Experiment No: 03

Aim: 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).

Theory

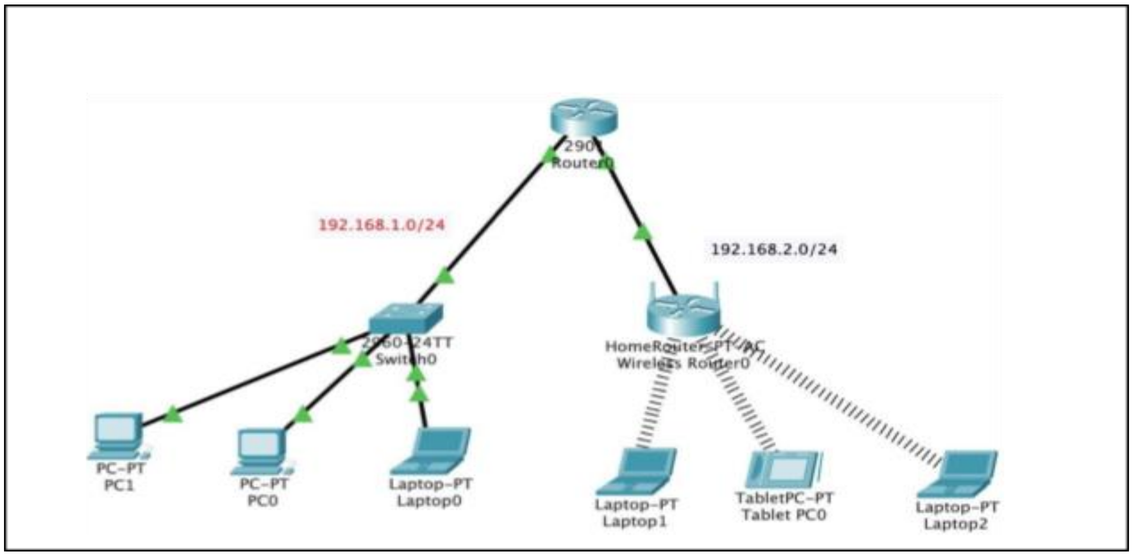
Introduction

A computer network simply means any set of computers and other devices like smartphones, smart TVs, video game systems and routers that are able to communicate back and forth with each other. Different protocols, or rule systems, exist that let computers understand each other and transmit data.

A LAN, or local area network, is a small network, often within a home or business or perhaps within a larger environment like a corporate office park or a college campus. Devices on a LAN often use the LAN's infrastructure to connect to the public internet, but they can often communicate with each other directly through the LAN more quickly. For instance, it's not usually necessary to send a file to the public internet in order to get it to a printer on the same LAN. A LAN can use wireless communication, wired connections or both.

A wide area network usually traverses multiple geographical areas. The internet is the most prominent example of the WAN network type, though other wide area networks exist for scientific purposes, military and government work, and to connect far-flung offices and data centers within some big corporations.

TOPOLOGY



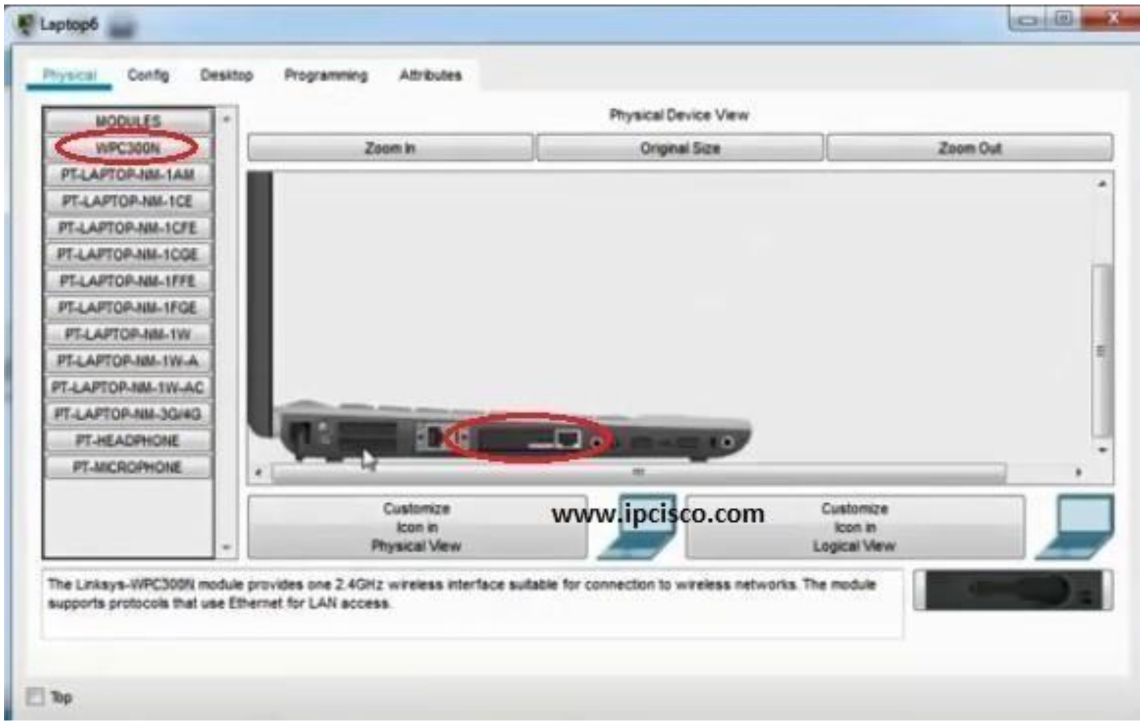
WLAN Configuration

WLANs (Wireless LANs) are very common in today's World. Everywhere there are a lot of wireless networks. Even now, you are in many of these wireless signals. It is not an healthty life but they are in our lives. In this lesson we will focus **WLAN Packet Tracer Configuration** and we will learn **How to Configure a WLAN on Packet Tracer**.
For Our **WLAN Configuration on Packet Tracer**, we will use the below topology that is consist of one One Wireless Access Point, One Server , Three Laptops and One Smartphone. Smartphones are everywhere, even in **Cisco Packet Tracer** for many years: Now, let's sumamrize what will we do for **Packet Tracer WLAN**

Configuration :

- **Place Wireless Interface Card to Laptops**
- **IP Check on WLAN Devices**
- **DHCP Server Configuration**
- **IP Check on WLAN Devices again Place Wireless Interface Card to Laptops**

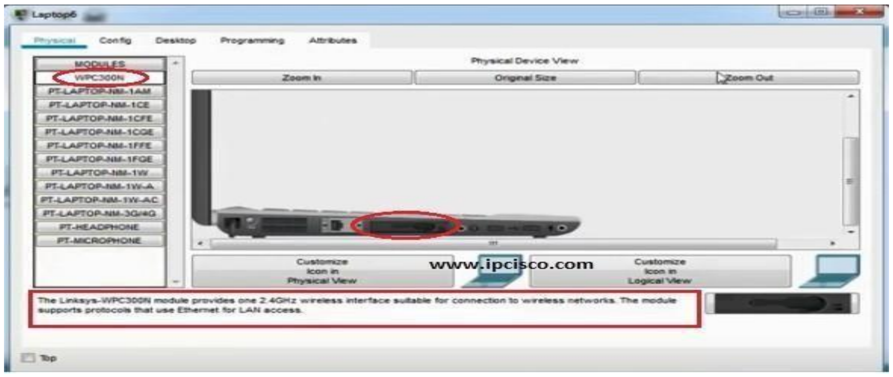
By **default** laptops has classic **Ethernet card**. To involve in a wireless network, we should have a wireless interface card. So, in each laptop, we should turn off the



The Linksys-WPC300N module provides one 2.4GHz wireless interface suitable for connection to wireless networks. The module supports protocols that use Ethernet for LAN access.

laptop, and remove the classical Ethernet, instead of it we place **Wireless Interface Card (WPC300N)**. Then, we power on the laptop again.

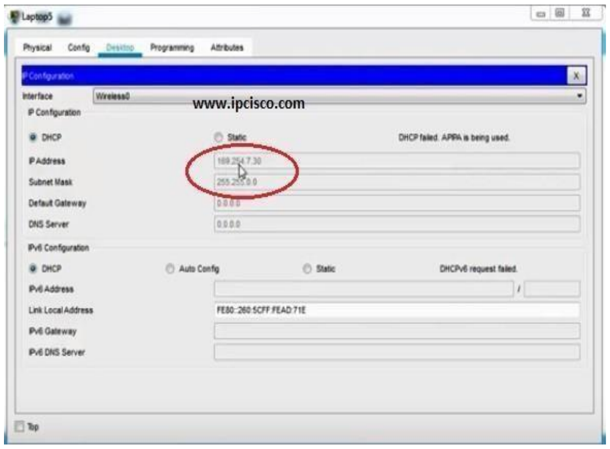
After this process each laptop connects to the wireless Access Point in Packet Tracer. Smartphone devices in Packet Tracer connects to **Access Points (AP)** by default. So, there is nothing to do on them



The Linksys-WPC300N module provides one 2.4GHz wireless interface suitable for connection to wireless networks. The module supports protocols that use Ethernet for LAN access.

IP Check on WLAN Devices

We will check the IP addresses of the laptops. For now, checking only one of them is enough. Because, in the beginning, if there is no Static IP Configuration and no DHCP, an IP from a special block is assigned to the devices. These are **APIPA (Automatic Private IP Addressing)** addresses. These addresses are from the block "**169.254.x.x/25**". Simple, when we say this type of IP address in a device, we can say that it has no IP address.



Interface: Wireless0

IP Configuration
● DHCP ○ Static DHCP failed. APIPA is being used.
IP Address: 169.254.7.30
Subnet Mask: 255.255.0.0
Default Gateway: 0.0.0.0
DNS Server: 0.0.0.0

IPv6 Configuration
● DHCP ○ Auto Config ○ Static DHCPv6 request failed.
IPv6 Address: /
Link Local Address: FE80::260:5CFF:FEAD:71E
IPv6 Gateway:
IPv6 DNS Server:

Conclusion: Successfully done the Setup for 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).

Coding Efficiency	Viva	Timely Completion	Total	Dated Sign of Course In-charge
5	3	2	10	

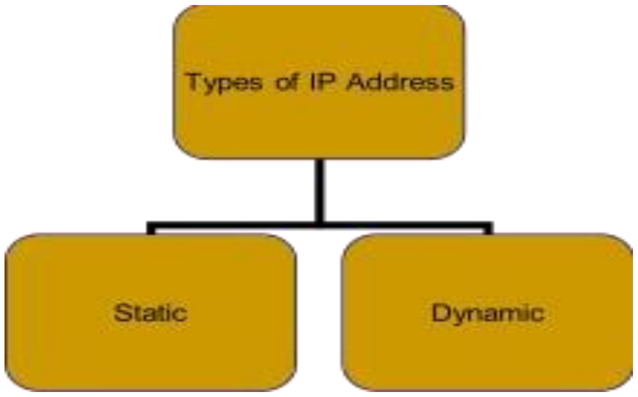
Experiment No: 04

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

Theory

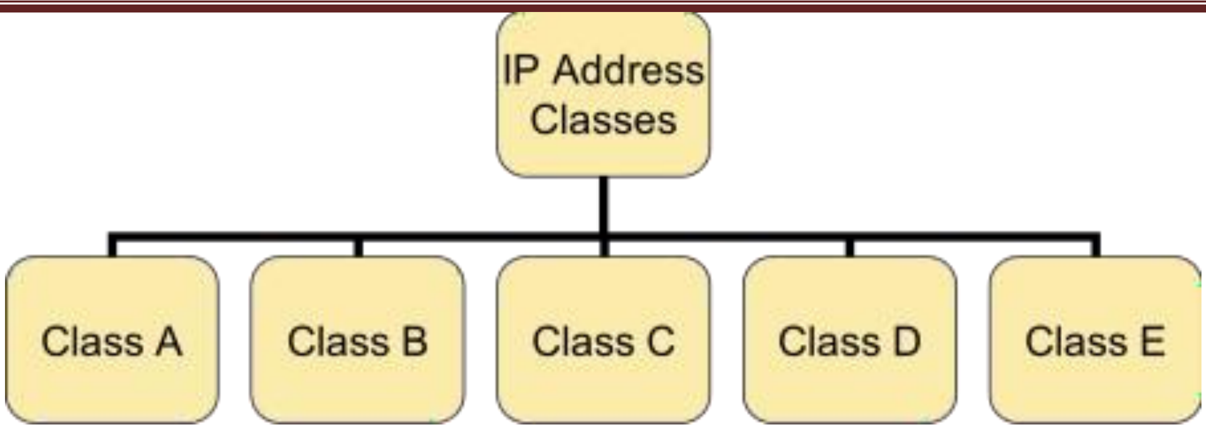
Introduction

- 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



The 5 IP classes are split up based on the value in the 1st octet:

