

Meerut Institute of Engineering & Technology

N.H. 58, Delhi-Roorkee Highway, Baghpat Road Bypass Crossing,
Meerut-250005, UP(India)



Department of Computer Science & Engineering

B.Tech (Session 2018-19)

Even-Semester

COMPUTER NETWORK LAB

(RCS-651)

**L T P
0 0 2**



Department Of Computer Science & Engineering

Course Outcomes

Subject Name: Computer Networks Lab

Subject code: RCS-651

The students are expected to be able to demonstrate the following knowledge, skills and attitudes after completing this course:

1. To understand the basic concepts of network devices and connectivity.
2. To analyze network traffic using wireshark tool.
3. To design and configure a network using Cisco Packet Tracer.
4. To implement a client/server chatting program using socket programming

Table of Content

S.No	Name of the practical	Page No.
1	OSI model simulation.	
2	To study of Network CONNECTING DEVICES.	
3	Verify the connectivity of your workstation to the internet.	
4	Implementation of the IPCONFIG network command.	
5	Program to count Even and Odd Parity.	
6	Program for stuffing & De- stuffing of Bits.	
7	Program to implement Cyclic Redundancy Check CRC.	
8	Implementation of Distance Vector Routing to find suitable path for transmission.	
9	C code to implement RSA Algorithm (Encryption and Decryption).	
10	Write a C program for IPV4, Implementation of decimal to binary, Implementation of binary to decimal.	
11	To implement network using Cisco packet tracer.	
12	To Study packet's information through Wireshark Simulator.	
13	Program to implement Socket Programming.	

Program No. 1

Objective: OSI model simulation.

PROGRAM DEFINITION: This is an open system interconnection program that transmit message from sender to receiver through server different layers.

PROGRAM DESCRIPTION:

The OSI Model deals with connecting open system. This model does not specify the exact services and protocols to use in each layer. Therefore, the OSI Model is not a network architecture. This model has seven layers. They are Physical layer, Data link layer, Presentation layer, Network layer, Session layer, Transport layer and Application layer. At sender side, each layer adds the header. The length of string i.e., number of bytes are not restricted up to Session layer.

ALGORITHM:

1. Read the input string and address.
2. Add application header.
3. Print the string.
4. Add the presentation layer header.
5. Print the string.
6. Add the Session layer header.
7. Print the string.
8. Add the Transport layer header.
9. Print the string.
10. Add the Network layer header.
11. Print the string.
12. Add the Data link layer header.
13. Print the string
14. Add the physical layer header.
15. Print the string.

INPUT: Enter the string: hai

OUTPUT: TRANSMITTER:

APPLICATION LAYER: AH hai

PRESENTATION LAYER: PHAH hai

SESSION LAYER: SHPHAH hai

TRANSPORT LAYER: THSHPHAH hai

NETWORK LAYER: NHTSHPHAH hai

DATALINK LAYER: DHNHTSHPHAH hai

MESSAGE ENTERED INTO PHYSICAL LAYER AND TRANSMITTED.

Practical no. 2

Objective: To study of Network CONNECTING DEVICES.

Passive Hubs

A passive hub is just a connector. It connects the wires coming from different branches. In a star-topology Ethernet LAN, a passive hub is just a point where the signals coming from different stations collide; the hub is the collision point. This type of a hub is part of the media; its location in the Internet model is below the physical layer.

Repeaters

A repeater is a device that operates only in the physical layer. Signals that carry information within a network can travel a fixed distance before attenuation endangers the integrity of the data. A repeater receives a signal and, before it becomes too weak or corrupted, regenerates the original bit pattern. The repeater then sends the refreshed signal.

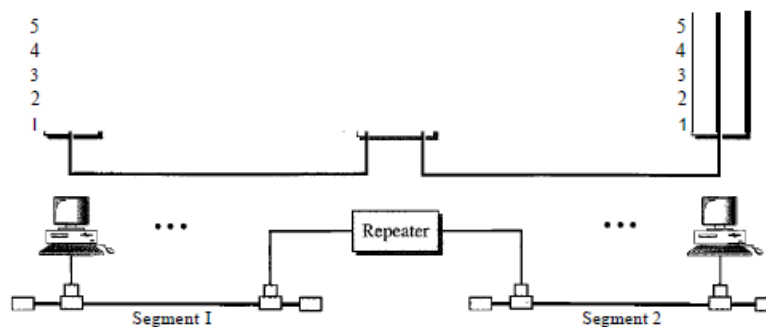


Figure : A repeater connecting two segments of a LAN

A repeater does not actually connect two LANs; it connects two segments of the same LAN. The segments connected are still part of one single LAN. A repeater is not a device that can connect two LANs of different protocols.

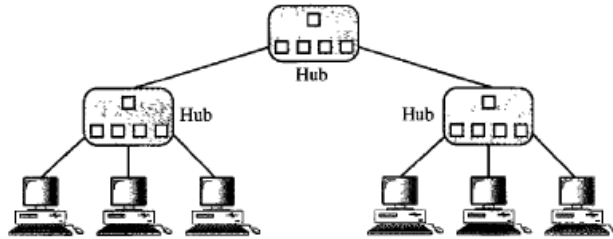
A repeater connects segments of a LAN.

A repeater forwards every frame; it has no filtering capability.

A repeater is a regenerator, not an amplifier.

Active Hubs

An active hub is actually a multipart repeater. It is normally used to create connections between stations in a physical star topology. We have seen examples of hubs in some Ethernet implementations (10Base-T, for example). However, hubs can also be used to create multiple levels of hierarchy, as shown in Figure. The hierarchical use of hubs removes the length limitation of 10Base-T (100 m).



Bridges

A bridge operates in both the physical and the data link layer. As a physical layer device, it regenerates the signal it receives. As a data link layer device, the bridge can check the physical (MAC) addresses (source and destination) contained in the frame.

Transparent Bridges

A transparent bridge is a bridge in which the stations are completely unaware of the bridge's existence. If a bridge is added or deleted from the system, reconfiguration of the stations is unnecessary. According to the IEEE 802.1 d specification, a system equipped with transparent bridges must meet three criteria:

1. Frames must be forwarded from one station to another.
2. The forwarding table is automatically made by learning frame movements in the network.
3. Loops in the system must be prevented.

Two-Layer Switches

When we use the term *switch*, we must be careful because a switch can mean two different things. We must clarify the term by adding the level at which the device operates. We can have a two-layer switch or a three-layer switch. A **three-layer switch** is used at the network layer; it is a kind of router. The **two-layer switch** performs at the physical and data link layers.

A two-layer switch is a bridge, a bridge with many ports and a design that allows better (faster) performance. A bridge with a few ports can connect a few LANs together. A bridge with many ports may be able to allocate a unique port to each station, with each station on its own independent entity. This means no competing traffic (no collision, as we saw in Ethernet).

A two-layer switch, as a bridge does, makes a filtering decision based on the MAC address of the frame it received. However, a two-layer switch can be more sophisticated. It can have a buffer to hold the frames for processing. It can have a switching factor that forwards the frames faster. Some new two-layer switches, called *cut-through* switches, have been designed to forward the frame as soon as they check the MAC addresses in the header of the frame.

Routers

A router is a three-layer device that routes packets based on their logical addresses (host-to-host addressing). A router normally connects LANs and WANs in the Internet and has a routing table that is used for making decisions about the route. The routing tables are normally dynamic and are updated using routing protocols.

Three-Layer Switches

A three-layer switch is a router, but a faster and more sophisticated. The switching fabric in a three-layer switch allows faster table lookup and forwarding. In this book, we use the terms *router* and *three-layer switch* interchangeably.

Gateway

A gateway is normally a computer that operates in all five layers of the Internet or seven layers of OSI model. A gateway takes an application message, reads it, and interprets it. This means that it can be used as a connecting device between two internetworks that use different models. For example, a network designed to use the OSI model can be connected to another network using the Internet model. The gateway connecting the two systems can take a frame as it arrives from the first system, move it up to the OSI application layer, and remove the message.

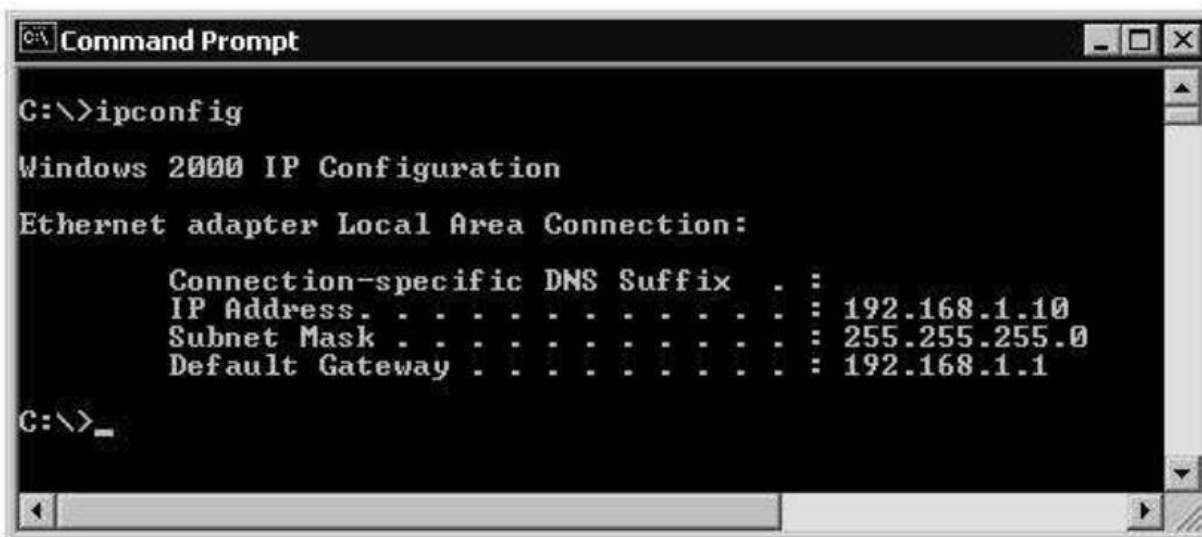
Practical no. 3

Objective: Verify the connectivity of your workstation to the internet.

Experiment

1. Verify the connectivity of your workstation to the internet.
2. Open the Command Prompt of the operating system using either of the following methods:
Click on **Start > All Programs > Accessories > Command Prompt** or
Click on **Start > Run**, enter **cmd** (short for command) and click on **ok**.
A Command Prompt screen should open.
3. Gather TCP/IP configuration information: Type **ipconfig** (short for IP configuration) and press **Enter**. The screen will show the IP address, subnet mask, and default gateway for your computer's connection.

