Assignment No: 1

Title: Network connecting devices and addresses

Learning Objectives:

- To understand the structure and working of various networks including the interconnecting devices used in them.
- To understand different addresses

Problem Statement:
To study working principles of all network connecting devices and IP, MAC and Port address

Learning Outcome: Students will able to

- Describe various networks interconnecting devices.
- Differentiate IP, MAC and PORT Addresses

Software and Hardware Requirement:

- PC with Network

Theory:

1. Repeater – A repeater operates at the physical layer. Its job is to regenerate the signal over the same network before the signal becomes too weak or corrupted so as to extend the length to which the signal can be transmitted over the same network. An important point to be noted about repeaters is that they do no amplify the signal. When the signal becomes weak, they copy the signal bit by bit and regenerate it at the original strength. It is a 2 port device.

2. Hub – A hub is basically a multiport repeater. A hub connects multiple wires coming from different branches, for example, the connector in star topology which connects different stations. Hubs cannot filter data, so data packets are sent to all connected devices. In other words, collision domain of all hosts connected through Hub remains one. Also, they do not have intelligence to find out best path for data packets which leads to inefficiencies and wastage. Hub falls in two categories:

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

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

3. Bridge – A bridge operates at data link layer. A bridge is a repeater, with add on functionality of filtering content by reading the MAC addresses of source and destination. It is also used for interconnecting two LANs working on the same protocol. It has a single input and single output port, thus making it a 2 port device. There are mainly three types in which bridges can be characterized:

Transparent Bridge: As the name signifies, it appears to be transparent for the other devices on the network. The other devices are ignorant of its existence. It only blocks or forwards the data as per the MAC address.

Source Route Bridge: It derives its name from the fact that the path which packet takes through the network is implanted within the packet. It is mainly used in Token ring networks.

Translational Bridge: The process of conversion takes place via Translational Bridge. It converts the data format of one networking to another. For instance Token ring to Ethernet and vice versa.

4. Switch – A switch is a multi port bridge with a buffer and a design that can boost its efficiency (large number of ports imply less traffic) and performance. Switch is data link layer device. Switch can perform error checking before forwarding data, that makes it very efficient as it does not forward packets that have errors and forward good packets selectively to correct port only. In other words, switch divides collision domain of hosts, but broadcast domain remains same. The following method will elucidate further how data transmission takes place via switches:

Cut-through transmission: It allows the packets to be forwarded as soon as they are received. The method is prompt and quick but the possibility of error checking gets overlooked in such kind of packet data transmission.

Store and forward: In this switching environment the entire packet are received and 'checked' before being forwarded ahead. The errors are thus eliminated before being propagated further. The downside of this process is that error checking takes relatively longer time consequently making it a bit slower in processing and delivering.

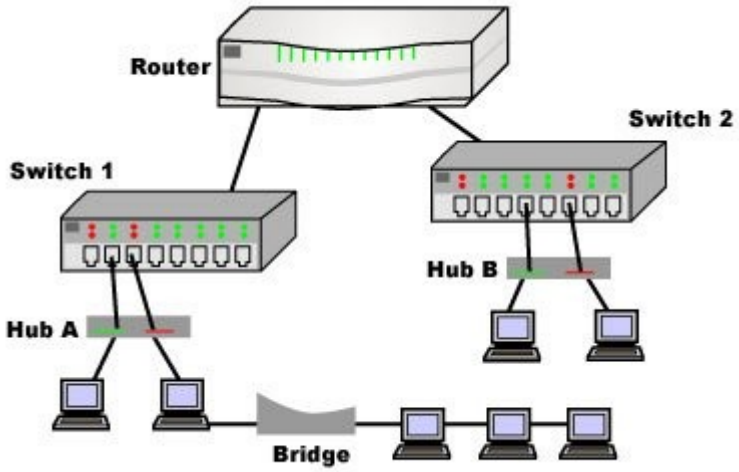
Fragment Free: In a fragment free switching environment, a greater part of the packet is examined so that the switch can determine whether the packet has been caught up in a collision. After the collision status is determined, the packet is forwarded.

5. Routers – A router is a device like a switch that routes data packets based on their IP addresses. Router is mainly a Network Layer device. Routers normally connect LANs and WANs together and have a dynamically updating routing table based on which they make decisions on routing the data packets. Router divide broadcast domains of hosts connected through it. When a router receives the data, it determines the destination address by reading the header of the packet.