Class A: 0-127

Class B: 128-191

Class C: 192-223

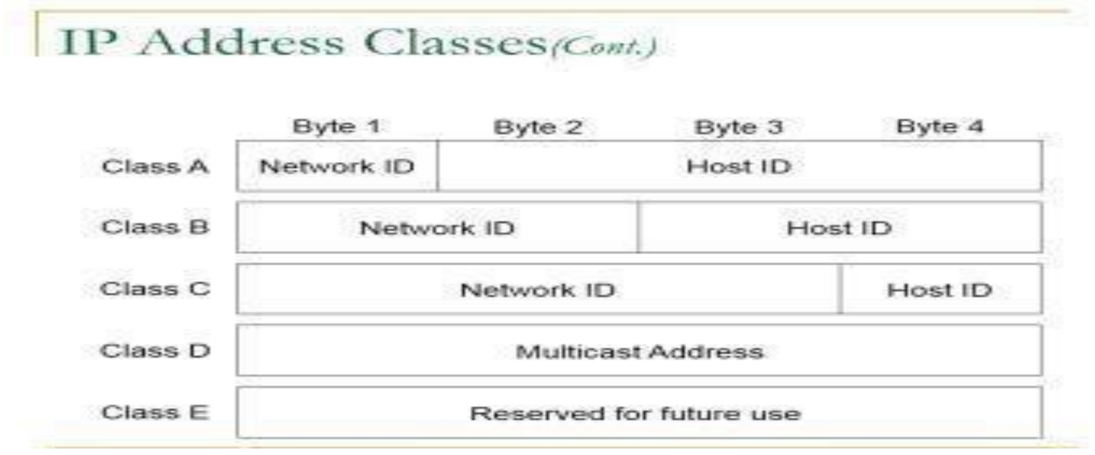
Class D: 224-239

Class E: 240-2552

Examples of IP 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

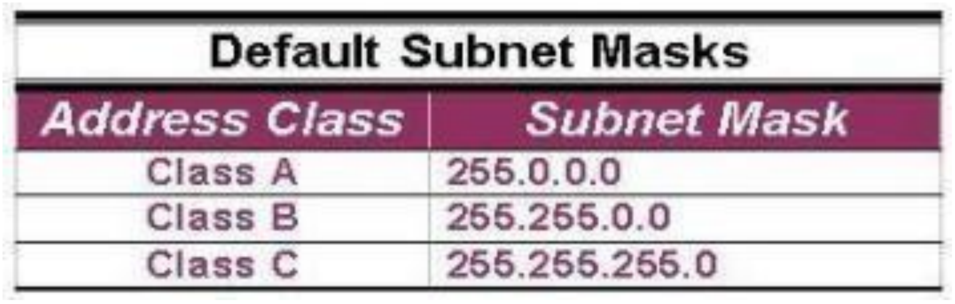
Default Mask:

- 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

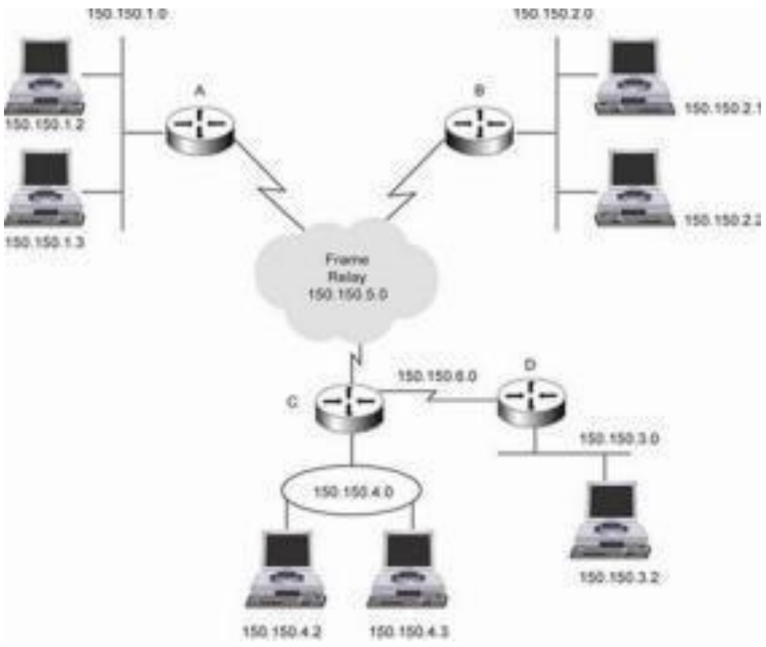
Address Class	Subnet Mask
Class A	255.0.0.0
Class B	255.255.0.0
Class C	255.255.255.0

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 form 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)

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 ives:

1. 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

2. Express the result in Dotted Decimal Notation

3. 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 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 Sub-netting and the importance of sub-netting.

Coding Efficiency	**Viva**	**Timely Completion**	**Total**	**Dated Sign of Course In-charge**
5	**3**	**2**	**10**	

Experiment No: 05

Aim:Write a program using TCP socket for wired network for following a. Say Hello to Each other b. File transfer

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 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(), aprogrammer 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 networkbyte 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 acall to listen(). Accepting incoming connections, via a call to accept(). This blocks until anincoming connection is received, and then returns a socket descriptor for theaccepted connection. The initial descriptor remains a listening descriptor, andaccept() can be called again at any time with this socket, until itis closed. Communicating with the remote host, which can be done through send() andrecv(). Eventually closing each socket that was opened, once it is no longer needed, using close(). Note that if there were any calls to fork(), eachprocess 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 thesame 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 structurewith the sin_family set to AF_INET or AF_INET6, sin_port set to the port theendpoint is listening (in network byte order), and sin_addr set to the IPv4 or IPv6address ofthe listening server (also in network byte order.) 1. Communicating with the server by send()ing and recv()ing.Terminating the connection and cleaning up with a call to close(). Again, if therewere any calls to fork(), each process must close() the socket. Functions: 1. 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: Computer Networks Lab 3 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 andSOCK_STREAM, UDP for those PF_ values and SOCK_DGRAM), but whichcan 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(intsockfd, conststructsockaddr *serv_addr, socklen_taddrlen); 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 onmy_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(intsockfd, structsockaddr *my_addr, socklen_taddrlen); 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(intsockfd, 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'saddress information into. addrlen, a pointer to the socklen_t integer that will indicate to accept() how largethe sockaddr structure pointed to by cliaddr is. When accept() returns, the ocklen_t integer then indicates how many bytes of the cliaddr structure wereactually used. The function returns a socket corresponding to the accepted connection, or -1 if an error occurs. Prototype: int accept(intsockfd, structsockaddr*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 DataInputStream is = newDataInputStream(client.getInputStream());

4. DataOutputStreamos=newDataOutputStream(client.getOutputStream());

5. Perform communication with client Receive from client: String line = is.readLine(); Sendto client: os.writeBytes("Hello\n")

6. 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 = newDataOutputStream(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 andthe 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.

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

SERVER SOCKET METHODS

Sr.No.	Method & Description
1	**s.bind()**This method binds address (hostname, port number pair) to socket.
2	**s.listen()**This method sets up and start TCP listener.
3	**s.accept()**This passively accept TCP client connection, waiting until connection arrives (blocking).

CLIENT SOCKET METHODS

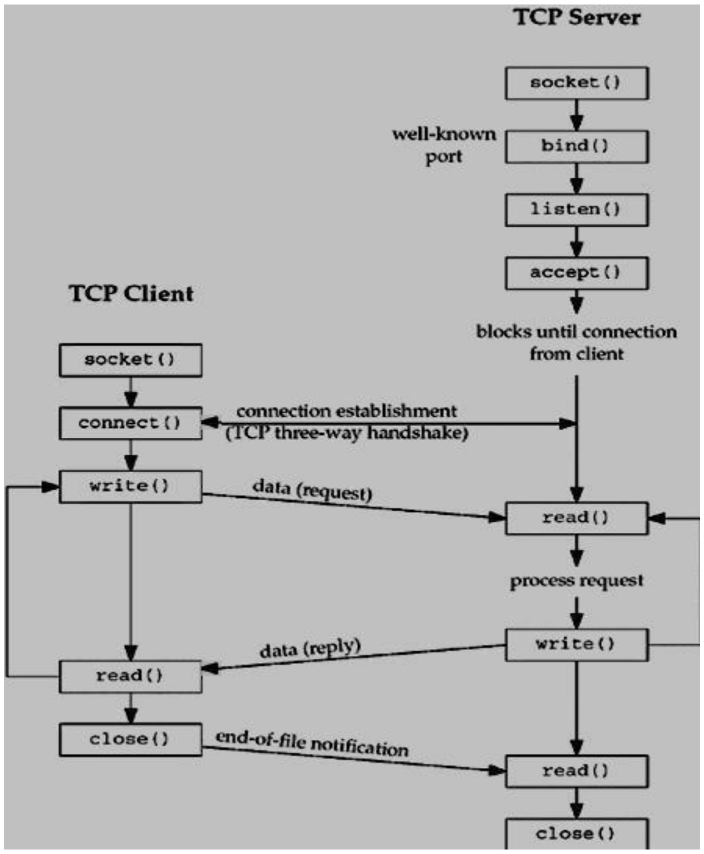
Sr.No.	Method & Description
1	**s.connect()**This mthod actively initiates TCP server connection.

GENERAL SOCKET METHODS

Sr.No.	Method & Description
1	**s.recv()**This method receives TCP message
2	**s.send()**This method transmits TCP message
3	**s.recvfrom()**This method receives UDP message

4	**s.sendto()**This method transmits UDP message
5	**s.close()**This method closes socket
6	**socket.gethostname()**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

Coding Efficiency	Viva	Timely Completion	Total	Dated Sign of Course In-charge
5	3	2	10	

Experiment No: 06

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

Theory:

Introduction

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 figur 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: Definitions for the most basic of socket structures with the BSD socket API Basic data types associated with structures within the BSD socket API Definitions for the socketaddr_in{} and other base data structures.

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 upa UDP server on port 7654 as follows:

```
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 connectionsqueue.

```
 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 theparameters: l socket l pointer to buffer for data l size of buffer l 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);s

 a.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 ofthe 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 theclient.
- Process request and send reply back with send(packet).

Conclusion: Thus we have studied Working of UDP Socket.

Coding Efficiency	Viva	Timely Completion	Total	Dated Sign of Course In-charge
5	3	2	10	

Experiment No: 07

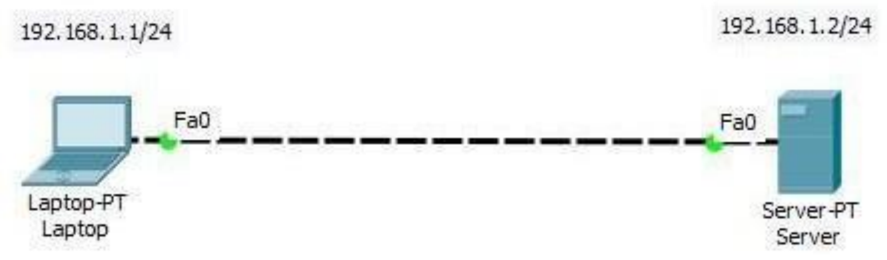
Aim : Study and Analyze the performance of HTTP, HTTPS and FTP protocol using Packet tracer tool.

Theory: 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.