Notice the values in the Command Prompt. The IP address and the default gateway should be in the same network or subnet, otherwise this host would not be able to communicate outside the network. In Fig. 3, the subnet mask tells us that the first three octets of the IP address and the default gateway must be the same in order to be in the same network.



```
C:\>ipconfig

Windows 2000 IP Configuration

Ethernet adapter Local Area Connection:

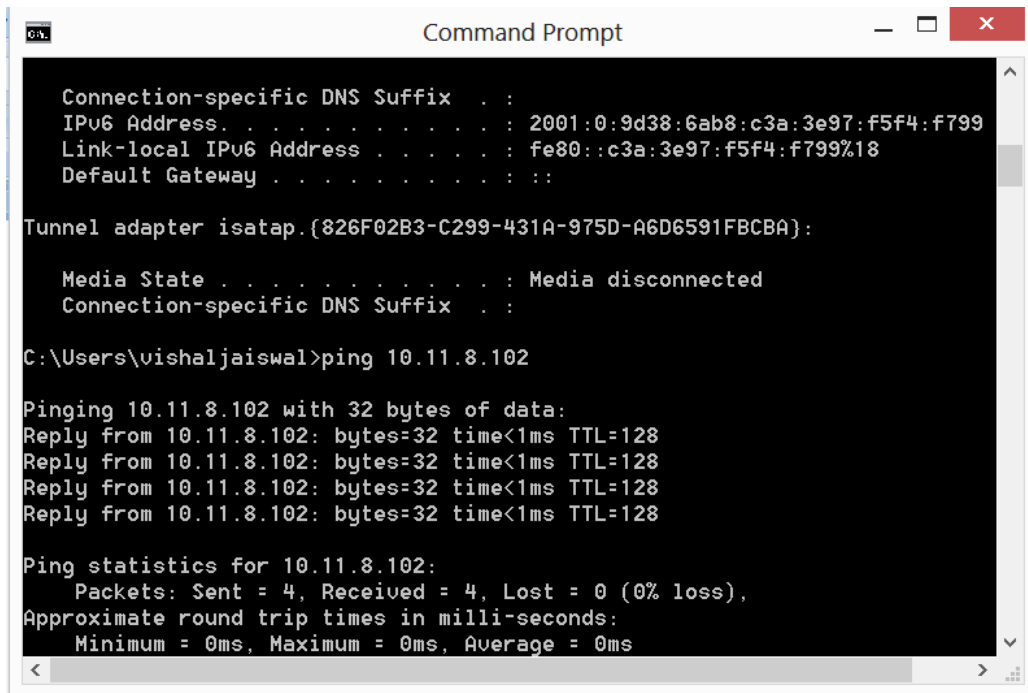
    Connection-specific DNS Suffix  . : .
    IP Address. . . . .               : 192.168.1.10
    Subnet Mask . . . . .             : 255.255.255.0
    Default Gateway . . . . .         : 192.168.1.1

C:\>_
```

Figure 3. The TCP/IP configuration information of a workstation

4. Check more detailed TCP/IP configuration information: Type **ipconfig /all** and press **Enter**. What are the DNS and DHCP server addresses? What are their functions? What is the MAC of the network interface card?
5. Ping the IP address of another computer. Note that for the ping and tracert commands to work the PC firewalls have to be disabled. Why do you think this is so? Ask the IP address of the workstation that is being used by another group of students. Then type **ping**, space, and the IP address that you received, then press

Enter. Notice the outputs. Fig. 4 shows a successful result of a ping to a given IP address.



```
Command Prompt

Connection-specific DNS Suffix . : 
IPv6 Address. . . . . : 2001:0:9d38:6ab8:c3a:3e97:f5f4:f799
Link-local IPv6 Address . . . . . : fe80::c3a:3e97:f5f4:f799%18
Default Gateway . . . . . : ::

Tunnel adapter isatap.{826F02B3-C299-431A-975D-A6D6591FBCBA}:

Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . : 

C:\Users\vishaljaiswal>ping 10.11.8.102

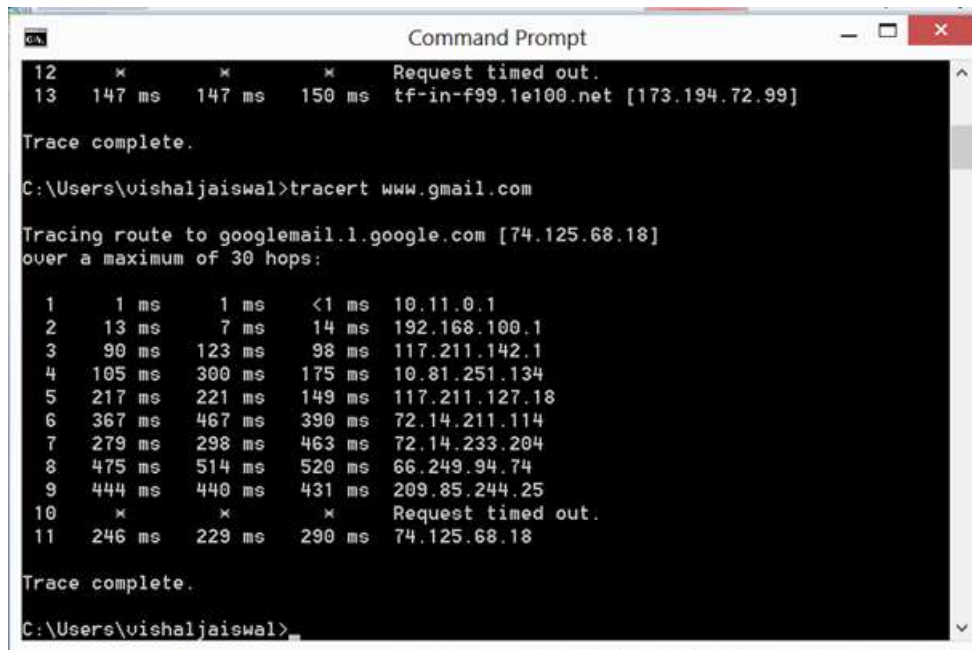
Pinging 10.11.8.102 with 32 bytes of data:
Reply from 10.11.8.102: bytes=32 time<1ms TTL=128
Reply from 10.11.8.102: bytes=32 time<1ms TTL=128
Reply from 10.11.8.102: bytes=32 time<1ms TTL=128
Reply from 10.11.8.102: bytes=32 time<1ms TTL=128

Ping statistics for 10.11.8.102:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

Figure 4. A successful result of a ping to a certain IP address

6. Ping the IP address of the gateway router from the details that have been observed in the output of step 4 above. If the ping is successful, it means that there is a physical connectivity to the router on the local network and probably the rest of the world.
7. Ping the Loopback IP address of your computer. Type the following command: ping **127.0.0.1**. The IP address 127.0.0.1 is reserved for loopback testing. If the ping is successful, then TCP/IP is properly installed and functioning on this computer.
8. You can also ping using names like websites. Ping the IP address of the cisco website. Type **ping**, space and **www.cisco.com**, then press **Enter**. Notice the outputs. A DNS server will resolve the name to an IP address and the ping will be successful only in the existence of the DNS server.
9. Ping www.ee.uct.ac.za and observe the results. Is there a difference in time between the results shown by pinging www.cisco.com and www.ee.uct.ac.za. If so why and if not why?

10. Trace the route to the Cisco website. Type **tracert www.cisco.com** and press **enter**. In a successful output, you will see listings of all routers the tracert requests had to pass through to get to the destination.
11. Trace the route to the website of the Department of Electrical Engineering. Type **tracert www.ee.uct.ac.za** and press **enter**. The output should take less time than that of step 9.



```
12      x      x      x      Request timed out.
13  147 ms  147 ms  150 ms  tf-in-f99.1e100.net [173.194.72.99]

Trace complete.

C:\Users\vishaljaiswal>tracert www.gmail.com

Tracing route to googlemail.1.google.com [74.125.68.18]
over a maximum of 30 hops:

  0  0 ms  0 ms  0 ms  10.11.0.1
  1  1 ms  1 ms  <1 ms  10.11.0.1
  2  13 ms  7 ms  14 ms  192.168.100.1
  3  90 ms  123 ms  98 ms  117.211.142.1
  4  105 ms  300 ms  175 ms  10.81.251.134
  5  217 ms  221 ms  149 ms  117.211.127.18
  6  367 ms  467 ms  390 ms  72.14.211.114
  7  279 ms  298 ms  463 ms  72.14.233.204
  8  475 ms  514 ms  520 ms  66.249.94.74
  9  444 ms  440 ms  431 ms  209.85.244.25
 10      x      x      x      Request timed out.
 11  246 ms  229 ms  290 ms  74.125.68.18

Trace complete.

C:\Users\vishaljaiswal>
```

Figure 5. A traceroute output

Program No.4

Ojective: **Implementation of the IPCONFIG network command**
Configure IP (*internet protocol* configuration)

Syntax:

IPCONFIG /all:

Display full configuration information.

IPCONFIG /release [adapter]:

Release the IP address for the specified adapter.

IPCONFIG /renew [adapter]:

Renew the IP address for the specified adapter.

IPCONFIG /flushdns:

Purge the DNS Resolver cache.

IPCONFIG /registerdns:

Refresh all DHCP leases and re-register DNS names.

IPCONFIG /displaydns:

Display the contents of the DNS Resolver Cache.

IPCONFIG /showclassid adapter:

Display all the DHCP class IDs allowed for adapter.

IPCONFIG /setclassid adapter [classid]:

Modify the dhcp class id.

If the Adapter name contains spaces, use quotes: "Adapter Name" wildcard characters * and ? allowed, see the examples below The default is to display only the IP address, subnet mask and default gateway for each adapter bound to TCP/IP.

For Release and Renew, if no adapter name is specified, then the IP address leases for all

For Setclassid, if no ClassId is specified, then the ClassId is removed.

Examples:

ipconfig ... Show information.

ipconfig /all ... Show detailed information

ipconfig /renew ... renew all adapters

ipconfig /renew EL* ... renew any connection that has its name starting with EL

ipconfig /release *Con* ... release all matching connections, e.g. "Local Area Connection 1" or "Local Area Connection 2"

ipconfig /setclassid "Local Area Connection" TEST ... set the DHCP class ID for the named adapter to = TEST

```
Command Prompt

For Setclassid and Setclassid6, if no ClassId is specified, then t
removed.