Once the address is determined, it searches in its routing table to get know how to reach the destination and then forwards the packet to the higher hop on the route. The hop could be the final destination or another router. The two ways through which a router can receive information are:

Static Routing: In static routing, the routing information is fed into the routing tables manually. It does not only become a time-taking task but gets prone to errors as well. The manual updating is also required in case of statically configured routers when change in the topology of the network or in the layout takes place. Thus static routing is feasible for tinniest environments with minimum of one or two routers.

Dynamic Routing: For larger environment dynamic routing proves to be the practical solution. The process involves use of peculiar routing protocols to hold communication. The purpose of these protocols is to enable the other routers to transfer information about to other routers, so that the other routers can build their own routing tables.



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

7. IP address - An IP address or Internet Protocol address is the address assigned to your mobile, printer or computer by the network that uses Internet protocol for communication. Your IP can change with the change in network. IP addresses are divided into classes. A,B,C,D,E mostly we use class B and D
If you want to find your IP address open command prompt and enter ipconfig on windows or ifconfig in our linux or mac terminal you will find your IP4 and IP6 address.

The following table lists the important differences between IPv4 and IPv6.

IPv4	**IPv6**
IPv4 addresses are 32 bit length.	IPv6 addresses are 128 bit length.
IPv4 addresses are binary numbers represented in decimals.	IPv6 addresses are binary numbers represented in hexadecimals.
IPSec support is only optional.	Inbuilt IPSec support.
Fragmentation is done by sender and forwarding routers.	Fragmentation is done only by sender.
No packet flow identification.	Packet flow identification is available within the IPv6 headerusing the Flow Label field.
Checksum field is available in IPv4 header	No checksum field in IPv6 header.
Options fields are available in IPv4 header.	No option fields, but IPv6 Extension headers are available.
Address Resolution Protocol (ARP) is available to map IPv4 addresses to MAC addresses.	Address Resolution Protocol (ARP) is replaced with a function of Neighbor Discovery Protocol (NDP).
Internet Group Management Protocol (IGMP) is used to manage multicast group membership.	IGMP is replaced with Multicast Listener Discovery (MLD) messages.
Broadcast messages are available.	Broadcast messages are not available. Instead a link-local scope "All nodes" multicast IPv6 address (FF02::1) is used for broadcast similar functionality.
Manual configuration (Static) of IPv4 addresses or DHCP (Dynamic configuration) is required to configure IPv4 addresses.	Auto-configuration of addresses is available.

8.MAC Address-The MAC address is a hardware address, which means it is unique to the network card installed on your PC. No two devices on a local network should ever have the same MAC address. In the unlikely event this occurs, the two devices will have major communication problems. During the manufacturing process, the vendor "burns" a specific MAC address into each network card's ROM. When the serial numbers have all been used, they start from the beginning, as it's very unlikely anyone would buy two network cards from the same vendor, and they will contain the same MAC address. For a laptop with WiFi, there may be 2 MAC address, one for WiFi and the other for the Ethernet Port. It is a hexadecimal address. An example of a MAC address: DE-56-0A-DC-E6-88.

9.Port-In programming, a port is a "logical connection place" and specifically, using the Internet's protocol, TCP/IP, the way a client program specifies a particular server program on a computer in a network. Higher-level applications that use TCP/IP such as the Web protocol, Hypertext Transfer Protocol, have ports with pre-assigned numbers. Port numbers are from 0 to 65535. Ports 0 to 1024 are reserved for use by certain privileged services. For the HTTP service, port 80 is defined as a default

A port number is used at a higher layer of communication, primarily to differentiate between the "Types" of communication that is happening. For example, you open a web browser, and have skype running at the same time; http web browsing starts a "session" to the server's Port no. 80, while skype will start a different "session" towards skype's servers with a different destination port number. Each different application or program you are concurrently running will have a different source and destination port from your computer/device to the servers on the internet hosting each service.

Algorithm:

Not Applicable

Application:

- Networking devices are used to interconnect different pc.