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

Laptop: IP address: 192.168.1.1 Subnet Mask:

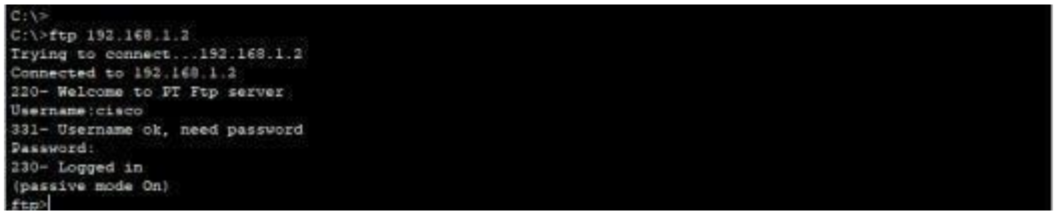
255.255.255.0 Server: IP address: 192.168.1.2

Subnet Mask: 255.255.255.0

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

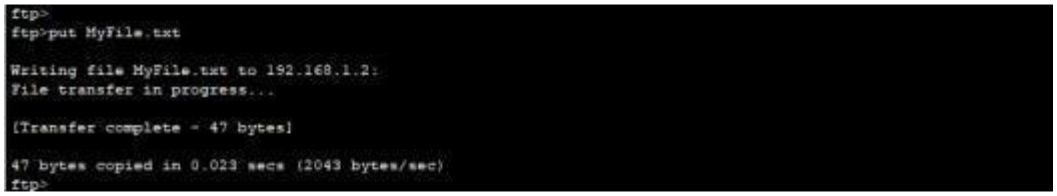
With that in mind, we can do something extra. So let's do this:

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

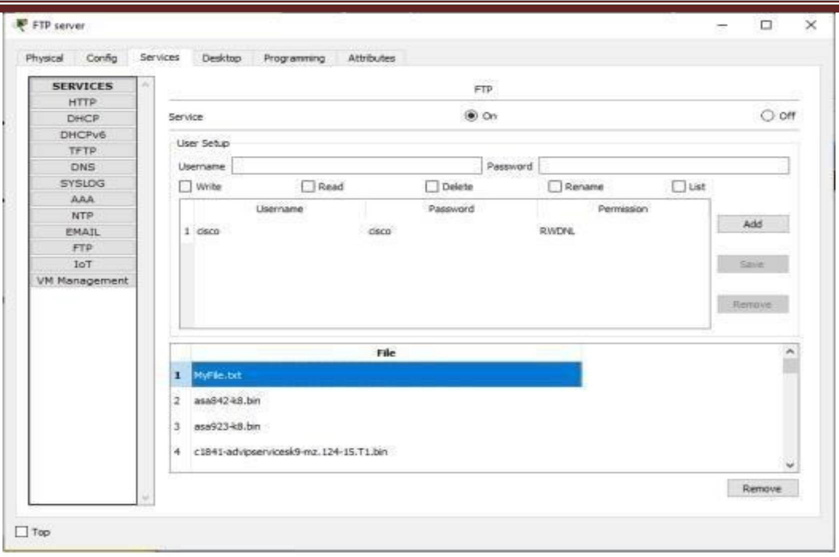

```
ftp>
ftp>put MyFile.txt

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>
```

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.

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.



FTP

Service (● On) ○ Off

User Setup

Username Password

☐ Write ☐ Read ☐ Delete ☐ Rename ☐ List

	Username	Password	Permission
1	cisco	cisco	RWDNL

	File
1	MyFile.txt
2	asa842-k8.bin
3	asa923-k8.bin
4	c1841-advipservicesk9-mz.124-15.T1.bin

For example: get MyFile.txt

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

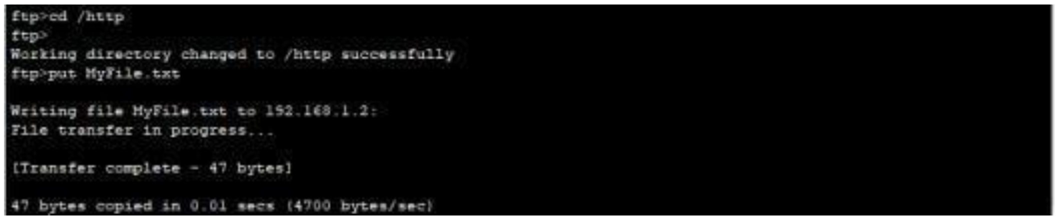
For example: delete MyFile.txt

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 changethe 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 uploadinga file to an HTTP folder(directory) using FTP.

For example: put MyFile.txt

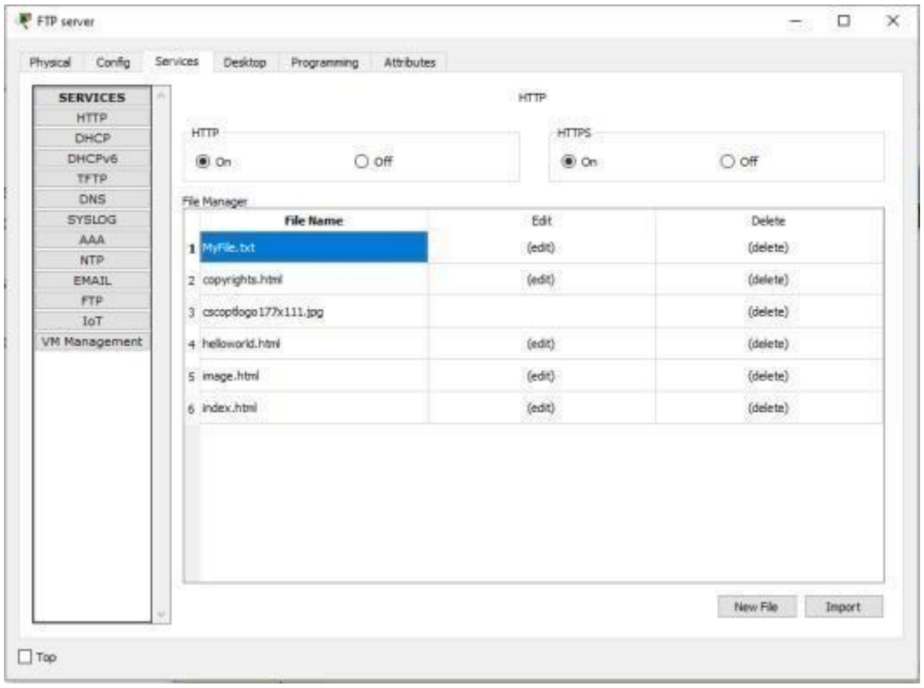


```
ftp>cd /http
ftp>
Working directory changed to /http successfully
ftp>put MyFile.txt

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 fromthe Laptop(MyFile.txt) is well received:



HTTP

HTTP: [✓ On] Off

HTTPS: [✓ On] Off

File Manager

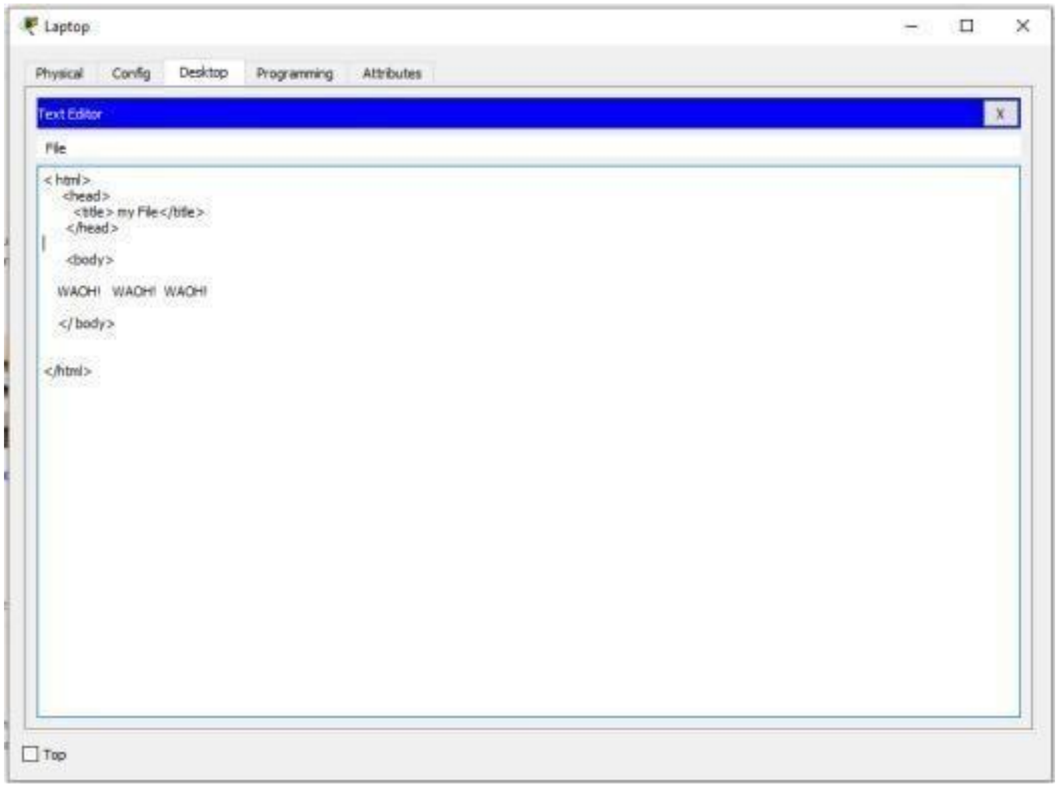
	File Name	Edit	Delete
1	MyFile.txt	(edit)	(delete)
2	copyrights.html	(edit)	(delete)
3	cscoptlogo177x111.jpg		(delete)
4	helloworld.html	(edit)	(delete)
5	image.html	(edit)	(delete)
6	index.html	(edit)	(delete)

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 developedbefore 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.

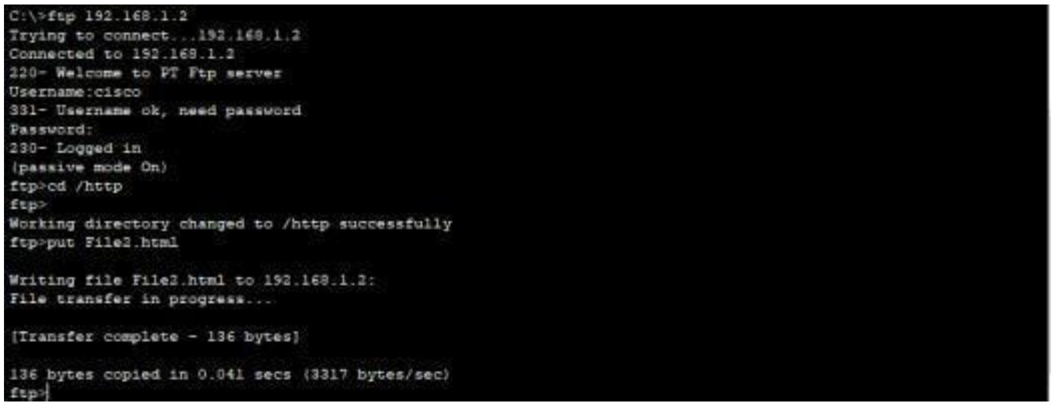
We'll create an html file in our Laptop, upload it to HTTP server directory using FTP, then try to access the file from the Laptop's browser.

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



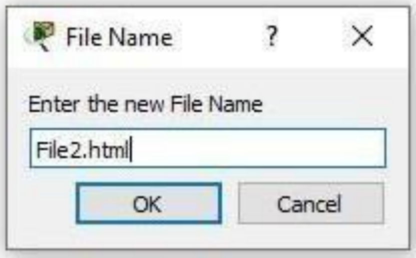
```
< html>
   <head>
     <title> my File</title>
   </head>

   <body>

 WAOH!  WAOH!  WAOH!