Examples:
> ipconfig ... Show information
> ipconfig /all ... Show detailed information
> ipconfig /renew ... renew all adapters
> ipconfig /renew EL* ... renew any connection that
name starting with EL
> ipconfig /release *Con* ... release all matching conn
eg. "Wired Ethernet Conne
"Wired Ethernet Conne
> ipconfig /allcompartments ... Show information about al
compartments
> ipconfig /allcompartments /all ... Show detailed information
compartments
```

Program No. 5

Objective: Program to count Even and Odd Parity.

Parity: Parity of a number refers to whether it contains an odd or even number of 1-bits. The number has “odd parity”, if it contains odd number of 1-bits and is “even parity” if it contains even number of 1-bits.

Main idea of the below solution is – Loop while n is not 0 and in loop unset one of the set bits and invert parity.

Algorithm: getParity(n)

1. Initialize parity = 0
2. Loop while n != 0
 - a. Invert parity
parity = !parity
 - b. Unset rightmost set bit
n = n & (n-1)
3. return parity

Example:

Initialize: n = 13 (1101) parity = 0

n = 13 & 12 = 12 (1100) parity = 1

n = 12 & 11 = 8 (1000) parity = 0

n = 8 & 7 = 0 (0000) parity = 1

Program No. 6

Objective: Program for stuffing & De- stuffing of Bits.

- (1) Write a program to implement bit stuffing & De-stuffing.
- (2) Write a program to implement character stuffing & De-stuffing.

(1) Write a program to implement bit stuffing & De-stuffing.

Resources: Turbo C, C++.

Bit Stuffing and Destuffing

- ☒ ☒ Include <iostream.h>,<conio.h>,<io.h> files both in transmitter & receiver programs.
- ☒ ☒ During the transmission, attach a flag pattern (01111110) at the start & end of data unit.
- ☒ ☒ If transmitter sees five consecutive one's in data, it stuffs zero bit in data.
- ☒ ☒ At the receiving end, whenever in data it finds five consecutive one's and the next bit are zero then the receiver will de stuff that zero bit. e.g. If the Pattern to be transmitted is 00011110111110000, then at the transmitter side will be 000111101111100000 because as 5 consecutive 1's are detected, one 0 should be stuffed and at the receiver side again as it will detect 0 after 5 consecutive 1's , it will de-stuff it.

(2) Write a program to implement character stuffing & De-stuffing.

Resources: Turbo C, C++.

Character Stuffing and Destuffing

- ☒ ☒ Include <iostream.h>,<conio.h>,<io.h> files both in transmitter & receiver programs.
- ☒ ☒ This is type of Framing Method.
- ☒ ☒ During the transmission attach a ASCII Code pattern DLE STX at the start & DLE ETX end of data Unit.
- ☒ ☒ If transmitter sees DLE stuff another DLE text in data.
- ☒ ☒ At the receiving end, whenever the data it finds five consecutive DLE then receiver will destuff One DLE.

Program No. 7

Objective: **Program to implement Cyclic Redundancy Check CRC.**

CRC or Cyclic Redundancy Check is a method of detecting accidental changes/errors in communication channel.

CRC uses **Generator Polynomial** which is available on both sender and receiver side. An example generator polynomial is of the form like $x^3 + x + 1$. This generator polynomial represents key 1011. Another example is $x^2 + 1$ that represents key 101.

n : Number of bits in data to be sent

from sender side.

k : Number of bits in the key obtained

from generator polynomial.

Sender Side (Generation of Encoded Data from Data and Generator Polynomial (or Key)):

1. The binary data is first augmented by adding k-1 zeros in the end of the data
2. Use **modulo-2 binary division** to divide binary data by the key and store remainder of division.
3. Append the remainder at the end of the data to form the encoded data and send the same

Receiver Side (Check if there are errors introduced in transmission)

Perform modulo-2 division again and if remainder is 0, then there are no errors.

In this article we will focus only on finding the remainder i.e. check word and the code word.

Modulo 2 Division:

The process of modulo-2 binary division is the same as the familiar division process we use for decimal numbers. Just that instead of subtraction, we use XOR here.

- In each step, a copy of the divisor (or data) is XORed with the k bits of the dividend (or key).
- The result of the XOR operation (remainder) is (n-1) bits, which is used for the next step after 1 extra bit is pulled down to make it n bits long.
- When there are no bits left to pull down, we have a result. The (n-1)-bit remainder which is appended at the sender side.

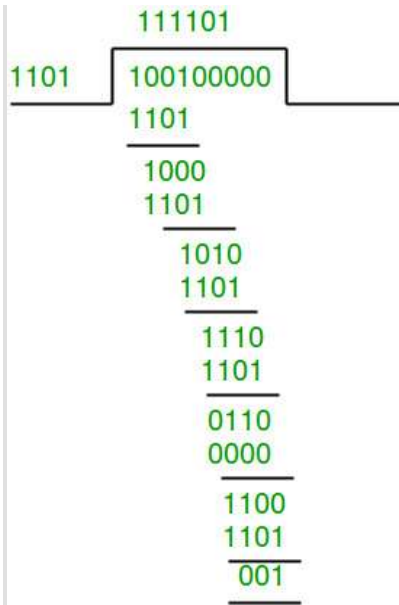
Illustration:

Example 1 (No error in transmission):

Data word to be sent - 100100

Key - 1101 [Or generator polynomial $x^3 + x^2 + 1$]

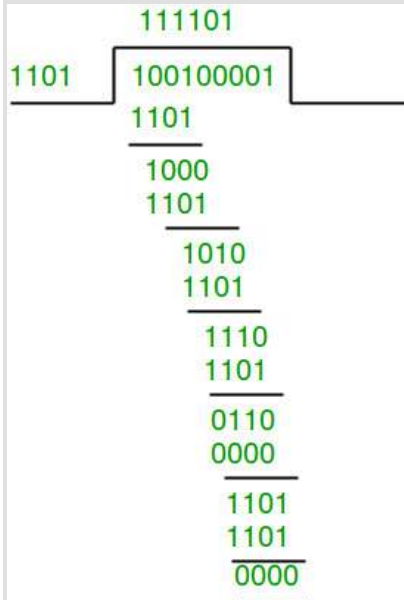
Sender Side:



Therefore, the remainder is 001 and hence the encoded data sent is 100100001.