Conclusion

Hence we studied the working principle of various network devices and addresses used at OSI layer/TCP/IP

Assignment No: 2

Title: Demonstration of Ping and Wireshark Packet Analyzer Tool

Learning Objectives:

- To configure machine using IP address
- To get hands on experience of making and testing cables
- To setup a LAN using switch
- To analyze packet captured through Wireshark

Problem Statement:

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

Part B: Extend the same Assignment for Wireless using Access Point

Learning Outcome: Students will able to

- Setup wired and Wi-Fi network

Software and Hardware Requirement:

- PC with Network
- Cables and Switch
- Wireshark
- PC with Linux Operation System

Theory:

LAN - Local Area Network

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

MAN-Metropolitan Area Network

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

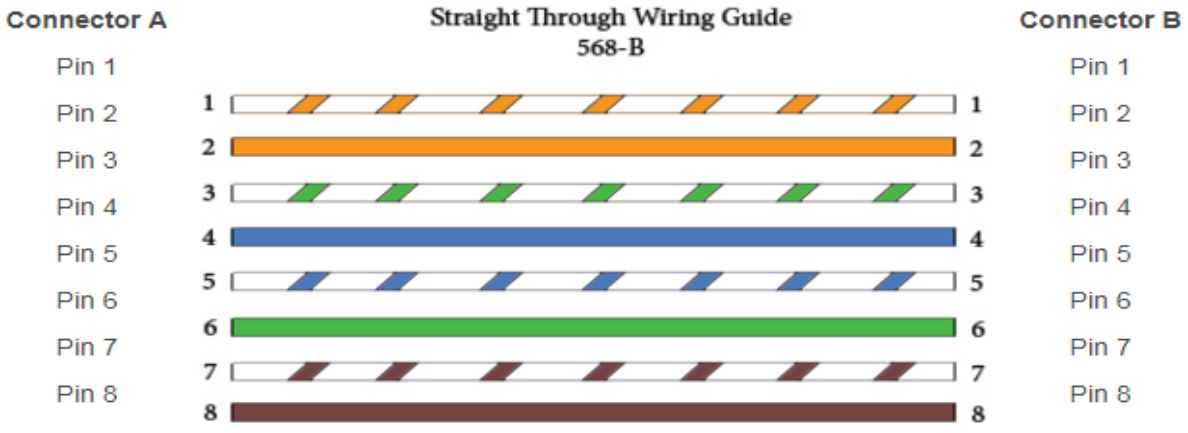
WAN - Wide Area Network

As the term implies, a WAN spans a large physical distance. The Internet is the largest WAN, spanning the Earth.

A WAN is a geographically-dispersed collection of LANs. A network device called a router connects LANs to a WAN. In IP networking, the router maintains both a LAN address and a WAN address.

The Purpose of Straight-Through Cables

Straight-Through refers to cables that have the pin assignments on each end of the cable. In other words Pin 1 connector A goes to Pin 1 on connector B, Pin 2 to Pin 2 ect. Straight-Through wired cables are most commonly used to connect a host to client. When we talk about cat5e patch cables, the Straight-Through wired cat5e patch cable is used to connect computers, printers and other network client devices to the router switch or hub (the host device in this instance).

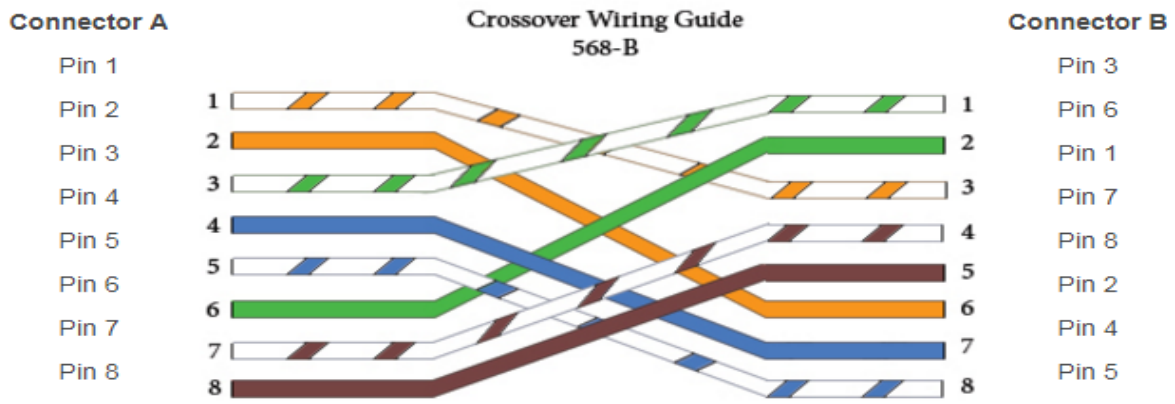


Use a straight-through cable when:

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

The purpose of Crossover Cables

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

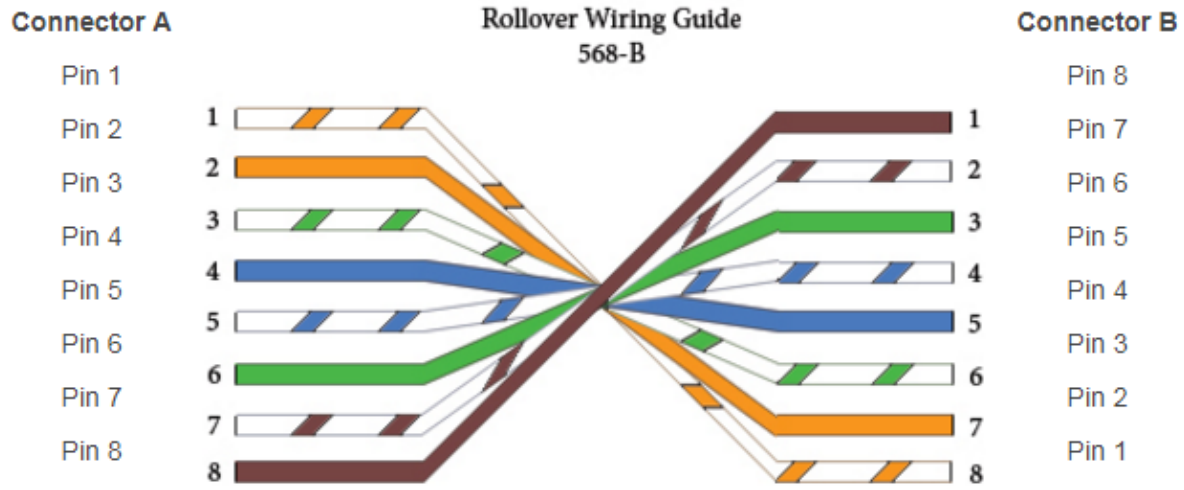


Use a crossover cable when:

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

The Purpose of Rollover Cables

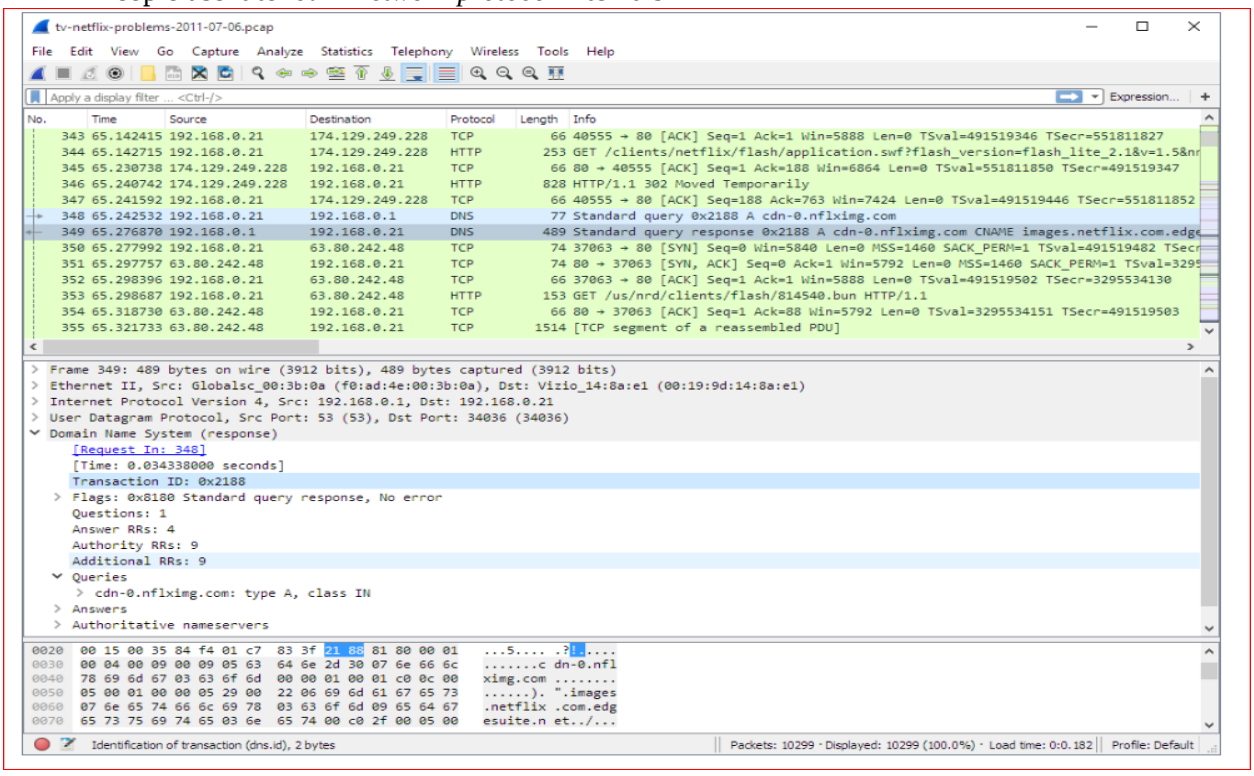
Rollover wired cables most commonly called rollover cables, have opposite Pin assignments on each end of the cable or in other words it is "rolled over". Pin 1 of connector A would be connected to Pin 8 of connector B. Pin 2 of connector A would be connected to Pin 7 of connector B and so on. Rollover cables, sometimes referred to as Yost cables are most commonly used to connect to a devices console port to make programming changes to the device. Unlike crossover and straight-wired cables, rollover cables are not intended to carry data but instead create an interface with the device.