  </ body>


</html>
```

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:



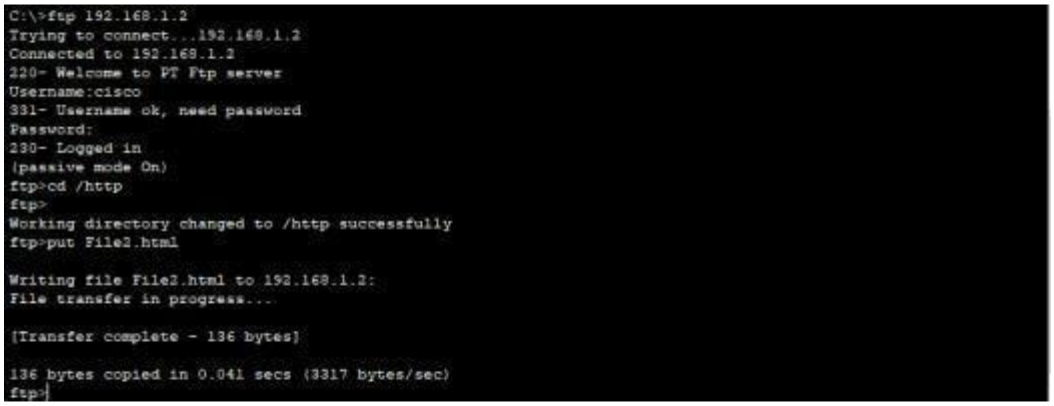
Enter the new File Name

File2.html

File2 html.PNG

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

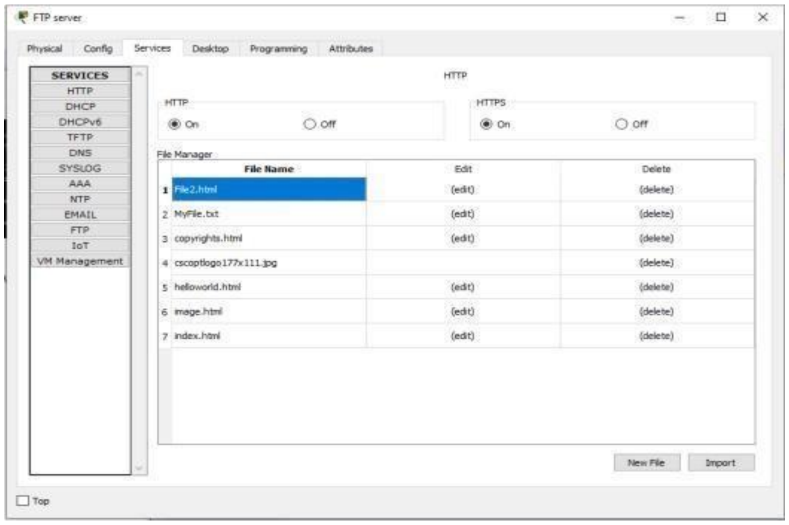
If you're already in the HTTP directory, you just need to type: put File2.html. If no, first ftp the server(ftp 192.168.1.2), provide the login username(cisco) and password(cisco); change the current directory to HTTP(cd /http) , and finally upload the html file onto the HTTP directory(put File2.html)



```
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 FileManager.



HTTP

HTTP: (●) On ( ) Off

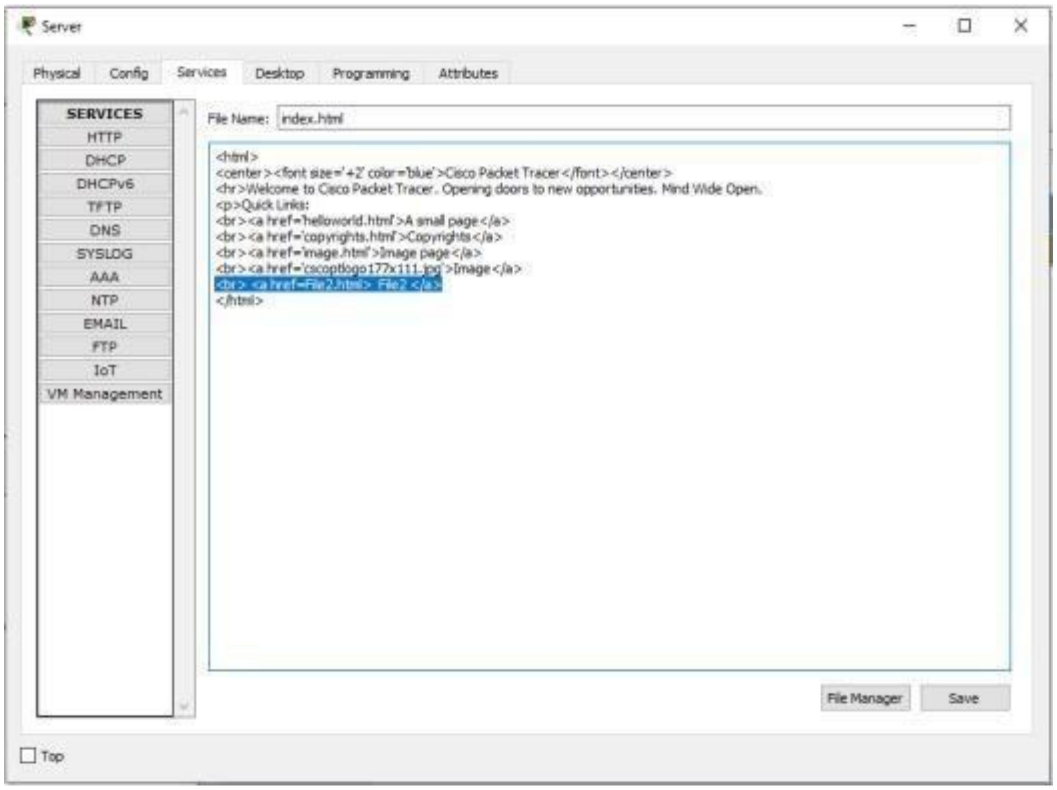
HTTPS: (●) On ( ) Off

File Manager

	File Name	Edit	Delete
1	File2.html	(edit)	(delete)
2	MyFile.txt	(edit)	(delete)
3	copyrights.html	(edit)	(delete)
4	cscoptlogo177x111.jpg		(delete)
5	helloworld.html	(edit)	(delete)
6	image.html	(edit)	(delete)
7	index.html	(edit)	(delete)

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.



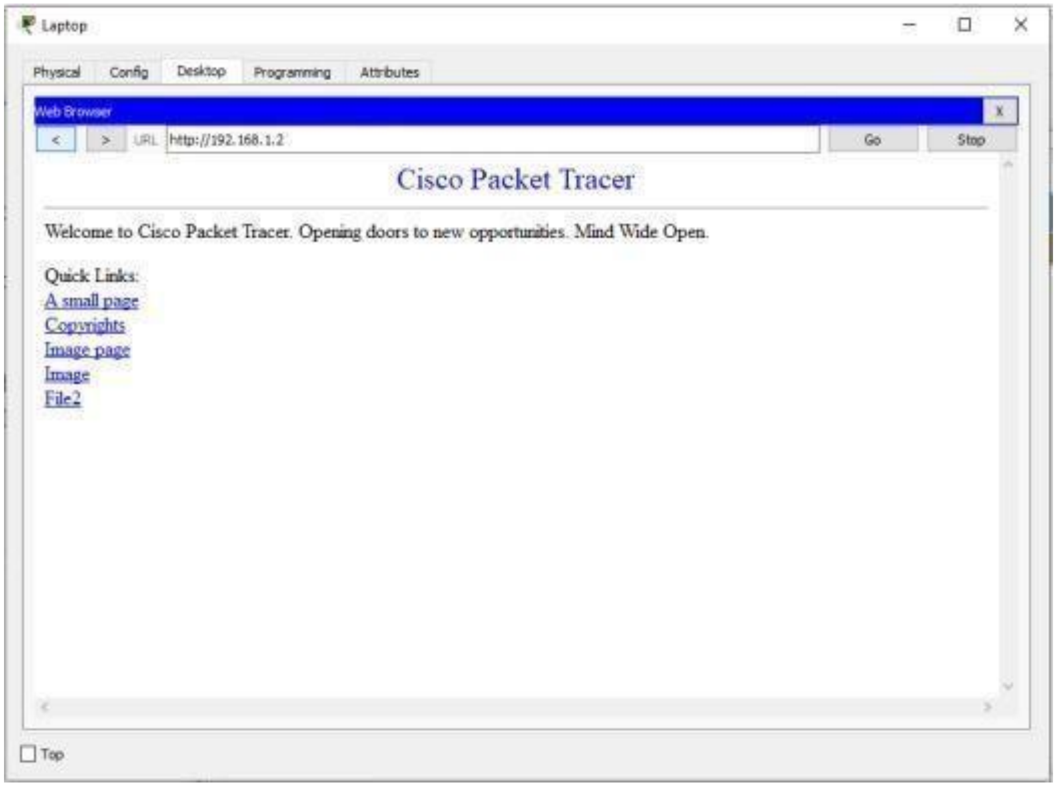
File Name: index.html

```
<html>
<center><font size='+2' color='blue'>Cisco Packet Tracer</font></center>
<hr>Welcome to Cisco Packet Tracer. Opening doors to new opportunities. Mind Wide Open.
<p>Quick Links:
<br><a href='helloworld.html'>A small page</a>
<br><a href='copyrights.html'>Copyrights</a>
<br><a href='image.html'>Image page</a>
<br><a href='cscoptlogo177x111.jpg'>Image</a>
<br> <a href=File2.html> File2 </a>
</html>
```

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



http://192.168.1.2

Cisco Packet Tracer

Welcome to Cisco Packet Tracer. Opening doors to new opportunities. Mind Wide Open.

Quick Links:
A small page
Copyrights
Image page
Image
File2

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

Conclusion:: Successfully studied and Analyze the performance of HTTP, HTTPS and FTPprotocol using Packet tracer tool.

Coding Efficiency	Viva	Timely Completion	Total	Dated Sign of Course In-charge
5	3	2	10	

Experiment No: 08

Aim: To study the SSL protocol by capturing the packets using Wireshark tool while visiting any SSL securedwebsite (banking, e-commerce etc.)

Theory:

SSL, or Secure Sockets Layer, is an encryption-based Internet security protocol. It was first developed by Netscape in 1995 for the purpose of ensuring privacy, authentication, and data integrity in Internet communications. SSL is the predecessor to the modern TLS encryption used today.

How does SSL/TLS work?

In order to provide a high degree of privacy, SSL encrypts data that is transmitted across the web. This means that anyone who tries to intercept this data will only see a garbled mix of characters that is nearly impossible to decrypt.

SSL initiates an authentication process called a handshake between two communicating devices to ensure that both devices are really who they claim to be.

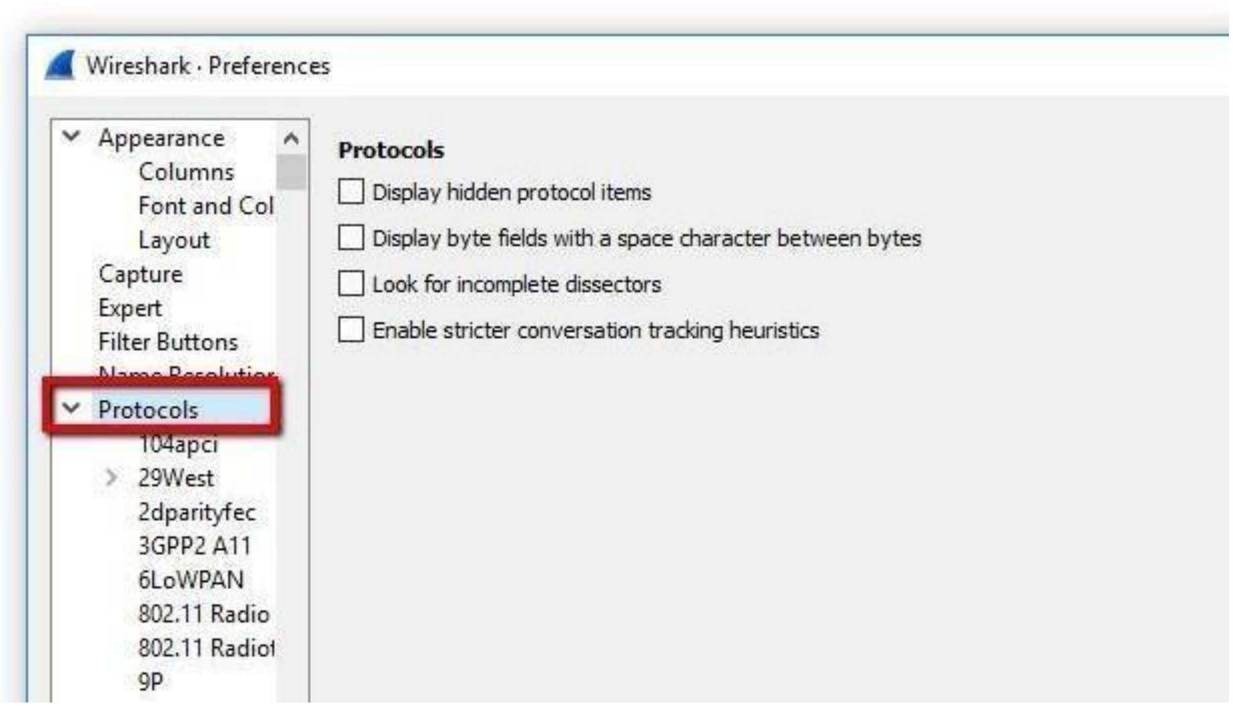
SSL also digitally signs data in order to provide data integrity, verifying that the data is not tampered with before reaching its intended recipient.