Receiver Side:

Code word received at the receiver side 100100001



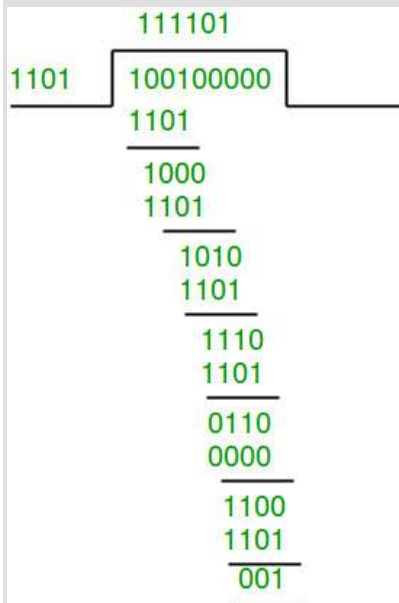
Therefore, the remainder is all zeros. Hence, the data received has no error.

Example 2: (Error in transmission)

Data word to be sent - 100100

Key - 1101

Sender Side:

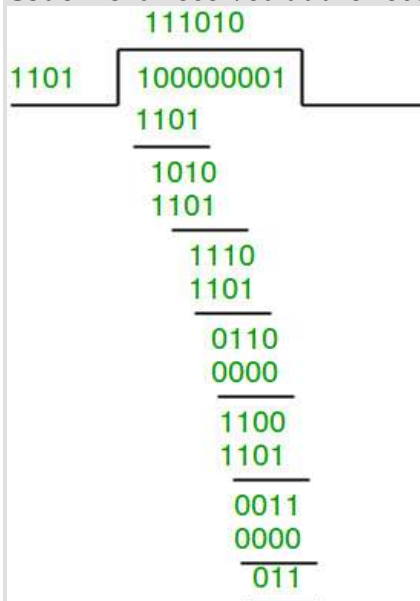


Therefore, the remainder is 001 and hence the code word sent is 100100001.

Receiver Side

Let there be error in transmission media

Code word received at the receiver side - 100000001



Since the remainder is not all zeroes, the error is detected at the receiver side.

Program No. 8

Objective: Implementation of Distance Vector Routing to find suitable path for transmission.

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

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

Information kept by DV router -

- Each router has an ID
- Associated with each link connected to a router,
- there is a link cost (static or dynamic).
- Intermediate hops

Distance Vector Table Initialization -

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

Distance Vector Algorithm –

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

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

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

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

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

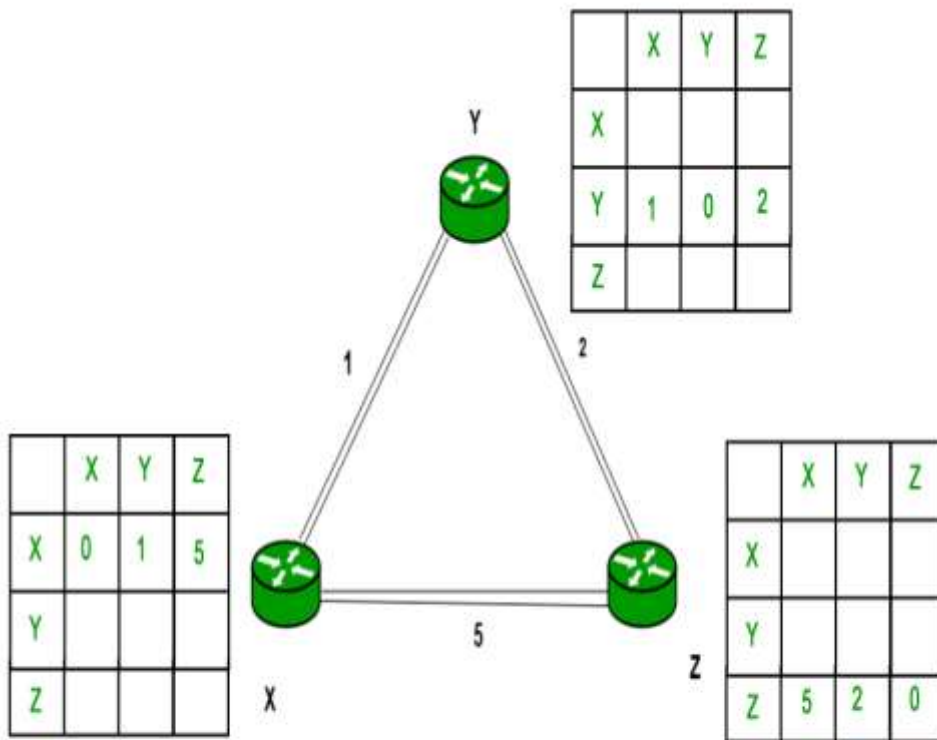
Node x also maintains its neighbors' distance vectors

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

Note –

- From time-to-time, each node sends its own distance vector estimate to neighbors.
- When a node x receives new DV estimate from any neighbor v, it saves v's distance vector and it updates its own DV using B-F equation:
- $D_x(y) = \min \{ C(x,v) + D_v(y) \}$ for each node $y \in N$

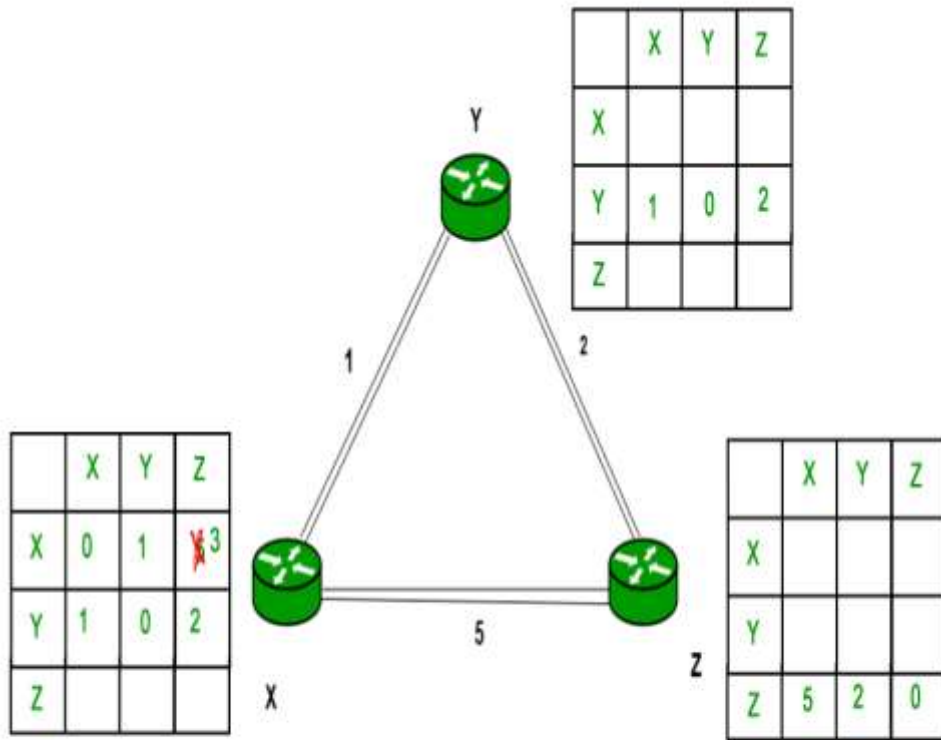
Example – Consider 3-routers X, Y and Z as shown in figure. Each router have their routing table. Every routing table will contain distance to the destination nodes.



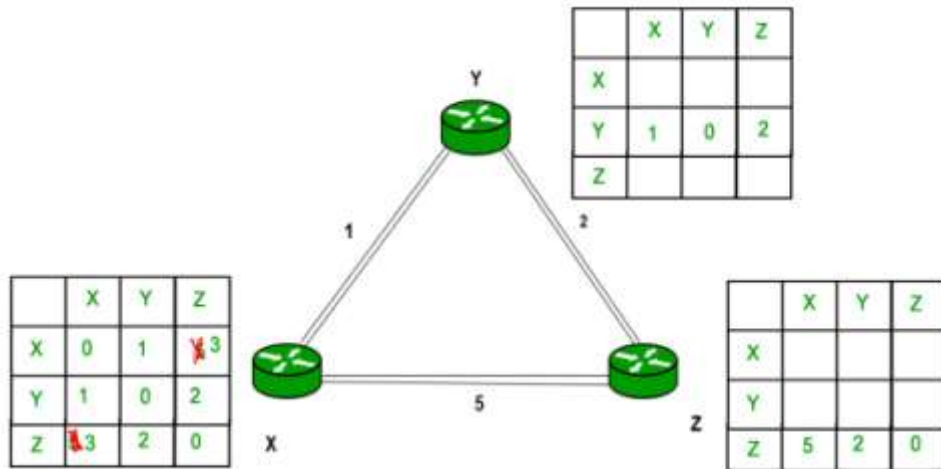
Consider router X, X will share its routing table to neighbors and neighbors will share their routing table to it to X and distance from node X to destination will be calculated using the Bellman-Ford equation.

$$D_x(y) = \min \{ C(x,v) + D_v(y) \} \text{ for each node } y \in N$$

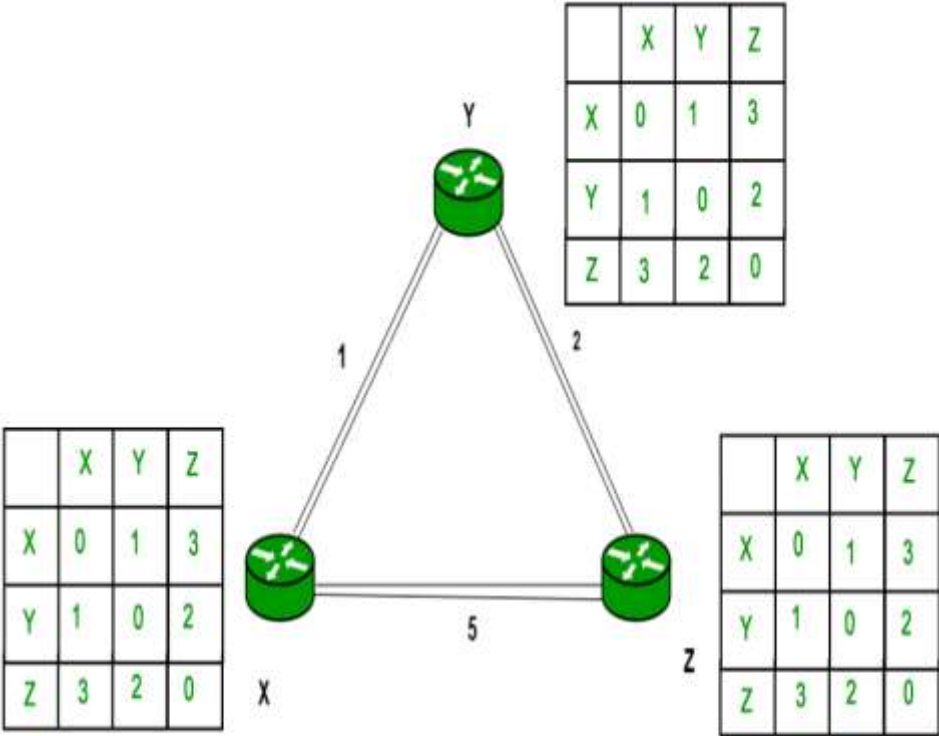
As we can see that the distance will be less going from X to Z when Y is an intermediate node (hop) so it will be updated in routing table X.



Similarly for Z also -



Finally the routing table for all -



Program No. 9

Objective: C code to implement RSA Algorithm(Encryption and Decryption)

RSA Algorithm in Cryptography

RSA algorithm is asymmetric cryptography algorithm. Asymmetric actually means that it works on two different keys i.e. **Public Key** and **Private Key**. As the name describes that the Public Key is given to everyone and Private key is kept private.

An example of asymmetric cryptography :

1. A client (for example browser) sends its public key to the server and requests for some data.
2. The server encrypts the data using client's public key and sends the encrypted data.
3. Client receives this data and decrypts it.

Since this is asymmetric, nobody else except browser can decrypt the data even if a third party has public key of browser.

The idea! The idea of RSA is based on the fact that it is difficult to factorize a large integer. The public key consists of two numbers where one number is multiplication of two large prime numbers. And private key is also derived from the same two prime numbers. So if somebody can factorize the large number, the private key is compromised. Therefore encryption strength totally lies on the key size and if we double or triple the key size, the strength of encryption increases exponentially. RSA keys can be typically 1024 or 2048 bits long, but experts believe that 1024 bit keys could be broken in the near future. But till now it seems to be an infeasible task.

Key Generation

Select p, q	p and q both prime, $p \neq q$
Calculate $n = p \times q$	
Calculate $\phi(n) = (p - 1)(q - 1)$	
Select integer e	$\text{gcd}(\phi(n), e) = 1; 1 < e < \phi(n)$
Calculate d	$d \equiv e^{-1} \pmod{\phi(n)}$
Public key	$PU = \{e, n\}$
Private key	$PR = \{d, n\}$

Encryption

Plaintext: $M < n$

Ciphertext: $C = M^e \text{ mod } n$

Decryption

Ciphertext: C

Plaintext: $M = C^d \text{ mod } n$

RSA Example - Key Setup

1. Select primes: $p = 17$; $q = 11$
2. Calculate $n = pq = 17 \times 11 = 187$
3. Calculate $\phi(n) = (p-1)(q-1) = 16 \times 10 = 160$
4. Select e : $\text{GCD}(e, 160) = 1$; choose $e = 7$
5. Derive d : $de = 1 \text{ mod } 160$ and $d < 160$
