS server and determine its address. Click Add to

include that server in the list of DNS servers that are assigned to the DHCP clients. Click Next, and then follow the same steps. If you are using a Windows Internet Naming Service (WINS) server, by adding its name and IP address, click Next.

9. Click **Yes, I want to activate this scope now** to activate the scope and allow clients to obtain leases from it, and then click Next.

10. Click Finish.

11. In the console tree, click the server name, and then click Authorize on the Action menu.

CONCLUSION :

We have successfully installed and configured DHCP server.

Assignment No: 12

AIM: Program for DNS lookup

OBJECTIVE:

1. To learn DNS concept.
2. To get the host name and IP address.
3. To map the host name with IP address and Vice-versa

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

OUTCOME: Students will able to

1. summarize DNS lookup.

THEORY-CONCEPT:

NEED FOR DNS

To identify an entity, TCP/IP protocols use the IP address, which uniquely identifies the connection of a host to the Internet. However, people prefer to use names instead of numeric addresses. Therefore, we need a system that can map a name to an address or an address to a name. When the Internet was small, mapping was done using a host file. The host file had only two columns: name and address. Every host could store the host file on its disk and update it periodically from a master host file. When a program or a user wanted to map a name to an address, the host consulted the host file and found the mapping. Today, however, it is impossible to have one single host file to relate every address with a name and vice versa. The host file would be too large to store in every host. In addition, it would be impossible to update all the host files every time there is a change. One solution would be to store the entire host file in a single computer and allow access to this centralized information to every computer that needs mapping. But we know that this would create a huge amount of traffic on the Internet. Another solution, the one used today, is to divide this huge amount of information into smaller parts and

store each part on a different computer. In this method, the host that needs mapping can contact the closest computer holding the needed information. This method is used by the Domain Name System (DNS).

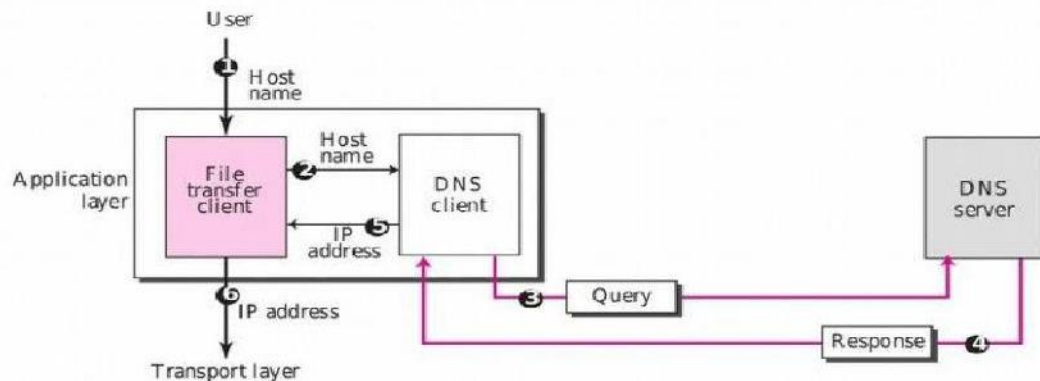


Figure 1: How TCP/IP uses a DNS client and a DNS server to map a name to an address; the reverse mapping is similar.

In Figure 1, a user wants to use a file transfer client to access the corresponding file transfer server running on a remote host. The user knows only the file transfer server name, such as forouzan.com. However, the TCP/IP suite needs the IP address of the file transfer server to make the connection.

The following six steps map the host name to an IP address.

1. The user passes the host name to the file transfer client.
2. The file transfer client passes the host name to the DNS client.
3. We know that each computer, after being booted, knows the address of one DNS server. The DNS client sends a message to a DNS server with a query that gives the file transfer server name using the known IP address of the DNS server.
4. The DNS server responds with the IP address of the desired file transfer server.
5. The DNS client passes the IP address to the file transfer server.
6. The file transfer client now uses the received IP address to access the file transfer server.

NAME SPACE

To be unambiguous, the names assigned to machines must be carefully selected from a name space with complete control over the binding between the names and IP addresses. In other words, the names must be unique because the addresses are unique. A name space that maps each address to a unique name can be organized in two ways: flat or hierarchical.