There have been several iterations of SSL, each more secure than the last. In 1999 SSL was updated to become TLS.

Why is SSL/TLS important?

Originally, data on the Web was transmitted in plaintext that anyone could read if they intercepted the message. For example, if a consumer visited a shopping website, placed an order, and entered their credit card number on the website, that credit card number would travel across the Internet unconcealed.

Configure Wireshark to decrypt SSL



Protocols
- [ ] Display hidden protocol items
- [ ] Display byte fields with a space character between bytes
- [ ] Look for incomplete dissectors
- [ ] Enable stricter conversation tracking heuristics

Once your browser is logging pre-master keys, it's time to configure Wireshark to use those logs to decrypt SSL.

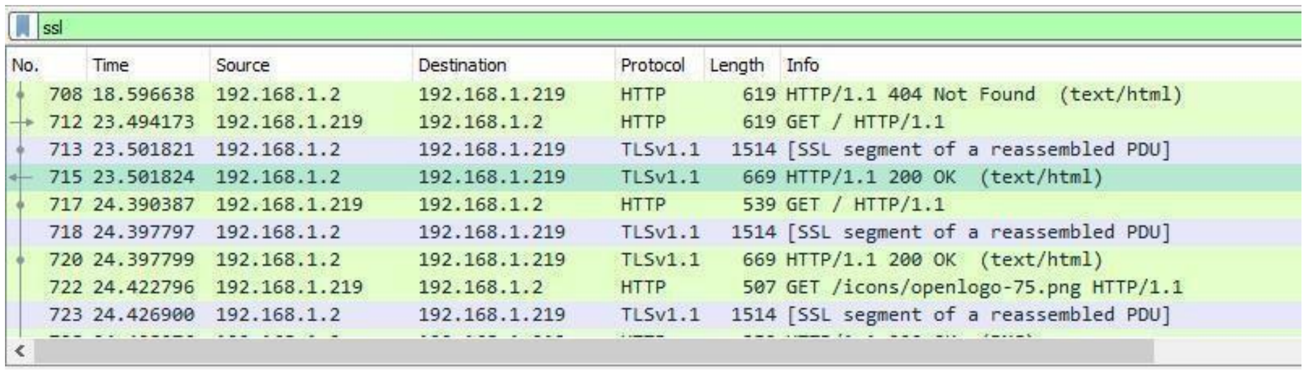
Open Wireshark and click **Edit**, then **Preferences**. The **Preferences** dialog will open, and on the left, you'll see a list of items. Expand **Protocols**, scroll down, then click **SSL**.

Capture the session and decrypt SSL

The final step is to capture a test session and make sure that Wireshark decrypts SSL successfully.

- Start an unfiltered capture session, minimize it, and open your browser.
- Visit a secure site in order to generate data, and optionally set a display filter of 'ssl' to minimize the session noise.
- Click on any frame containing encrypted data.

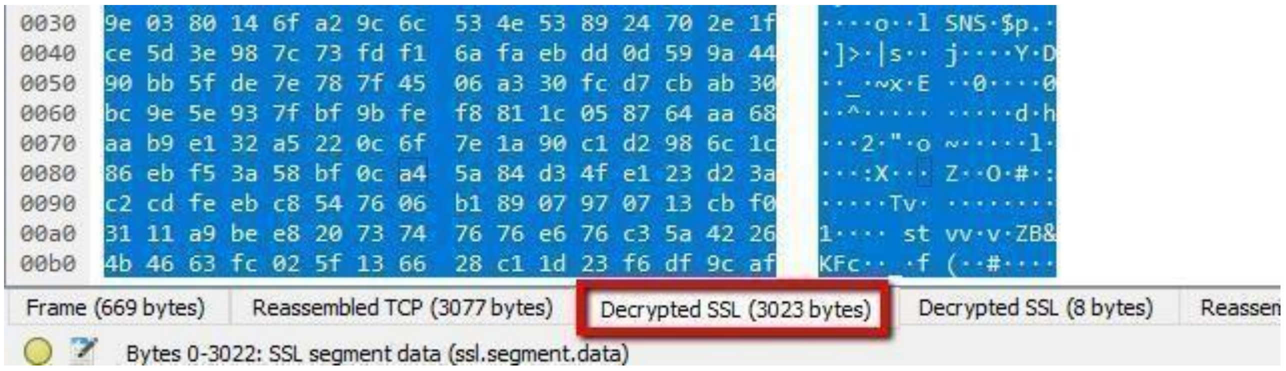
In my case, I'll select one that contains HTTP traffic with text/HTML encoding, since I'd like to see the source code the web server is sending to my browser. But any encrypted transmissions that use a pre-master secret or private key will work with this method. That includes all data utilizing Perfect Forward Encryption (PFE) through Diffie-Hellman or comparable key exchanges.



ssl

```
No.   Time       Source         Destination    Protocol Length Info
 708 18.596638  192.168.1.2    192.168.1.219  HTTP        619 HTTP/1.1 404 Not Found  (text/html)
 712 23.494173  192.168.1.219  192.168.1.2    HTTP        619 GET / HTTP/1.1
 713 23.501821  192.168.1.2    192.168.1.219  TLSv1.1    1514 [SSL segment of a reassembled PDU]
 715 23.501824  192.168.1.2    192.168.1.219  TLSv1.1     669 HTTP/1.1 200 OK  (text/html)
 717 24.390387  192.168.1.219  192.168.1.2    HTTP        539 GET / HTTP/1.1
 718 24.397797  192.168.1.2    192.168.1.219  TLSv1.1    1514 [SSL segment of a reassembled PDU]
 720 24.397799  192.168.1.2    192.168.1.219  TLSv1.1     669 HTTP/1.1 200 OK  (text/html)
 722 24.422796  192.168.1.219  192.168.1.2    HTTP        507 GET /icons/openlogo-75.png HTTP/1.1
 723 24.426900  192.168.1.2    192.168.1.219  TLSv1.1    1514 [SSL segment of a reassembled PDU]
```

Once you've selected an encrypted data frame, look at the **Packet byte view**, and specifically the tabs underneath the view. You should see an entry for **Decrypted SSL** data, among others.

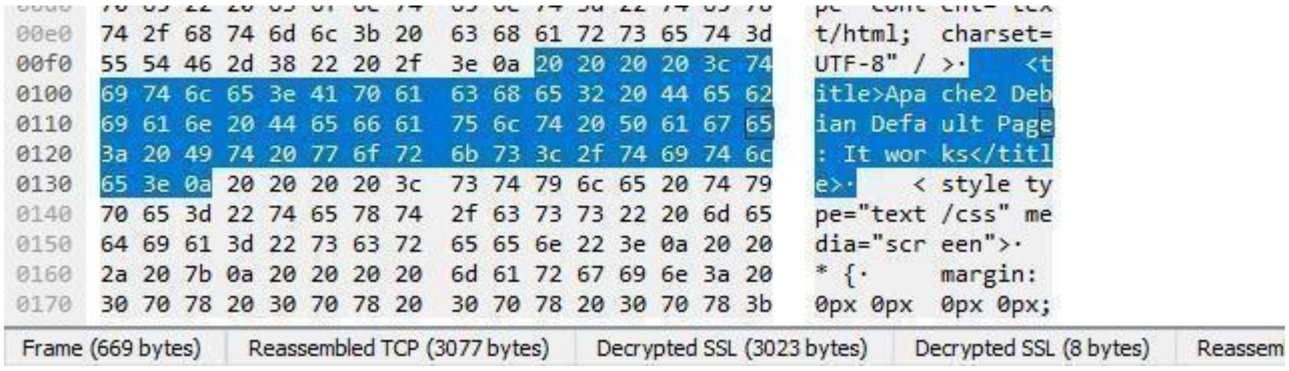


```
0030  9e 03 80 14 6f a2 9c 6c  53 4e 53 89 24 70 2e 1f   ····o··l SNS·$p.·
0040  ce 5d 3e 98 7c 73 fd f1  6a fa eb dd 0d 59 9a 44   ·]>·|s·· j····Y·D
0050  90 bb 5f de 7e 78 7f 45  06 a3 30 fc d7 cb ab 30   ··_·~x·E ··0····0
0060  bc 9e 5e 93 7f bf 9b fe  f8 81 1c 05 87 64 aa 68   ··^····· ·····d·h
0070  aa b9 e1 32 a5 22 0c 6f  7e 1a 90 c1 d2 98 6c 1c   ···2·"·o ~·····l·
0080  86 eb f5 3a 58 bf 0c a4  5a 84 d3 4f e1 23 d2 3a   ···:X··· Z··O·#·:
0090  c2 cd fe eb c8 54 76 06  b1 89 07 97 07 13 cb f0   ·····Tv· ········
00a0  31 11 a9 be e8 20 73 74  76 76 e6 76 c3 5a 42 26   1···· st vv·v·ZB&
00b0  4b 46 63 fc 02 5f 13 66  28 c1 1d 23 f6 df 9c af   KFc··_·f (··#····
```

Frame (669 bytes) | Reassembled TCP (3077 bytes) | Decrypted SSL (3023 bytes) | Decrypted SSL (8 bytes) | Reassem

Bytes 0-3022: SSL segment data (ssl.segment.data)

You'll notice that my session still looks like it's full of garbage, and no HTML is visible. That's because my web server (and most Apache servers) use GZIP compression by default.

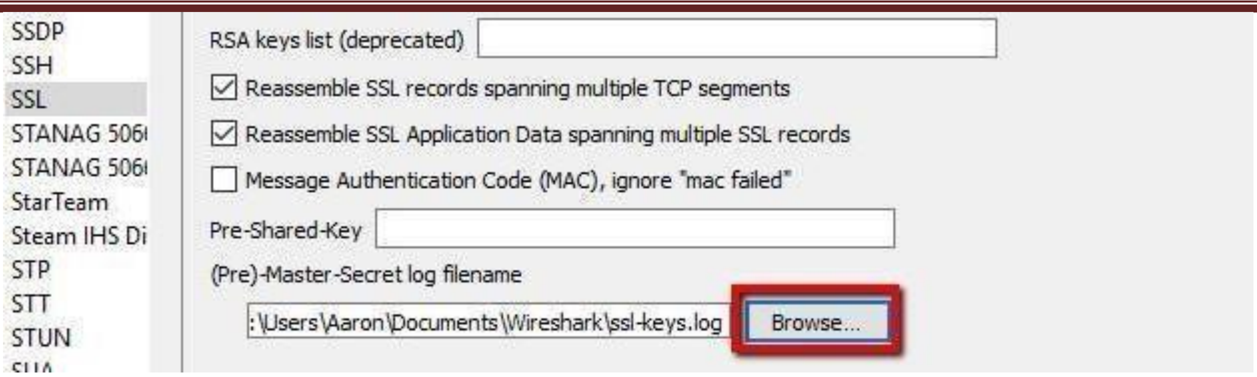


```
00e0  74 2f 68 74 6d 6c 3b 20  63 68 61 72 73 65 74 3d   t/html;  charset=
00f0  55 54 46 2d 38 22 20 2f  3e 0a 20 20 20 20 3c 74   UTF-8" / >·    <t
0100  69 74 6c 65 3e 41 70 61  63 68 65 32 20 44 65 62   itle>Apa che2 Deb
0110  69 61 6e 20 44 65 66 61  75 6c 74 20 50 61 67 65   ian Defa ult Page
0120  3a 20 49 74 20 77 6f 72  6b 73 3c 2f 74 69 74 6c   : It wor ks</titl
0130  65 3e 0a 20 20 20 20 3c  73 74 79 6c 65 20 74 79   e>·    < style ty
0140  70 65 3d 22 74 65 78 74  2f 63 73 73 22 20 6d 65   pe="text /css" me
0150  64 69 61 3d 22 73 63 72  65 65 6e 22 3e 0a 20 20   dia="scr een">·
0160  2a 20 7b 0a 20 20 20 20  6d 61 72 67 69 6e 3a 20   * {·     margin:
0170  30 70 78 20 30 70 78 20  30 70 78 20 30 70 78 3b   0px 0px  0px 0px;
```

Frame (669 bytes) | Reassembled TCP (3077 bytes) | Decrypted SSL (3023 bytes) | Decrypted SSL (8 bytes) | Reassem

When you click the **Uncompressed entity body** tab, which only shows up in this case with SSL decryption enabled, you can view the source code of the site. For instance, here's the title element of the default Apache page in plaintext.