Get $d = 23$ since $23 \times 7 = 161 = 10 \times 160 + 1$
6. Publish public key: $\text{PU} = \{7, 187\}$
7. Keep private key secret: $\text{PR} = \{23, 187\}$

RSA Example - En/Decryption

- sample RSA encryption/decryption is:
- given message $M = 88$ (nb. $88 < 187$)
- encryption:
$$C = 88^7 \bmod 187 = 11$$
- decryption:
$$M = 11^{23} \bmod 187 = 88$$

Program No.10

Objective:

Write a C program for IPV4, Implementation of decimal to binary, Implementation of binary to decimal.

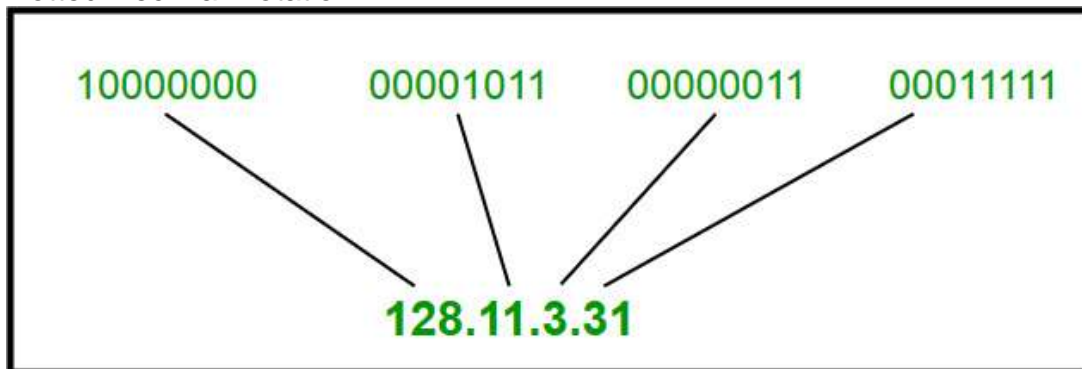
Resources: Turbo C, C++.

IP Addressing | Introduction and Classful Addressing

IP address is an address having information about how to reach a specific host, especially outside the LAN. An IP address is a 32 bit unique address having an address space of 2^{32} .

Generally, there are two notations in which IP address is written, dotted decimal notation and hexadecimal notation.

Dotted Decimal Notation



Hexadecimal Notation



Some points to be noted about dotted decimal notation :

1. The value of any segment (byte) is between 0 and 255 (both included).
2. There are no zeroes preceding the value in any segment (054 is wrong, 54 is correct).

Classful Addressing

The 32 bit IP address is divided into five sub-classes. These are:

- Class A
- Class B

- Class C
- Class D
- Class E

Each of these classes has a valid range of IP addresses. Classes D and E are reserved for multicast and experimental purposes respectively. The order of bits in the first octet determine the classes of IP address.

IPv4 address is divided into two parts:

- **Network ID**
- **Host ID**

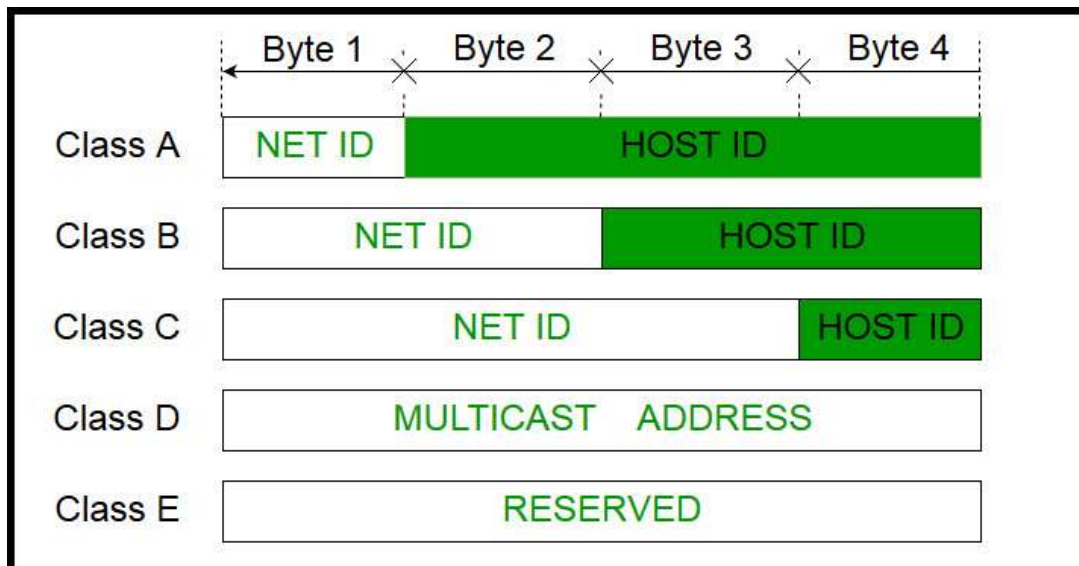
The class of IP address is used to determine the bits used for network ID and host ID and the number of total networks and hosts possible in that particular class. Each ISP or network administrator assigns IP address to each device that is connected to its network.

	First byte	Second byte	Third byte	Fourth byte
Class A	0			
Class B	10			
Class C	110			
Class D	1110			
Class E	1111			

a. Binary notation

	First byte	Second byte	Third byte	Fourth byte
Class A	0-127			
Class B	128-191			
Class C	192-223			
Class D	224-239			
Class E	240-255			

b. Dotted-decimal notation



Program No.11

OBJECTIVE: To implement a network using Cisco Packet Tracer.

THEORY: Packet Tracer is a protocol simulator developed by Dennis Frezzo and his team at Cisco Systems. Packet Tracer (PT) is a powerful and dynamic tool that displays the various protocols used in networking, in either Real Time or Simulation mode. This includes layer 2 protocols such as Ethernet and PPP, layer 3 protocols such as IP, ICMP, and ARP, and layer 4 protocols such as TCP and UDP. Routing protocols can also be traced.

Steps to simulate a network:

Step 1: Start Packet Tracer You will see the start screen as shown below.

Step 2: Choose "Hub" and then select "Generic"

Step 3: After selecting "Generic" click on the main area. You will see a Hub.

Step 4: Select "End Devices" and then click at "Generic" Choosing Devices and Connections We will begin building our network topology by selecting devices and the media in which to connect them. Several types of devices and network connections can be used.

Step 6: Select "Connections" from Power Cycle Devices and click on "Automatically choose Connection Type"

Step 7: Draw connections from Hub to PCs

Step 8: Double click on a PC, a box will appear. Click on the "Desktop" tab.

Step 9: Then select "IP configuration"

Step 10: Write the IP address of your network and click at the Subnet mask field. Subnet Mask will appear automatically.

Step11: Repeat Step 10 to set the IPs for all the PCs.

Step 12: Select "Add simple message"

Step 13: Drag and Drop the message to the source device and then to the Destination device In this case my source device is PC1 and destination device is PC4.

Step 14: Select the Simulation Mode at the bottom right corner.

Step15: Click at “Auto Capture / Play” Conclusion: Connection established successfully between Source and Destination.

Step 16: Observe the path of the Message from source to Hub, then to all devices. And then from Destination to Hub then back to the source.

Step 17: Finally observe the marks. If the source PC is marked correct it means you have successfully established the connection.

Screenshots:

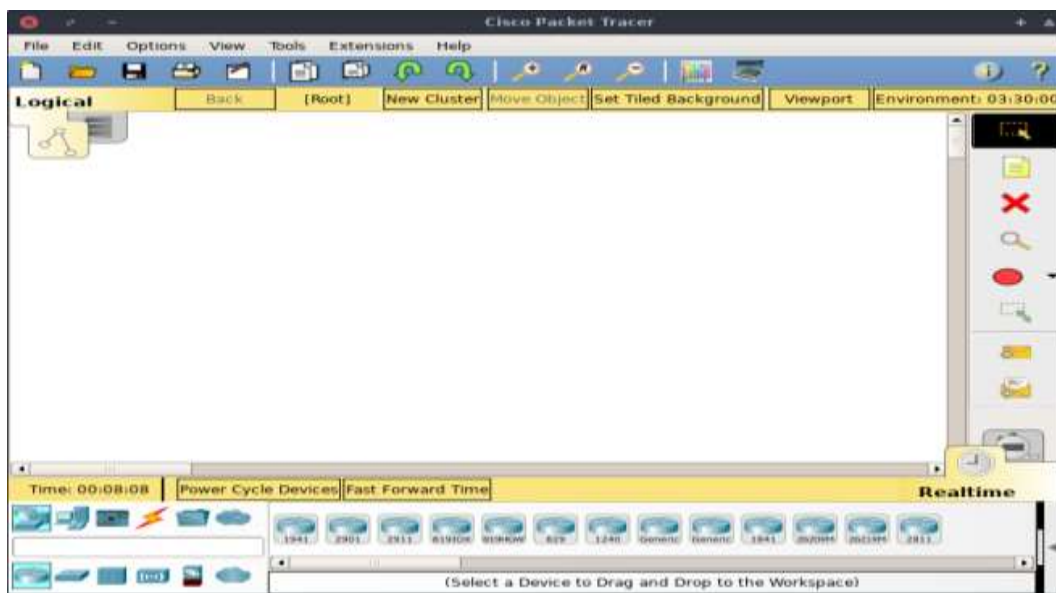


Fig1: Step 1

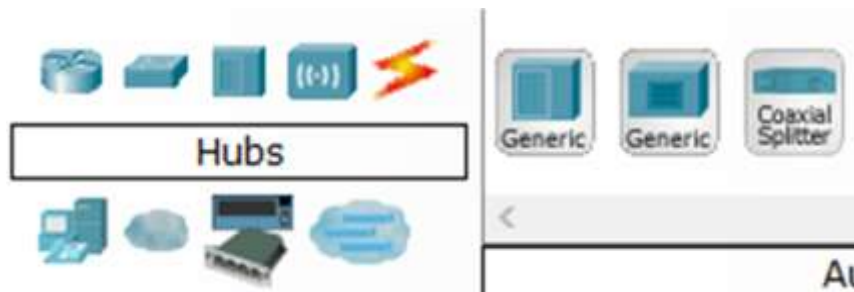


Fig2: Step 2



Fig3:Step3



Fig4:Step4

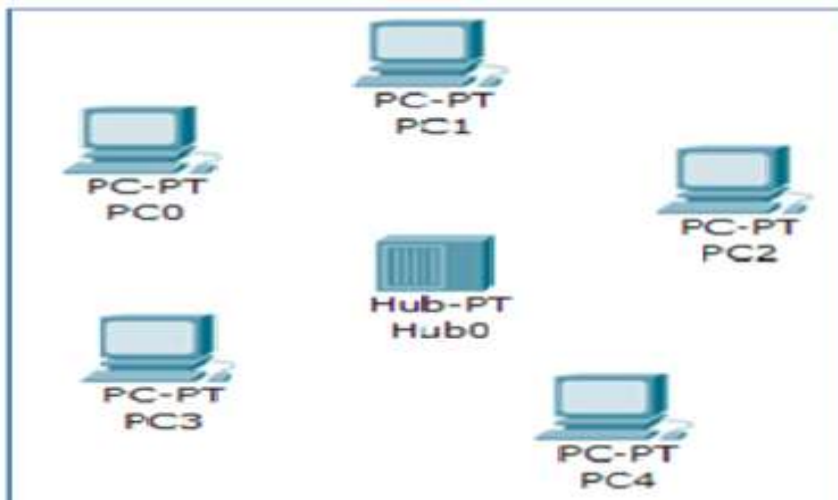


Fig5:Step5



Fig6:Step6

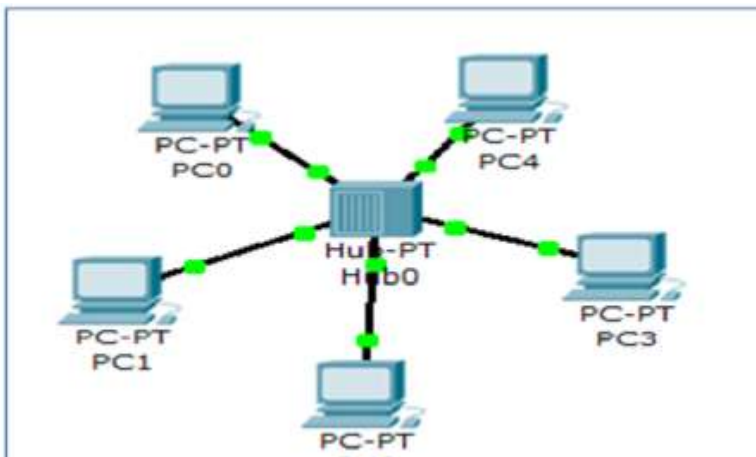


Fig7:step7



Fig8:Step8

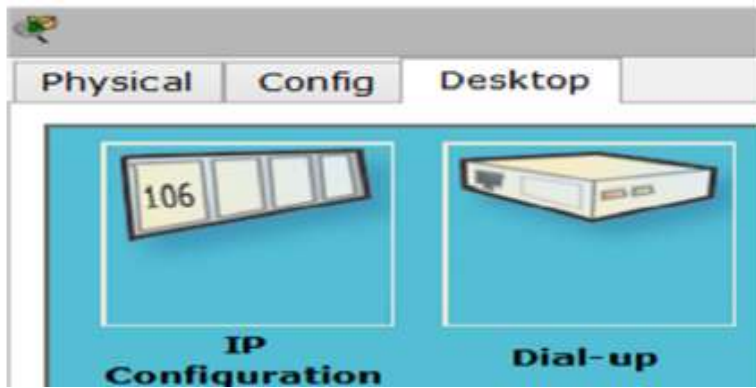


Fig9:Step9

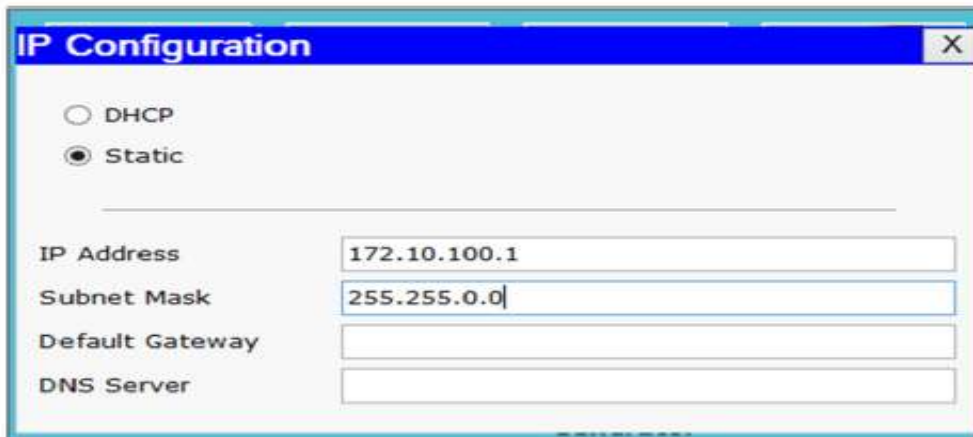


Fig10:Step10