FLAT NAME SPACE

In a flat name space, a name is assigned to an address. A name in this space is a sequence of characters without structure. The names may or may not have a common section; if they do, it has no meaning. The main disadvantage of a flat name space is that it cannot be used in a large system such as the Internet because it must be centrally controlled to avoid ambiguity and duplication.

HIERARCHICAL NAME SPACE

In a hierarchical name space, each name is made of several parts. The first part can define the nature of the organization, the second part can define the name of an organization, the third part can define departments in the organization, and so on. In this case, the authority to assign and control the name spaces can be decentralized. A central authority can assign the part of the name that defines the nature of the organization and the name of the organization.

DOMAIN NAME SPACE

To have a hierarchical name space, a domain name space was designed. In this design the names are defined in an inverted-tree structure with the root at the top. The tree can have only 128 levels: level 0 (root) to level 127 (see Figure 2)

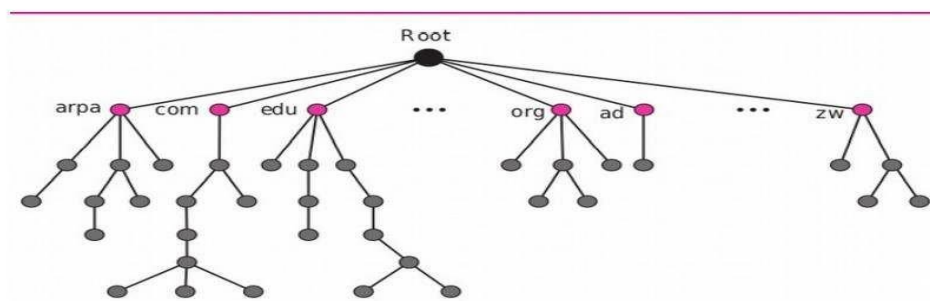


Figure 2 : Domain Name Space

LABEL

Each node in the tree has a label, which is a string with a maximum of 63 characters. The root label is a null string (empty string). DNS requires that children of a node (nodes that branch from the same node) have different labels, which guarantees the uniqueness of the domain names.

DOMAIN NAME

Each node in the tree has a domain name. A full domain name is a sequence of labels separated by dots (.). The domain names are always read from the node up to the root. The last

label is the label of the root (null). This means that a full domain name always ends in a null label, which means the last character is a dot because the null string is nothing.

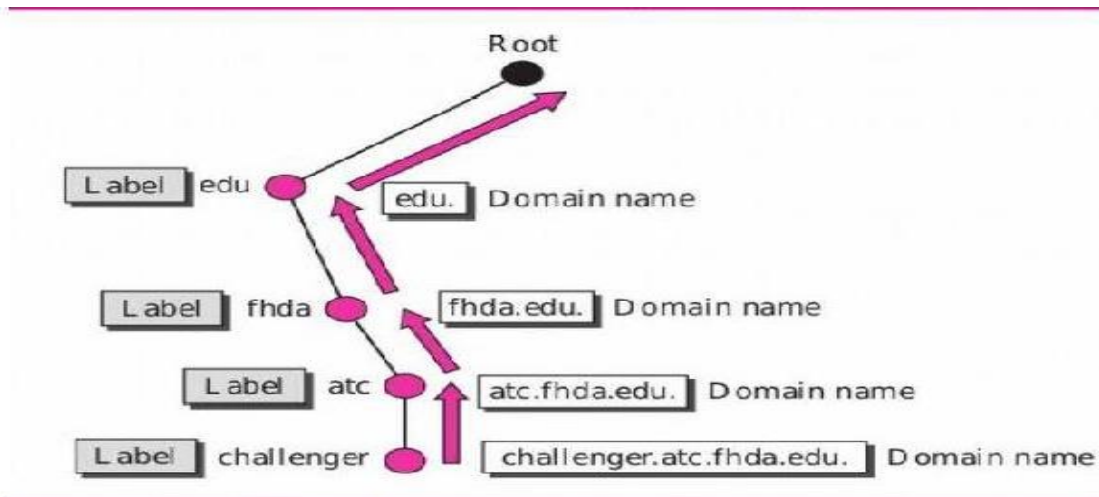


Figure 3 shows some domain names.

DOMAIN

A domain is a subtree of the domain name space. The name of the domain is the name of the node at the top of the subtree. Figure 4 shows some domains. Note that a domain may itself be divided into domains (or subdomains as they are sometimes called).