In the list of options for the SSL protocol, you'll see an entry for **(Pre)-Master-Secret log filename**. Browse to the log file you set up in the previous step, or just paste the path.



RSA keys list (deprecated)
☑ Reassemble SSL records spanning multiple TCP segments
☑ Reassemble SSL Application Data spanning multiple SSL records
☐ Message Authentication Code (MAC), ignore "mac failed"
Pre-Shared-Key
(Pre)-Master-Secret log filename
:\Users\Aaron\Documents\Wireshark\ssl-keys.log

When you've finished setting the **(Pre)-Master-Secret log filename**, click **OK** and return to Wireshark.

Conclusion:

Successfully studied the SSL protocol by capturing the packets using Wireshark tool while visiting any SSL secured website (banking, e-commerce etc.)

Coding Efficiency	Viva	Timely Completion	Total	Dated Sign of Course In-charge
5	3	2	10	

Experiment No: 09

Aim: Illustrate the steps for implementation of S/MIME email security through Microsoft® Office Outlook.

Theory

MIME (Secure/Multipurpose Internet Mail Extensions)

S/MIME allows users to send encrypted and digitally signed emails . This protocol allows recipients of the email to be certain the email they receive is the exact message that began with the sender. It also helps ensure that a message going to an outbound recipient is from a specific sender and not someone assuming a false identity.

How does S/MIME work?

S/MIME provides cryptographic-based security services like authentication, message integrity, and digital signatures. All these elements work together to enhance privacy and security for both the sender and recipient of an email.

S/MIME also works with other technologies such as Transport Layer Security (TLS) which encrypts the path between two email servers. The protocol is also compatible with Secure Sockets Layer (SSL) which masks the connection between email messages and Office 365 (a common email service) servers.

In addition, BitLocker works in conjunction with S/MIME protocol, which encrypts data on a hard drive in a data center so if a hacker gets access, he or she won't be able to interpret the information.

Benefits of encrypted email

1. **Safeguards sensitive data**

If you're sending information like your Social Security number over email, it's important that it'snot easily stolen by hackers.

2. **Economical**

Instead of purchasing security equipment, you can simply rely on email encryption that's integrated directly on the server.

3. **Timesaving**

Instead of wasting time using several programs to make sure a connection is secure, you can rely on email encryption to do most of the work for you.

4. **Regulation compliance**

If you work in the healthcare industry, for example, and you haven't taken the right steps to secure medical data, you could be in violation of HIPAA laws [6]. Encryption helps you avoid those missteps.

5. **Protects against malware**

Malicious emails sometimes contain viruses masked as innocent email attachments. If you or someone else send an attachment using encrypted email, the email has a digital signature toprove its authenticity.

How does email encryption work?

If you don't want anyone but the receiver to see the contents of a message, encryption is vital. Tothe outsider, an encrypted email will have a bunch of random letters, digits, or symbols instead ofreadable text. The person with the private key to decrypt it, typically the receiver, will be able to read the email as usual.

There are generally three encryption types available:

- S/MIME encryption works as long as both the sender and recipient have mailboxes that support it. Windows Outlook is the most popular version that works with this method. Gmail uses it as well.
- Office 365 Message Encryption is best for users with valid Microsoft Office licenses who can use this tool to encrypt the information and files
- sent via email. It's also a top choice for Outlook users
- PGP/MIME is a more affordable and popular option that other email clients may preferto use. It's reliable and integrated into many of the apps we use today

Other email products may have their own brand of encryption, but the science behind it is the same. Only senders and recipients who have exchanged keys or digital signatures can communicate within the encrypted network.

How to send encrypted email in Outlook

Encrypting email may sound complicated, but it's not. Microsoft has a reputation for providing its users with simple ways to encrypt data, from files to folders to emails, too. It makes sense that they would include built-in tools for Outlook, their proprietary email system. You don't need a separate software tool or plug-in to start sending secure messages. Just follow these steps to begin.

1. Create a digital certificate

For Outlook users, encrypting a single email is simple. First, you must have a digital signature. To create a digital signature:

1. Start in your Outlook window and click on the File tab
2. Select Options, then Trust Center, then Trust Center Settings
3. Select Email Security, Get a Digital ID
4. You'll be asked to choose a certification authority. This is entirely up to you as mostare rated the same
5. You'll receive an email with your digital certificate/ID included
6. Go back into Outlook and select Options and the Security tab
7. In the Security Settings Name field, type in a name of your choosing
8. Ensure that S/MIME is selected from the Secure Message Format box and that Default Security Settings is checked as well

9. Go to Certificates and Algorithms, select Signing Certificate, and click Choose
10. Make sure the box is checked next to Secure Email Certificate, and check the box nextto "Send These Certificates with Signed Messages"
11. Click OK to save your settings and start using Outlook

2. Use your digital signature

Now that you have a digital ID, you need to start using it:

1. Open a new message to access the Tools tab
2. Click that, then Customize, and finally the Commands tab
3. From Categories, select Standard
4. From Categories, select Digitally Sign Message

3. Encrypt Outlook messages

You can now send encrypted messages to a recipient with the next steps.

1. Open the window to compose a new message and select the Options tab, then MoreOptions
2. Click the dialog box (triangle with arrow pointing down) in the lower-right corner
3. Choose Security Settings and check the box next to Encrypt message contents andattachments
4. Write your message as normal and send

After you've sent and received a message that you've both signed and encrypted, you don't haveto sign it again. Outlook will remember your signature.

4. Encrypt all Outlook messages

You can encrypt each one, or you can use the steps below to encrypt all outgoing messages in Outlook:

- Open the File tab in Outlook
- Select Options, then Trust Center, and Trust Center Settings
- From the Email Security tab, select Encrypted email

- Check the box next to Encrypt content and attachments for outgoing messages
- Use Settings to customize additional options, including certificates

How to Send Encrypted Email

Have you ever wondered about the security of your private email conversations? Whether at work, school, or home, sending emails comes with a bit of a risk. There's one thing you can do to discourage data breaches and attacks on your sensitive data, however. Use encrypted email. Learn how to practice this common-sense method for communicating in our step-by-step guide. But first, let's look at why you should embrace encryption for your email correspondence.

How to Encrypt Email and Send Secure Messages

Emails sent over an open network can be intercepted and malicious actors can see email contents, attachments, or even take over your account.To drive home the importance of email security, take a look at some alarming statistics that show the widespread cybersecurity issues that may have affected you in the past and still pose a threat today:In 2016, 3 billion Yahoo accounts were hackedAccording to research by cybersecurity company, Symantec, emails with a malicious URL make up a total of 12.3% of all emails As these numbers illustrate, emails are a point of vulnerability for many unsuspecting users. However, it's not all doom and gloom, there are ways to protect yourself and your information.To help safeguard against hackers and ensure your privacy is maintained, you can use encryption. Encryption ensures that your emails remain unreadable, even if they fall into the wrong hands.

Conclusion: Thus we have studied the steps for implementation of S/MIME email security through Microsoft® Office Outlook.ets received from / sent to Face book, and how many of eachwere also HTTP packets successfully.

Coding Efficiency	Viva	Timely Completion	Total	Dated Sign of Course In-charge
5	3	2	10	

Experiment No: 10

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

Theory:

The IP security (IPSec) is an Internet Engineering Task Force (IETF) standard suite of protocols between 2 communication points across the IP network that provide data authentication, integrity, and confidentiality. It also defines the encrypted, decrypted and authenticated packets. The protocols needed for seucre key exchange and key management are defined in it.

What Ports Does IPSEC Operate On?

UDP port 500 should be opened as should IP protocols 50 and 51. UDP port 500 should be opened to allow for ISAKMP to be forwarded throughthe firewall while protocols 50 and 51 allow ESP and AH traffic to be forwarded respectively.2

What is ISAKMP?

ISAKMP stands for Internet Security Association andKey Management

Protocol. These are two key components of an IPSEC VPN that must be ipnlace in order for it to function normally and protect the public traffic that is being forwadred between the client and VPN server or VPN server to

VPN server.

What are ESP and AH?

No, ESP is not Extra-Sensory Perception! ESP stansdfor Encapsulating Security Protocol and AH stands for Authentication Header.

Encapsulating Security Protocol

ESP gives protection to upper layer new protocols, with a Signed area indicating where a protected data packet has been signed for integrity, and an Encrypted area which indicates the information that's protected with confidentiality. Unless a data packet is being tunneled, ESP protects only the IP data payload (hence the name), and not the IP header.

ESP may be used to ensure confidentiality, the authentication of data origins, connectionless integrity, some degree of traffic-level confidentiality, and an anti-replay service (a form of partial sequence integrity which guards against the use of commands or credentials which have been captured through password sniffing or similar attacks).3

Authentication Header

Authentication Header (AH) is a new protocol and prat of the Internet Protocol Security (IPsec) protocol suite, which authenticates the origin of I P packets (datagrams) and guarantees the integrity of the data. The AH confirms the originating source of a packet and ensures that its contents (both the header and payload) have not been changed since transmission.

If security associations have been established, AH can be optionally configured to defend against replay attacks using the sliding window technique.4

How Do They All Work Together?

When properly configured, an IPSEC VPN provides mutliple layers of security that ensure the security mode and integrity of the data that is being transmitted through the encrypted tunnel. This way an organization can feel confident that the data has not been intercepted and altered in transit and that they can rely on what they are seeing.

AH and ESP protocols

IPSec uses two distinct protocols, Authentication Header (AH) and Encapsulating SecurityPayload (ESP), which are defined by the IETF.
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. AHauthenticates 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.

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.

Either protocol can be used alone to protect an IP packet, or both protocols can be applied together to the same IP packet. The choice of IPSec protocol is determined by the security needs of your installation, and is configured by the administrator. It does not have to be applied system-wide, and can be configured differently for each set of connection endpoints. For a dynamic tunnel, the choice of IPSec protocol is configured using the IpDataOffer statement in anIP security policy configuration file. For a manual tunnel, the choice of IPSec protocol isconfigured using the IpManVpnAction statement in an IP security policy configuration file. For more details about the IpDataOffer statement and the IpManVpnAction statement, see z/OS Communications Server: IP Configuration Reference.

Conclusion: Successfully studied the IPsec (ESP and AH) protocol by capturing the packets using Wireshark tool.

Coding Efficiency	Viva	Timely Completion	Total	Dated Sign of Course In-charge
5	3	2	10	

Experiment No: 11

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

Theory:

Dynamic Host Configuration Protocol:

The Dynamic Host Configuration Protocol (DHCP) is a client/server protocol designed to provide the four pieces of information for a diskless computer or a computer that is booted for the first time. DHCP is a successor to BOOTP and is backward compatible with it. Although BOOTP is considered deprecated, there may be some systems that may still use BOOTP for host configuration. The part of the discussion in this chapter that does not deal with the dynamic aspect of DHCP can also be applied to BOOTP. The DHCP client and server can either be on the same network or on different networks. Let us discuss each situation separately.

Same Network:

Although the practice is not very common, the administrator may put the client and the server on the same network as shown in Figure 1.

In this case, the operation can be described as follows:

1. The DHCP server issues a passive open command on UDP port number 67 and waits fora client.

2. A booted client issues an active open command on port number 68. The message is encapsulated in a UDP user datagram, using the destination port number 67 and the source port number 68. The UDP user datagram, in turn, is encapsulated in an IP datagram. The reader may ask how a client can send an IP datagram when it knows neither its own IP address (the source address) nor the server's IP address (the destination address). The client uses all 0s as the source address and all 1s as the destination address.