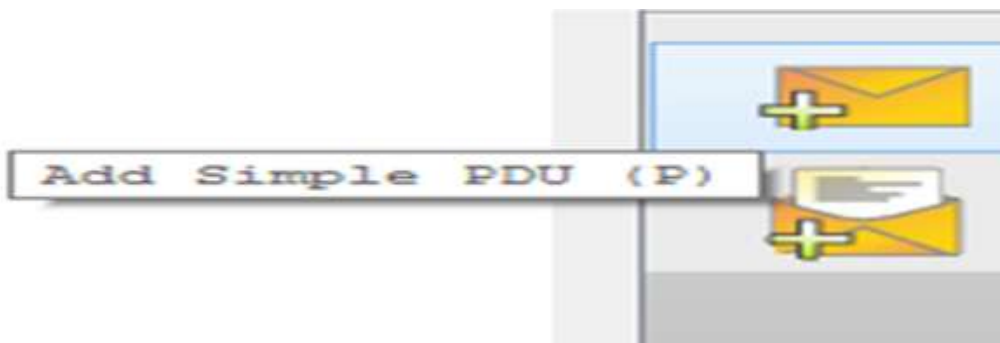


Fig12:Step12

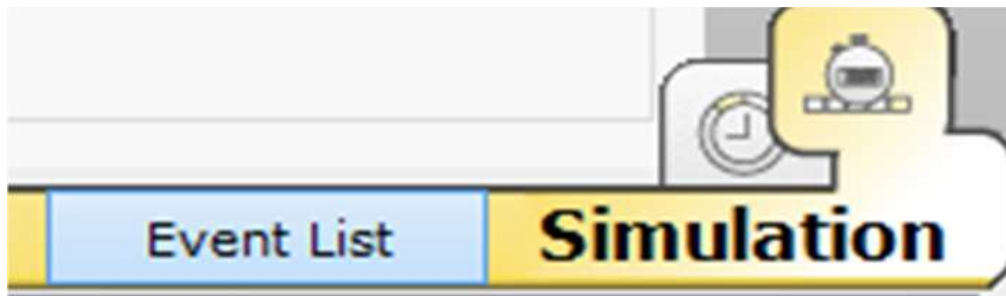


Fig14:Step14

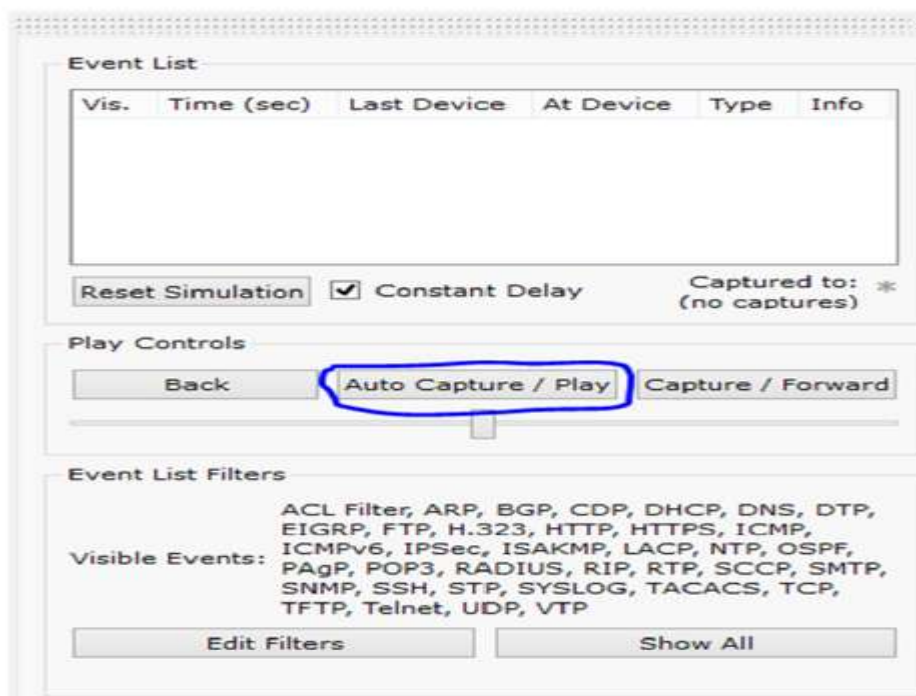


Fig15:Step15

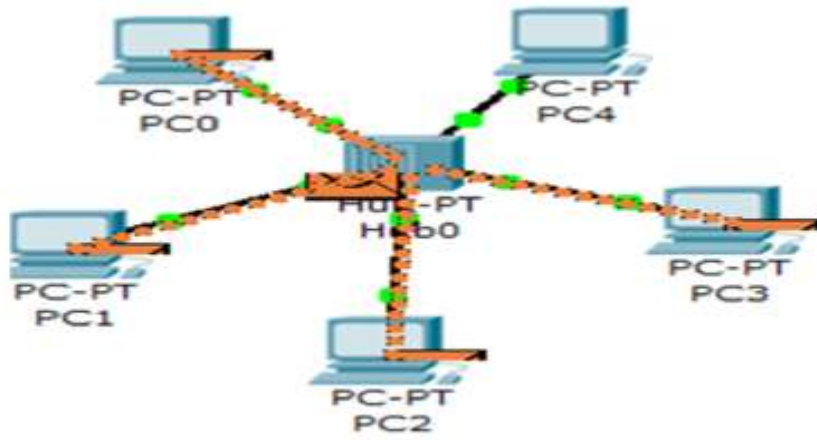


Fig16:Step16

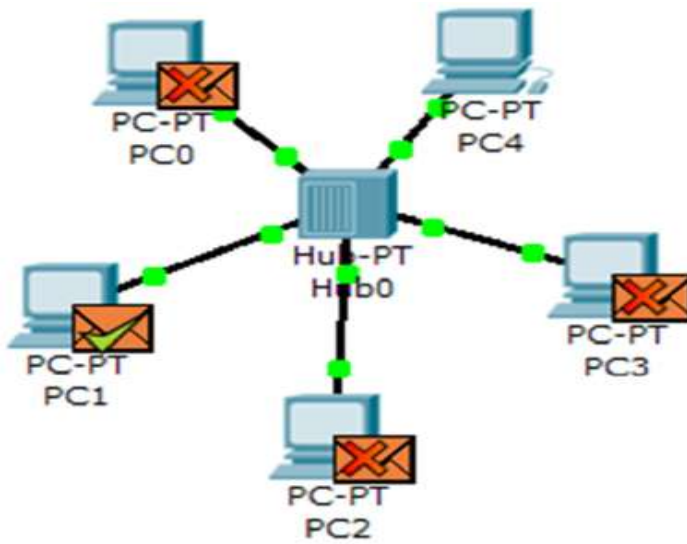


Fig17:Step17

Program No.3

OBJECTIVE: To Study packet's information through Wireshark Simulator.

THEORY

Wireshark is a tool that allows packet traces to be sniffed, captured and analysed. Before Wireshark (or in general, any packet capture tool) is used, careful consideration should be given to where in the network packets are to be captured. Refer to the [capture setup pages](#) in the wireshark.org wiki for technical details on various deployment scenarios. If it is unclear which deployment scenario should be used to capture traces for a particular problem, consider opening a service request with Novell Technical Services for assistance.

Obtain appropriate Wireshark package

Obtain a Wireshark package or installer for the operating system running on the system which is to be used for packet capture.

Wireshark is included in Novell's SUSE Linux products (for some products, under its old name, Ethereal). For other platforms, download a binary or installer from <http://www.wireshark.org>. With installers, ensure all product components are selected for installation.

Start Wireshark

Start Wireshark. On a Linux or Unix environment, select the Wireshark or Ethereal entry in the desktop environment's menu, or run "wireshark" (or "ethereal") from a root shell in a terminal emulator. In a Microsoft Windows environment, launch wireshark.exe from C:\Program Files\Wireshark.

Note that on Un*x systems, a non-GUI version of Wireshark called "tshark" (or "tethereal") may be available as well, but its use is beyond the scope of this document.

Configure Wireshark

After starting Wireshark, do the following:

1. Select **Capture | Interfaces**
2. Select the interface on which packets need to be captured.
3. If capture options need to be configured, click the **Options** button for the chosen interface. Note the following recommendations for traces that are to be analysed by Novell Technical Services:
 - **Capture packet in promiscuous mode:** This option allows the adapter to capture all traffic not just traffic destined for this workstation. It should be enabled.
 - **Limit each packet to:** Leave this option unset. Novell Support will always want to see full frames.

- **Filters:** Generally, Novell Support prefers an unfiltered trace. For documentation on filters, please refer to [TID 10084702 - How to configure a capture filter for Ethereal](#) (formerly NOVL90720).
 - **Capture file(s):** This allows a file to be specified to be used for the packet capture. By default Wireshark will use temporary files and memory to capture traffic. Specify a file for reliability.
 - **Use multiple files, Ring buffer with:** These options should be used when Wireshark needs to be left running capturing data data for a long period of time. The number of files is configurable. When a file fills up, it will wrap to the next file. The file name should be specified if the ring buffer is to be used.
 - **Stop capture after xxx packet(s) captured:** Novell Technical Support would most likely never use this option. Leave disabled.
 - **Stop capture after xxx kilobyte(s) captured:** Novell Technical Support would most likely never use this option. Leave disabled.
 - **Stop capture after xxx second(s):** Novell Technical Support would most likely never use this option. Leave disabled.
 - **Update list of packets in real time:** Disable this option if the problem that's being investigated is occurring on the same workstation as where Wireshark is running.
 - **Automatic scrolling in live capture:** Wireshark will scroll the window so that the most current packet is displayed.
 - **Hide capture info dialog:** Disable this option so that you can view the count of packets being captured for each protocol.
 - **Enable MAC name resolution:** Wireshark contains a table to resolve MAC addresses to vendors. Leave enabled.
 - **Enable network name resolution:** Wireshark will issue DNS queries to resolve IP host names. Also will attempt to resolve network network names for other protocols. Leave disabled.
 - **Enable transport name resolution:** Wireshark will attempt to resolve transport names. Leave disabled.
4. Now click the **Start** button to start the capture.
 5. Recreate the problem. The capture dialog should show the number of packets increasing. If not, then stop the capture. Examine the interface list and pick the one that is not associated with the WANIP. It will probably be a long alpha-numeric string. If packets are still not being captured, try removing any filters that have been defined.