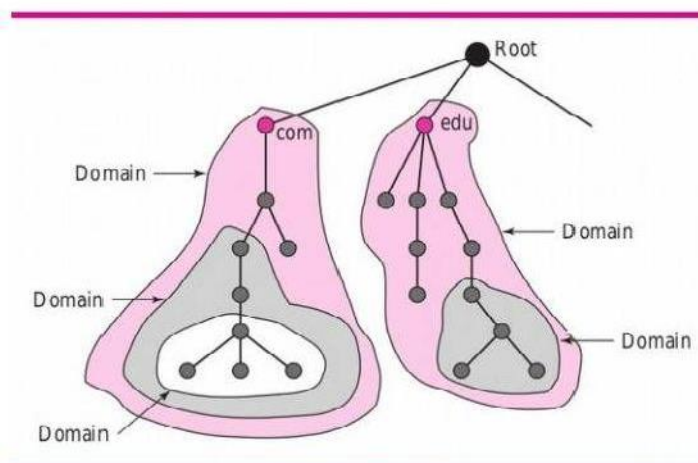


Figure 4 Domain

RESOLUTION

Mapping a name to an address or an address to a name is called name-address resolution. Resolver: DNS is designed as a client-server application. A host that needs to map an address to a name or a name to an address calls a DNS client called a resolver. The resolver accesses the closest DNS server with a mapping request. If the server has the information, it satisfies the

resolver; otherwise, it either refers the resolver to other servers or asks other servers to provide

PVGCOE & SSDIOM

the information. After the resolver receives the mapping, it interprets the response to see if it is a real resolution or an error, and finally delivers the result to the process that requested it.

Mapping Names to Addresses: Most of the time, the resolver gives a domain name to the server and asks for the corresponding address. In this case, the server checks the generic domains or the country domains to find the mapping. If the domain name is from the generic domains section, the resolver receives a domain name such as “chal.atc.fhda.edu.”. The query is sent by the resolver to the local DNS server for resolution: If the local server cannot resolve the query, it either refers the resolver to other servers or asks other servers directly. If the domain name is from the country domains section, the resolver receives a domain name such as “ch.fhda.cu.ca.us.”. The procedure is the same.

Mapping Addresses to Names A client can send an IP address to a server to be mapped to a domain name. As mentioned before, this is called a PTR query. To answer queries of this kind, DNS uses the inverse domain. However, in the request, the IP address is reversed and two labels, in-addr and arpa, are appended to create a domain acceptable by the inverse domain section. For example, if the resolver receives the IP address 132.34.45.121, the resolver first inverts the address and then adds the two labels before sending. The domain name sent is “121.45.34.132.inaddr. arpa.”, which is received by the local DNS and resolved.

Recursive Resolution: The client (resolver) can ask for a recursive answer from a name server. This means that the resolver expects the server to supply the final answer. If the server is the authority for the domain name, it checks its database and responds. If the server is not the authority, it sends the request to another server (the parent usually) and waits for the response. If the parent is the authority, it responds; otherwise, it sends the query to yet another server. When the query is finally resolved, the response travels back until it finally reaches the requesting client

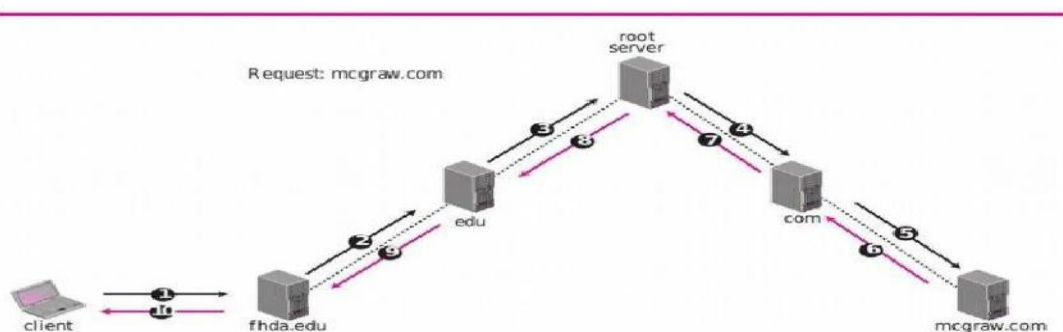


Figure 5: Recursive resolution.

CONCLUSION We have successfully performed DNS lookup program.