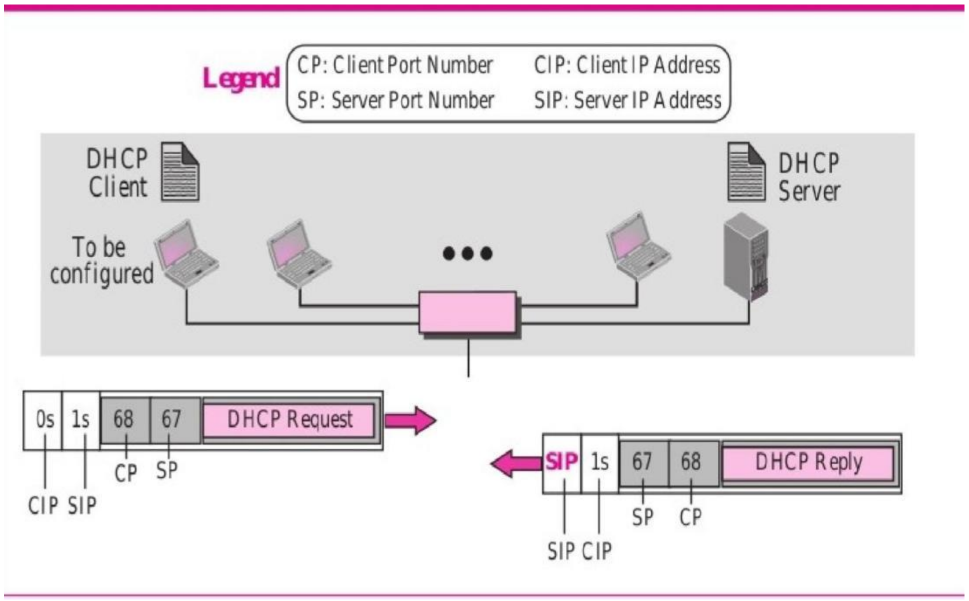


Figure 1: Client and server on the same network

3. The server responds with either a broadcast or a unicast message using UDP source port number 67 and destination port number 68. The response can be unicast because the server knows the IP address of the client. It also knows the physical address of the client, which means it does not need the services of ARP

for logical to physical address mapping. However, some systems do not allow the bypassing of ARP, resulting in the use of the broadcast address.

Different Networks:

As in other application-layer processes, a client can be in one network and the server in another, separated by several other networks. Figure2 shows the situation.

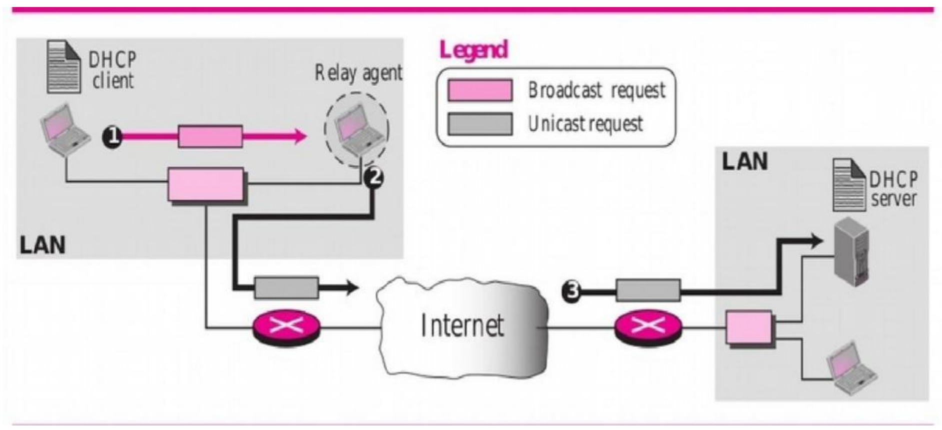


Figure 2 : Client and server on two different networks

However, there is one problem that must be solved. The DHCP request is broadcast because the client does not know the IP address of the server. A broadcast IP datagram cannot pass through any router. A router receiving such a packet discards it. Recall that an IP address of all 1s is a limited broadcast address. To solve the problem, there is a need for an intermediary. One of the hosts (or a router that can be configured to operate at the application layer) can be used as a relay. The host in this

case is called a relay agent. The relay agent knows the unicast address of a DHCP server and listens for broadcast messages on port 67. When it receives this type of packet, it encapsulates the message in a unicast datagram and sends the request to the DHCP server. The packet, carrying a unicast destination address, is routed by any router and reaches the DHCP server. The DHCP server knows the message comes from a relay agent because one of the fields in the request message defines the IP address of the relay agent. The relay agent, after receiving the reply, sends it to the DHCP client.

DHCP packet format:

The following briefly describes each field:

- ❑ Operation code. This 8-bit field defines the type of DHCP packet: request (1) or reply (2).
- ❑ Hardware type. This is an 8-bit field defining the type of physical network. Each type of network has been assigned an integer. For example, for Ethernet the value is 1.
- ❑ Hardware length. This is an 8-bit field defining the length of the physical address inbytes. For example, for Ethernet the value is 6.
- ❑ Hop count. This is an 8-bit field defining the maximum number of hops thepacket can travel.
- ❑ Transaction ID. This is a 4-byte field carrying an integer. The transaction identification isset by the client and is used to match a reply with the request. The server returns the same value in its reply.
- ❑ Number of seconds. This is a 16-bit field that indicates the number of seconds elapsed since the time the client started to boot.
- ❑

❑ Flag. This is a 16-bit field in which only the leftmost bit is used and the rest of the bits should be set to 0s. A leftmost bit specifies a forced broadcast reply (instead of unicast) from the server. If the reply were to be unicast to the client, the destination IP address of the IP packet is the address assigned to the client. Since the client does not know its IP address, it may discard the packet. However, if the IP datagram is broadcast, every host will receive and process the broadcast message.

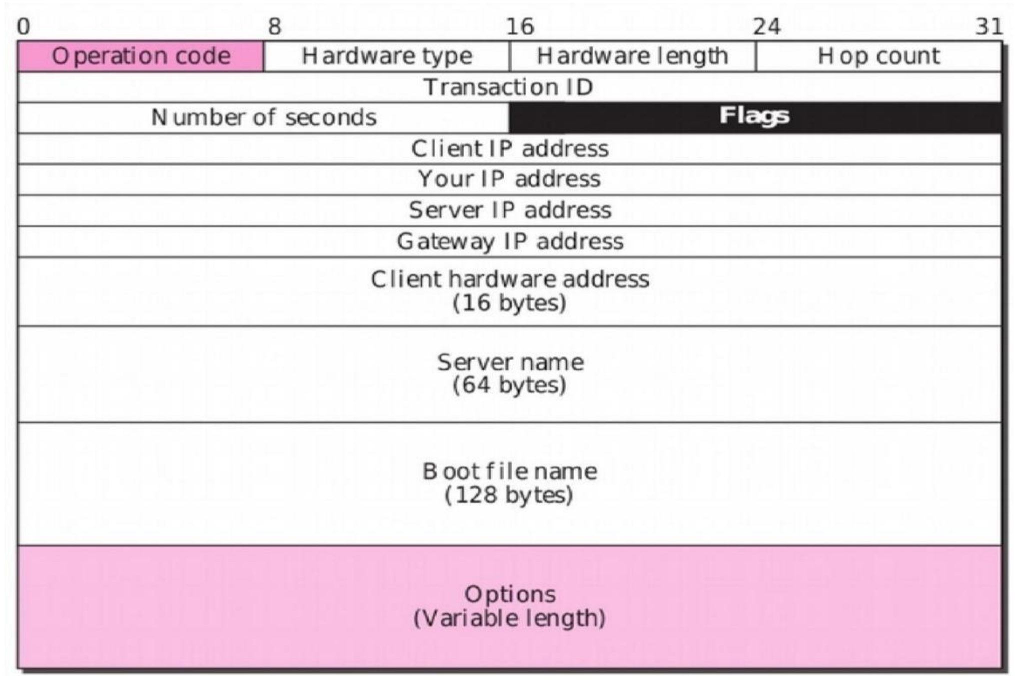


Figure 3 :DHCP packet format

Flag Format:

❑ Client IP address. This is a 4-byte field that contains the client IP address. If the client does not have this information, this field has a value of 0.

- ❑ Your IP address. This is a 4-byte field that contains the client IP address. It is filled by theserver (in the reply message) at the request of the client.
- ❑ Server IP address. This is a 4-byte field containing the server IP address. It is filled by theserver in a reply message.
- ❑ Gateway IP address. This is a 4-byte field containing the IP address of a router. It is filledby the server in a reply message.
- ❑ Client hardware address. This is the physical address of the client. Although the server can retrieve this address from the frame sent by the client, it is more efficient if the address is supplied explicitly by the client in the request message.
- ❑ Server name. This is a 64-byte field that is optionally filled by the server in a reply packet. It contains a null-terminated string consisting of the domain name of th server. If the server does not want to fill this field with data, the server must fill it with all 0s.
- ❑ Boot filename. This is a 128-byte field that can be optionally filled by the server in a reply packet. It contains a null-terminated string consisting of the full pathname of the boot file. The client can use this path to retrieve other booting information. If the server does not want to fill this field with data, the server must fill it with all 0s.
- ❑ Options. This is a 64-byte field with a dual purpose. It can carry either additional information (such as the network mask or default router address) or some specific vendor information. The field is used only in a reply message. The server uses a number, called a magic cookie, in the format of an IP address with the value of 99.130.83.99. When the client finishes reading the message, it looks for this magic cookie. If present, the next 60 bytes are options. An option is composed of three fields: a 1-byte tag field, a 1-byte length field, and a variable-

length value field. The length field defines the length of the value field, not the whole option.

DHCP client transition:

INIT State :

When the DHCP client first starts, it is in the INIT state (initializing state). The client broadcasts a DHCPDISCOVER message (a request message with the DHCPDISCOVER option), using port 67.

SELECTING State:

After sending the DHCPDISCOVER message, the client goes to the selecting state. Those servers that can provide this type of service respond with a DHCPOFFER message. In these messages, the servers offer an IP address. They can also offer the lease duration. The default is 1 hour. The server that sends a DHCPOFFER locks the offered IP address so that it is not available to any other clients. The client chooses one of the offers and sends a DHCP REQUEST message to the selected server. It then goes to the requesting state. However, if the client receives no DHCPOFFER message, it tries four more times, each with a span of 2 seconds. If there is no reply to any of these DHCP DISCOVER s, the client sleeps for 5 minutes before trying again.

REQUESTING State:

The client remains in the requesting state until it receives a DHCPACK message from the server that creates the binding between the client physical address and its IP address. After receipt ofthe DHCPACK, the client goes to the bound state.

BOUND State:

In this state, the client can use the IP address until the lease expires. When 50 percent of the lease period is reached, the client sends another DHCPREQUEST to ask for renewal. It then goes to the renewing state. When in the bound state, the client can also cancel the lease and go to the initializing state.

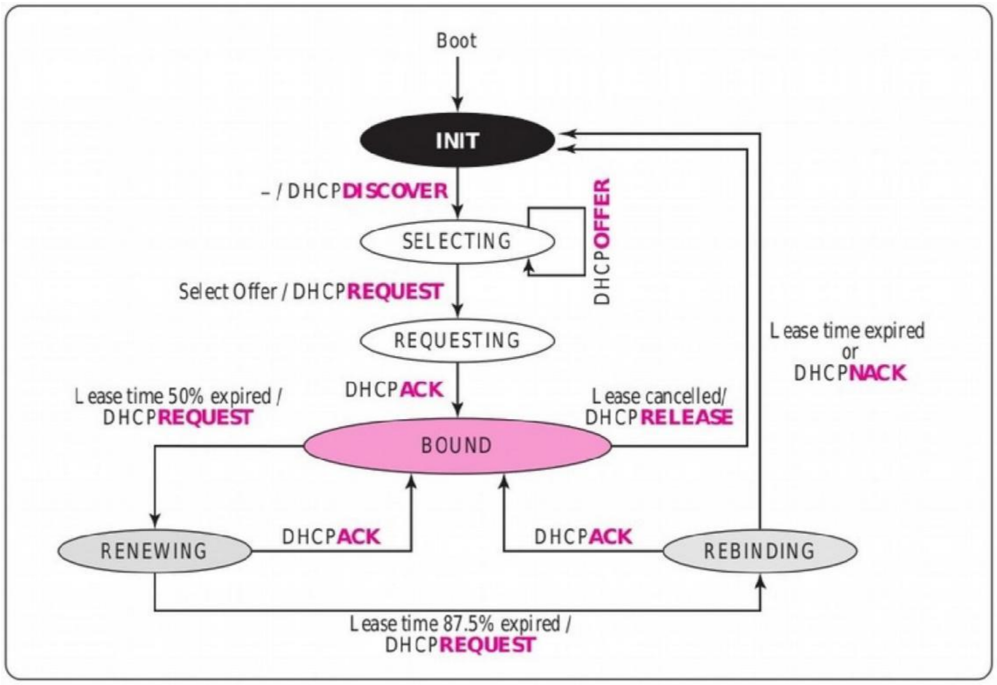

Boot
INIT
– / DHCPDISCOVER
SELECTING
DHCPOFFER
Select Offer / DHCPREQUEST
REQUESTING
DHCPACK
Lease time expired or DHCPNACK
Lease time 50% expired / DHCPREQUEST
Lease cancelled/ DHCPRELEASE
BOUND
RENEWING
DHCPACK
DHCPACK
REBINDING
Lease time 87.5% expired / DHCPREQUEST

Figure 4: DHCP client transition

RENEWING State:

The client remains in the renewing state until one of two events happens. It can receive a DHCPACK, which renews the lease agreement. In this case, the client resets its timer and goes back to the bound state. Or, if a DHCPACK is not received, and 87.5 percent of the lease time expires, the client goes to the rebinding state.