 6. Once the problem which is to be analyzed has been reproduced, click on **Stop**. It might take a few seconds for Wireshark to display the packets captured.

If the destination address is always displayed as FFFFFFFF (IPX) or always ends in

.255 (IP) then all that has been captured is broadcast traffic. **This is a useless trace.** This usually occurs when another machine is being traced (to start the trace while the target machine is powered off, in order to capture the bootup process). The capture setup needs to be reconsidered - port mirroring on the switch may need to be set up, or a dumb hub may need to be used to make the traffic reach the sniffing system. (Some devices advertised as "hubs" are in fact switches that may have the intelligence to prevent the workstations from seeing each other's packets; with these, getting a good trace may not be possible)

The Wireshark website has a good FAQ on this subject. Please refer to <http://www.wireshark.org/faq.html#q7.1>

7. Save the packet trace in any supported format. Just click on the **File** menu option and select **Save As**. By default Wireshark will save the packet trace in libpcap format. This is a filename with a.pcap extension. Use this default for files sent to Novell.
8. Create a trace_info.txt file with the IP and MAC address of the machines that are being traced as well as any pertinent information, such as:
 - What is the problem? (when did it start? steps to reproduce? any other pertinent information)
 - What steps were traced?
 - Give names of the servers and files being accessed.
 - If analysis of the trace has already been attempted, please provide Novell Support with analysis notes.

For example: Packets 1-30 are boot. Packets 31-500 are login. Packets 501 to 1,000 is my application loading. Packet 1,001 to 1,500 is me saving my file. The error occurred at approximately packet 1,480.

- Give the MAC addresses of hardware involved? (Workstation, servers, printers ...)
- What is the workstation OS and configuration?
- What version of client software is running?
- If it works with one version of the client (or a particular server patch), then get a trace of it working, and a trace of it not working.
- For Novell Client issues: Are there any client patches loaded?
- For Novell servers: What version of NetWare/OES (and other relevant products i.e. ZEN or NDPS) are running on the server?
- What patches have been applied?
- What is the configuration of the network? Are there routers involved? If so, what kind of routers?

Assignment Questions:

Part 1

1. Is your browser running HTTP version 1.0 or 1.1? What version of HTTP is the server running?
2. What languages (if any) does your browser indicate that it can accept to the server?
3. What is the IP address of your computer?
4. What is the status code returned from the server to your browser?
5. When was the HTML file that you are retrieving last modified at the server?
6. How many bytes of content are being returned to your browser?
7. By inspecting the raw data in the packet content window, do you see any headers within the data that are not displayed in the packet-listing window? If so, name one.

Part 2

8. Inspect the contents of the first HTTP GET request from your browser to the server. Do you see an "IF-MODIFIED-SINCE" line in the HTTP GET?
9. Inspect the contents of the server response. Did the server explicitly return the Contents of the file? How can you tell?
10. Now inspect the contents of the second HTTP GET request from your browser to the server. Do you see an "IF-MODIFIED-SINCE:" line in the HTTP GET? If so, what information follows the "IF-MODIFIED-SINCE:" header?
11. What is the HTTP status code and phrase returned from the server in response to this second HTTP GET? Did the server explicitly return the contents of the file? Explain.

Part 3

12. How many HTTP GET request messages did your browser send? Which packet number in the trace contains the GET message for the Bill of Rights?
13. Which packet number in the trace contains the status code and phrase associated with the response to the HTTP GET request?
14. What is the status code and Phrase in the response?
15. How many data-containing TCP segments were needed to carry the single HTTP response and the text of the Bill of Rights?

Part 4

16. How many HTTP GET request messages did your browser send? To which Internet addresses were these GET requests sent?
17. Can you tell whether your browser downloaded the two images serially, or whether they were downloaded from the two web sites in parallel? Explain.

Part 5

Let's try visiting a web site that is password-protected and examine the sequence of HTTP message exchanged for such a site. The URL http://gaia.cs.umass.edu/wireshark-labs/protected_pages/HTTP-wireshark-file5.html is password protected. The username is "wireshark-students" (without the quotes), and the password is "network" (again, without the quotes). So let's access this "secure"

Password-protected site. Do the following:

- Make sure your browser's cache is cleared, as discussed above, and close down

your browser. Then, start up your browser

- Start up the Wireshark packet sniffer
- Enter the following URL into your browser

http://gaia.cs.umass.edu/wireshark-labs/protected_pages/HTTP-wiresharkfile5.

Html Type the requested user name and password into the pop up box.

- Stop Wireshark packet capture, and enter “http” in the display-filter-specification window, so that only captured HTTP messages will be displayed later in the packet-listing window.

•(Note: If you are unable to run Wireshark on a live network connection, you can use the http-ethereal-trace-5 packet trace to answer the questions below; see footnote 2. This trace file was gathered while performing the steps above on one of the author’s computers.)

Now let’s examine the Wireshark output. You might want to first read up on HTTP authentication by reviewing the easy-to-read material on “HTTP Access Authentication Framework” at [http://frontier.userland.com/stories/storyReader\\$2159](http://frontier.userland.com/stories/storyReader$2159)

18. What is the server’s response (status code and phrase) in response to the initial HTTP GET message from your browser?

19. When your browser’s sends the HTTP GET message for the second time, what new field is included in the HTTP GET message?

OUTPUT

Answers:

1. Version 1.1

2. Languages supported en-us and en

3. 192.168.1.102

4.200 Ok

5.73 bytes

6. Last-Modified: Tue, 23 Sep 2003 05:29:00 GMT

7.No

8. NO

9. Yes, because it return's text/html on the webpage

10. yes, it tells the last modification date and time

11. Status code: 304 No it does not return any information explicitly as we cannot see any line based text data or any other return type.

12. one, packet no. 8

13.Packet no: 14

14. status code : 200 phrase: OK

15. 4 TCP segments

16. 3 HTTP GET request

IP1: 128.119.245.12 IP2: 165.193.1.102 IP3: 134.241.6.82

17. The browser downloaded the images serially as the arrival times of both the images are different and they are in separate tcp packet.

18. STATUS CODE: 401 PHRASES: Authorization Required

19. Authorization field

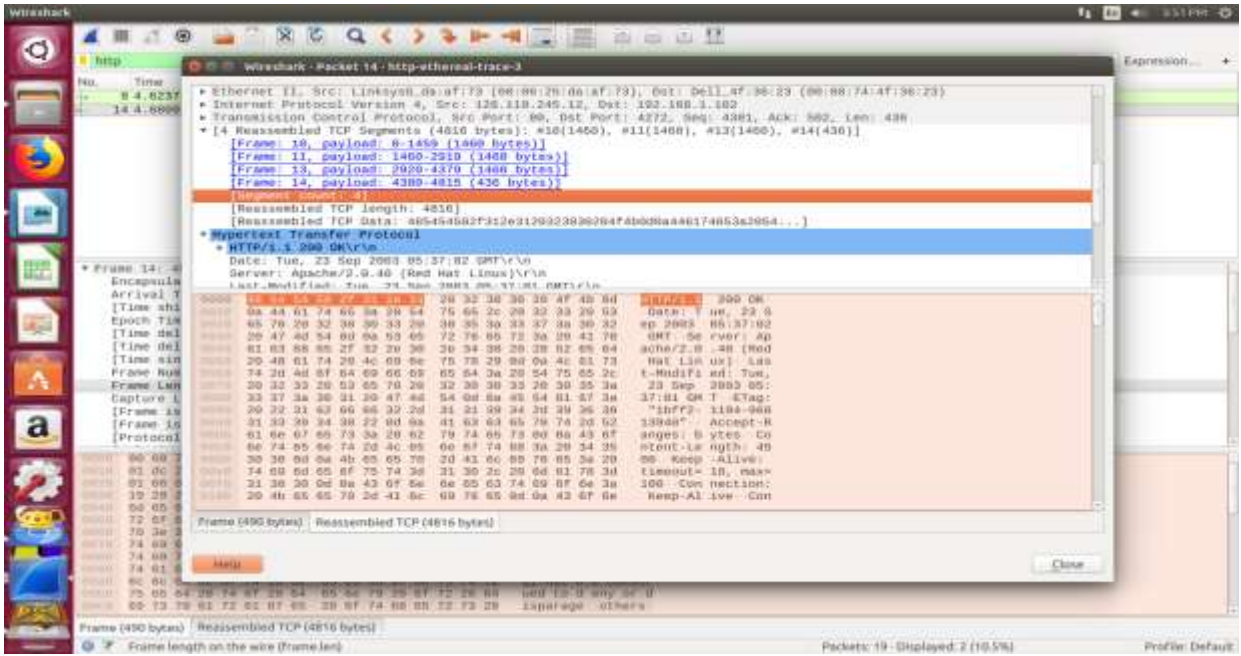


Figure 5: Snapshot of question 15.

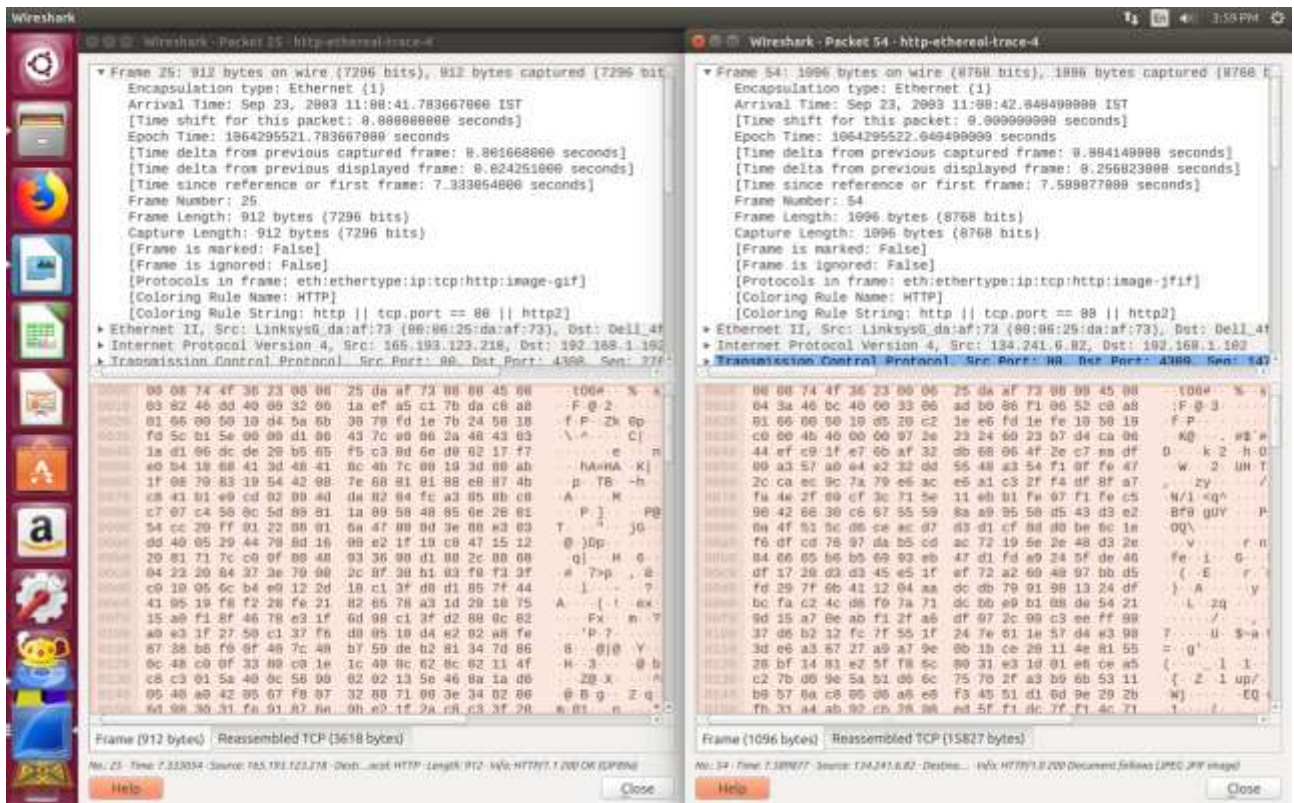


Figure 6: Snapshot of question 17.

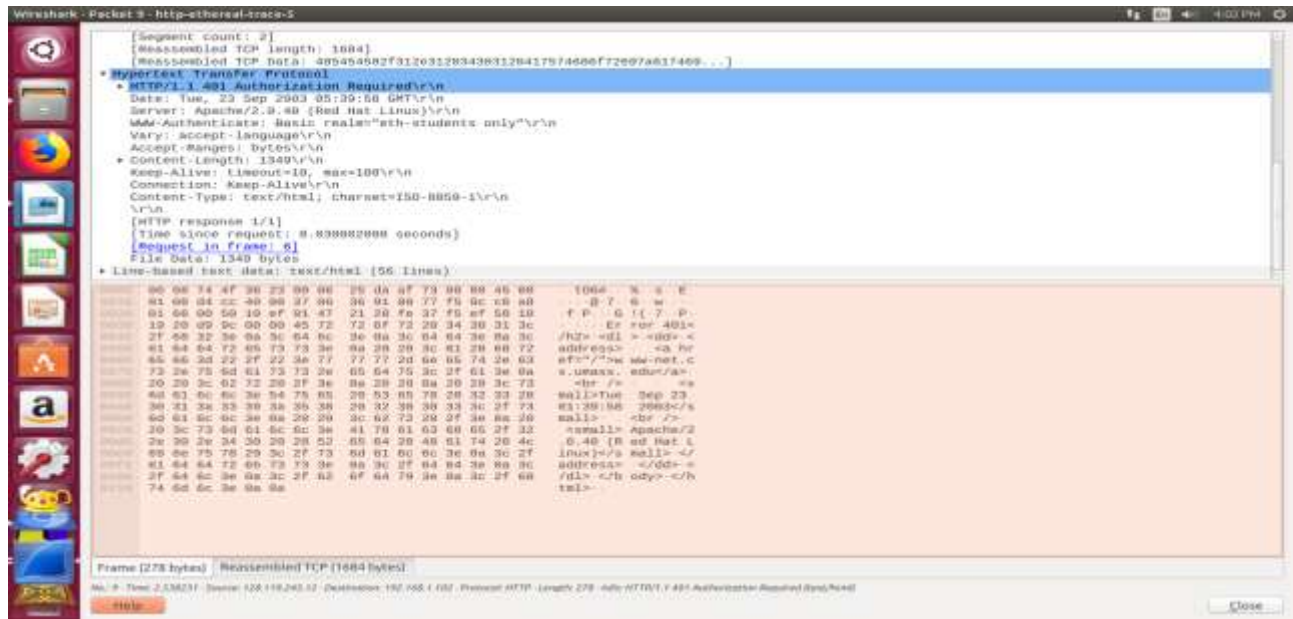


Figure7: Snapshot of question 18.

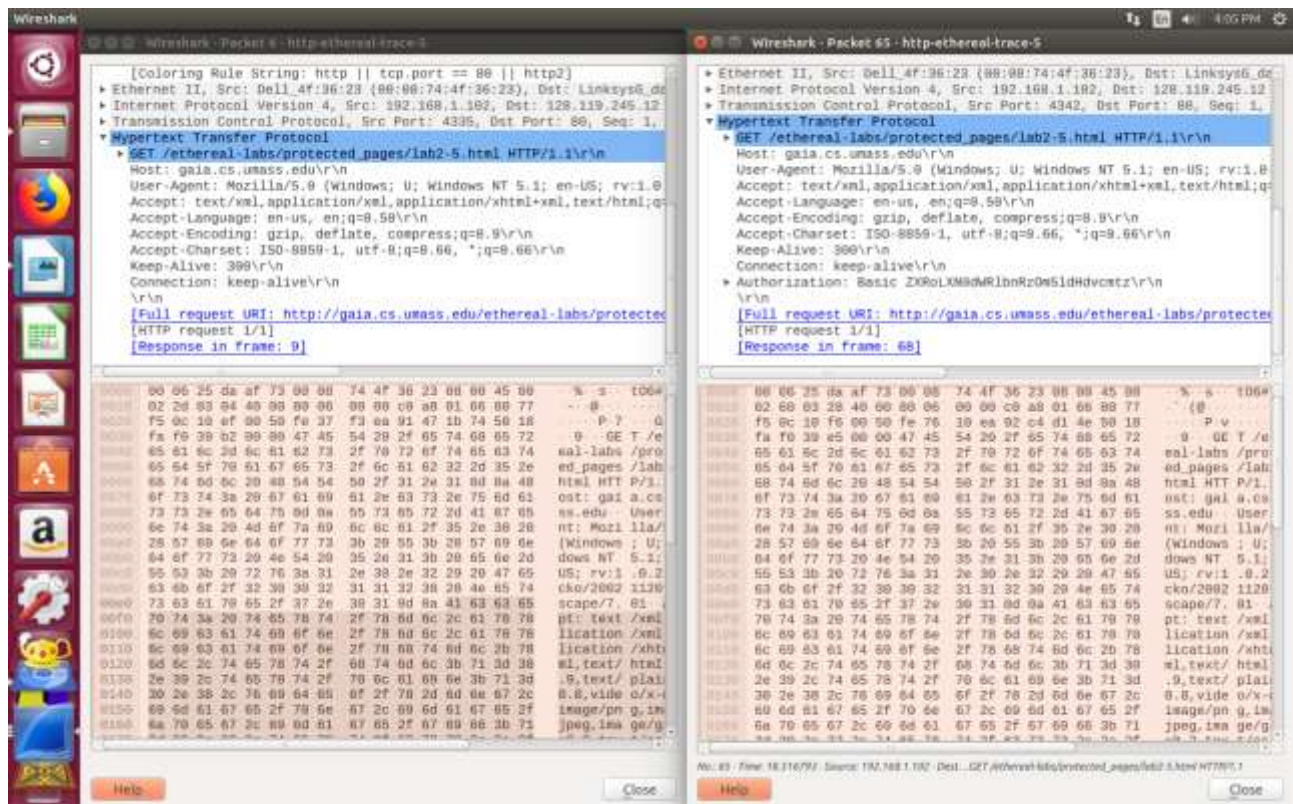


Figure7: Snapshot of question 19.

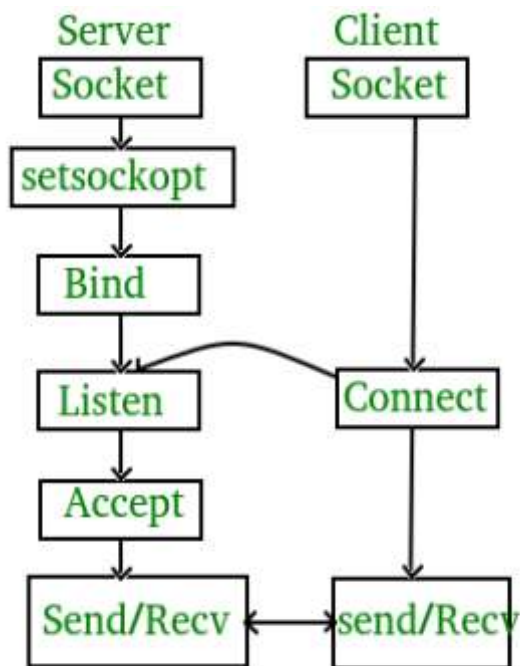
Program No. 7

OBJECTIVE: To implement socket programming using TCP.

THEORY:

Socket programming is a way of connecting two nodes on a network to communicate with each other. One socket (node) listens on a particular port at an IP, while other socket reaches out to the other to form a connection. Server forms the listener socket while client reaches out to the server.

Figure 1: State diagram for server and client model



Stages for server

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)

Setsockopt:

```
int setsockopt(int sockfd, int level, int optname,  
const void *optval, socklen_t optlen);
```

This helps in manipulating options for the socket referred by the file descriptor sockfd. This is completely optional, but it helps in reuse of address and port. Prevents error such as: "address already in use".

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.

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.

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

Socket connection: Exactly same as that of server's socket creation

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.

The steps involved in establishing a socket on the client side are as follows:

1. Create a socket with the socket() system call.
2. Connect the socket to the address of the server using the connect() system call
3. Send and receive data. There are a number of ways to do this, but the simplest is to use the read() and write() system calls.

The steps involved in establishing a socket on the server side are as follows:

1. Create a socket with the socket() system call
2. Bind the socket to an address using the bind() system call. For a server socket on the Internet, an address consists of a port number on the host machine.
3. Listen for connections with the listen() system call
4. Accept a connection with the accept() system call. This call typically blocks until a client connects with the server.
5. Send and receive data