REBINDING State

The client remains in the rebinding state until one of three events happens. If the client receives a DHCPNACK or the lease expires, it goes back to the initializing state and tries to get another IP address. If the client receives a DHCPACK, it goes to the bound state and resets the timer.

Configuring a DHCP Server:

- You can configure a DHCP server using the configuration file **/etc/dhcpd.conf.**
- DHCP also uses the file /var/lib/dhcp/dhcpd.leases to store the client leasedatabase.
- This file should not be modified by hand. DHCP lease information for each recently assigned IP address is automatically stored in the lease database. The information includes the length of the lease, to whom the IP address has been assigned, the start and end dates for the lease, and the

MAC address of the network interface card that was used to retrieve the lease

- Many RPM packages don't automatically install a /etc/dhcpd.conf file, but you can find a sample copy of dhcpd.conf in the following directory which you can always use as a guide:
- /usr/share/doc/dhcp-<version-number>/dhcpd.conf.sample
- # cp /usr/share/doc/dhcp-3.0pl1/dhcpd.conf.sample \ /etc/dhcpd.conf

/etc/dhcpd.conf

- There are two types of statements in the configuration file:

– Parameters — state how to perform a task, whether to perform a task, or

– what network configuration options to send to the client.

– Declarations — describe the topology of the network, describe the clients, provide addresses for the clients, or apply a group of parameters to a group of declarations.

- Some parameters must start with the option keyword and are referred to as options. **/etc/dhcpd.conf**

- The routers, subnet-mask, domain-name, domain-name- servers, and time-offset options are used for any host statements declared below it
- You must include a subnet declaration for every subnet in your network. If you do not, the

DHCP server will fail to start

- Clients are assigned an IP address within the range
- To assign an IP address to a client based on the MAC address of the network interface card, use the hardware ethernet parameter within a host declaration.

/etc/dhcpd.conf

subnet 192.168.1.0 netmask 255.255.255.0 { # The range of IP addresses the server will issue to #DHCP enabled PC clients booting up on the network range 192.168.1.10 192.168.1.100; range 192.168.1.201

192.168.1.220; # Set the amount of time in seconds that
# a client may keep the IP address default-lease-time 86400;
max-lease- time 86400;
# Set the default gateway to be used by # the PC clients
option routers 192.168.1.1;
# Don't forward DHCP requests from this # NIC interface to any other NIC interfaces optionip-forwarding off;

/etc/dhcpd.conf

# Set the broadcast address and subnet mask # to be used by the DHCP clients option broadcast-address 192.168.1.255; option subnet-mask 255.255.255.0;
# Set the DNS server to be used by the DHCP clients option domain-name-servers 192.168.1.100;
# If you specify a WINS server for your Windows clients,

# you need to include the following option in the dhcpd.conf file: optionnetbios-name- servers 192.168.1.100;

Starting and Stopping the Server:

- Before you start the DHCP server for the first time, it will fail unless there is an existing dhcpd.leases file. To create the file if it does not exist, use the command
- #touch /var/lib/dhcp/dhcpd.leases
- If you have more than one network interface attached to the system, but you only want the DHCP server to start on one of the interface, you can configure the DHCP server to start only on that device. In

/etc/sysconfig/dhcpd, add the name of the interface to the listof DHCPDARGS:

- DHCPDARGS=eth0
- Use the chkconfig command to get DHCP configured to start at boot:
- # chkconfigdhcpd on
- Use the /etc/init.d/dhcpd script to start/stop/restart DHCP after booting

- #

/etc/init.d/dhcpd start #
/etc/init.d/dhcpd stop
# /etc/init.d/dhcpd restart

Configuration Steps:

- 1 . To install DHCP server on ubuntu, Type following command on terminal.

– sudo apt-get install isc-dhcp-server

- 2 . Now we should configure DHCP server. Configuration file is stored at location /etc/dhcp/dhcpd.conf .
- Use gedit to edit dhcpd.conf

```
# A slightly different configuration for an internal
subnet.subnet 172.16.5.0 netmask 255.255.255.0
{   range   172.16.5.2 172.16.5.5;option
domain-name-servers  8.8.8.8;  option
routers  172.16.1.1;  option  broadcast-
address 172.16.5.255;

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

- 3 . Now restart service by using following command.

– sudo service isc-dhcp-server restart

- 4. Now on client computer, in network configuration setting just choose automatic configuration. Thats it client get IP address automatically

Steps for installation of Software on RemoteMachine:

- 1. Type following command for installation of ssh in command prompt– >sudo apt-get install ssh
- 2 . Proceed with installation steps on Remote machine
- 3. After installation, for obtaining remote access, type following command– > sudosshhostname@ipaddress

– For Example. >sudossh student@172.25.28.60

- 4 . Enter the password for host machine then enter the password for remote machine.
- 5. After login for installation of any package such as SBCL package type following command:– > sudo apt-get installpackage_name. – Example: >sudo apt-get installsbcl

Conclusion:

Dynamic Host Configuration Protocol (DHCP) is a client/server protocol that automatically provides an Internet Protocol (IP) host with its IP address and other related configurationinformation such as the subnet mask and default gateway.

Coding Efficiency	**Viva**	**Timely Completion**	**Total**	**Dated Sign of Course In-charge**
5	**3**	**2**	**10**	

Experiment No: 12

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

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
```

1.4.1 Domain Name Space

The domain name space refers 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:

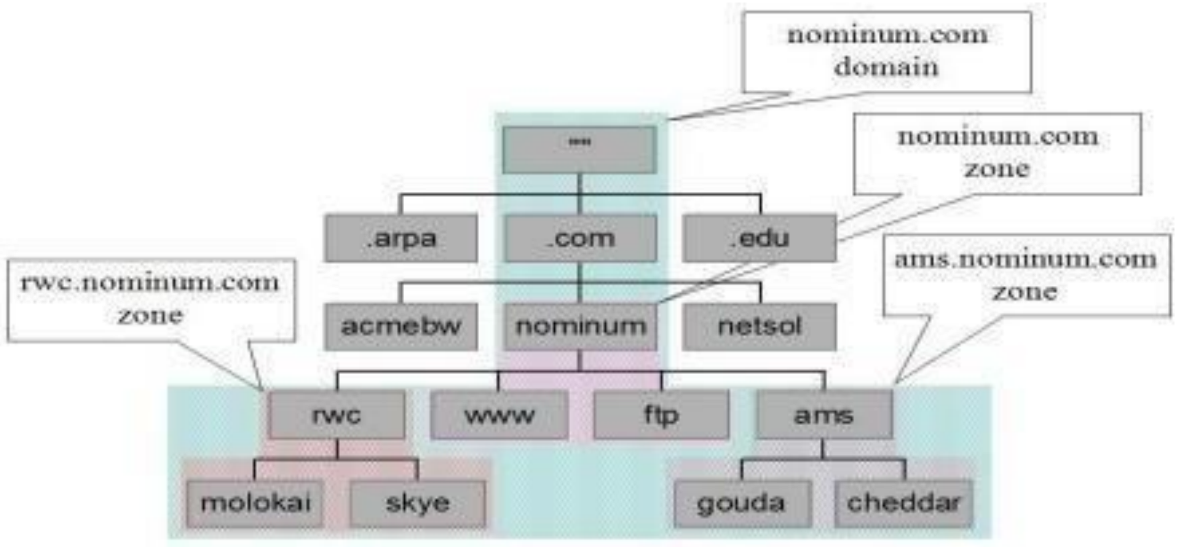
1.4.2 Name Server

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

1.4.3 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.

1.4.4 Types of Name Servers

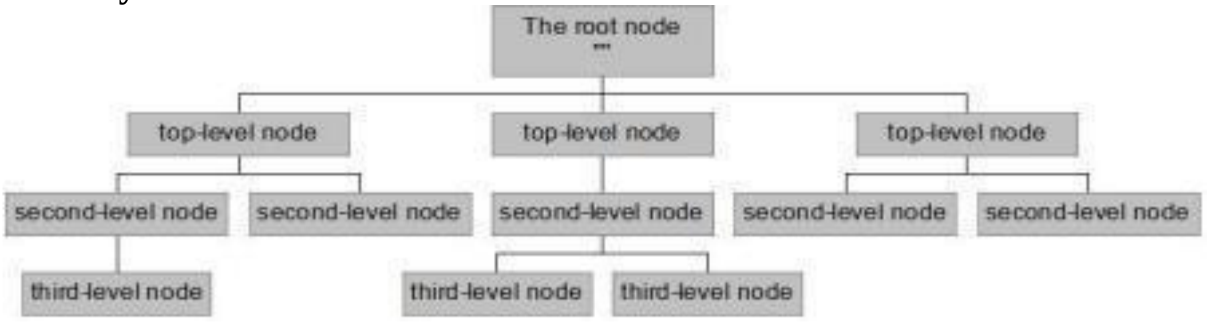
Following are the three categories of Name Servers that manages the entire Domain Name System:

1. Root Server
2. Primary Server
3. Secondary Server

1.4.5.1 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

1.4.5.2 PrimaryServers



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

1.4.5.3 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.

2.1 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.

2.2 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 clientsto 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:

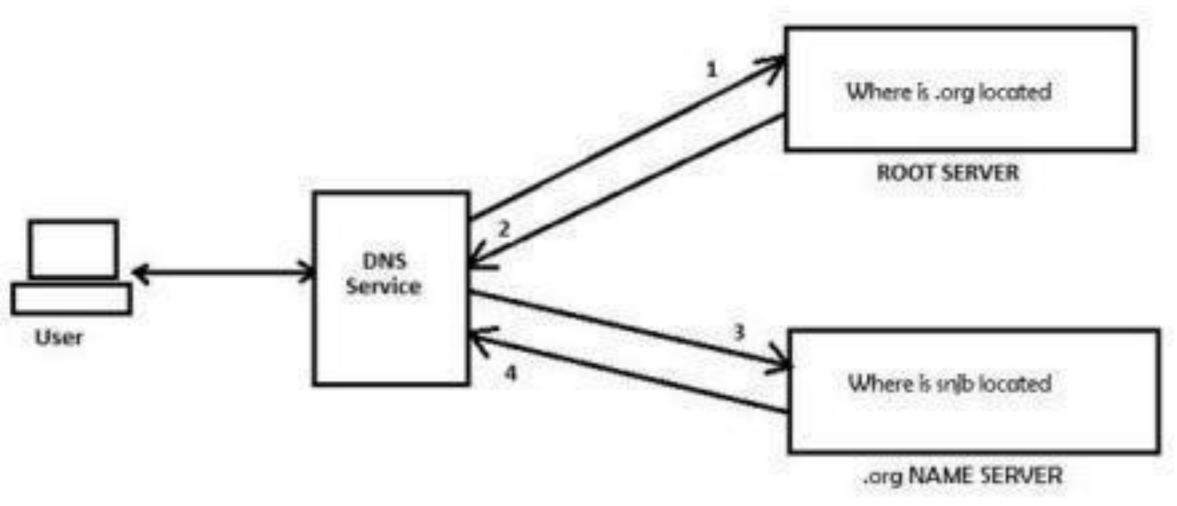
Acontacts Computer **A** contacts **B**
A contacts printer attached to **B**

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.

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

Following is an image that explains how DNS lookup works:



Understanding How DNS Lookup Works

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.

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.

Coding Efficiency	Viva	Timely Completion	Total	Dated Sign of Course In-charge
5	3	2	10	