Wireshark

Wireshark is a network packet analyzer. A network packet analyzer will try to capture network packets and tries to display that packet data as detailed as possible. Here are some examples people use Wireshark for:

- Network administrators use it to *troubleshoot network problems*
- Network security engineers use it to *examine security problems*
- Developers use it to *debug protocol implementations*
- People use it to *learn network protocol* internals



PING:

Packet InterNet Groper is a computer program that is used to test the network connectivity between two systems. ping is utility to determine whether a specific IP address is accessible or not. PING is used primarily to trouble shoot Internet connections. These systems can be a normal Personal computer, a Large Server, Router, switch or a gateway. It was created to verify whether a specific computer on a network exists on not. A series of packets are sent form one system to another over a network to test its connectivity, in order to generate a response from the other system. The other system reply with an acknowledgment. This will tell the sender that two systems are connected with each other, through an existing network. A single PING can measure the connectivity based on multiple parameters including minimum, maximum, and the mean round-trip time.

Algorithm:

Not Applicable

Application:

- Networking devices are used to interconnect different PCs.

Conclusion

Hence we studied testing of cables and setting up a wired LAN and used ping utility to test the network.

Assignment No: 3

Title: Go back N and Selective Repeat Sliding Window Protocol

Learning Objectives:

- To understand the working of Go Back N and Selective Repeat Sliding Window protocol

Problem Statement:
Write a program to simulate Go back N and Selective Repeat Modes of Sliding Window Protocol in peer to peer mode and demonstrate the packets captured traces using Wireshark Packet Analyzer Tool for peer to peer mode.

Learning Outcome: Students will able to

- Write code and Simulate Go Back N and Selective Repeat Sliding Window protocol

Software and Hardware Requirement:

- PC with network
- Linux Operating system
- Netbeans/Eclipse

Theory:
Go-Back-N ARQ

Go-Back-N ARQ is a specific instance of the automatic repeat request (ARQ) protocol, in which the sending process continues to send a number of frames specified by a window size even without receiving an acknowledgement (ACK) packet from the receiver. It is a special case of the general sliding window protocol with the transmit window size of N and receive window size of 1. It can transmit N frames to the peer before requiring an ACK.

The receiver process keeps track of the sequence number of the next frame it expects to receive, and sends that number with every ACK it sends. The receiver will discard any frame that does not have the exact sequence number it expects (either a duplicate frame it already acknowledged, or an out-of-order frame it expects to receive later) and will resend an ACK for the last correct in-order frame.[1] Once the sender has sent all of the frames in its window, it will detect that all of the frames since the first lost frame are outstanding, and will go back to the sequence number of the last ACK it received from the receiver process and fill its window starting with that frame and continue the process over again.

Go-Back-N ARQ is a more efficient use of a connection than Stop-and-wait ARQ, since unlike waiting for an acknowledgement for each packet; the connection is still being utilized as packets are being sent. In other words, during the time that would otherwise be spent waiting, more packets are being sent. However, this method also results in sending frames multiple times – if any frame was lost or damaged, or the ACK acknowledging them was lost or damaged, then that frame and all following frames in the window (even if they were received without error) will be re-sent. To avoid this, Selective Repeat ARQ can be used.

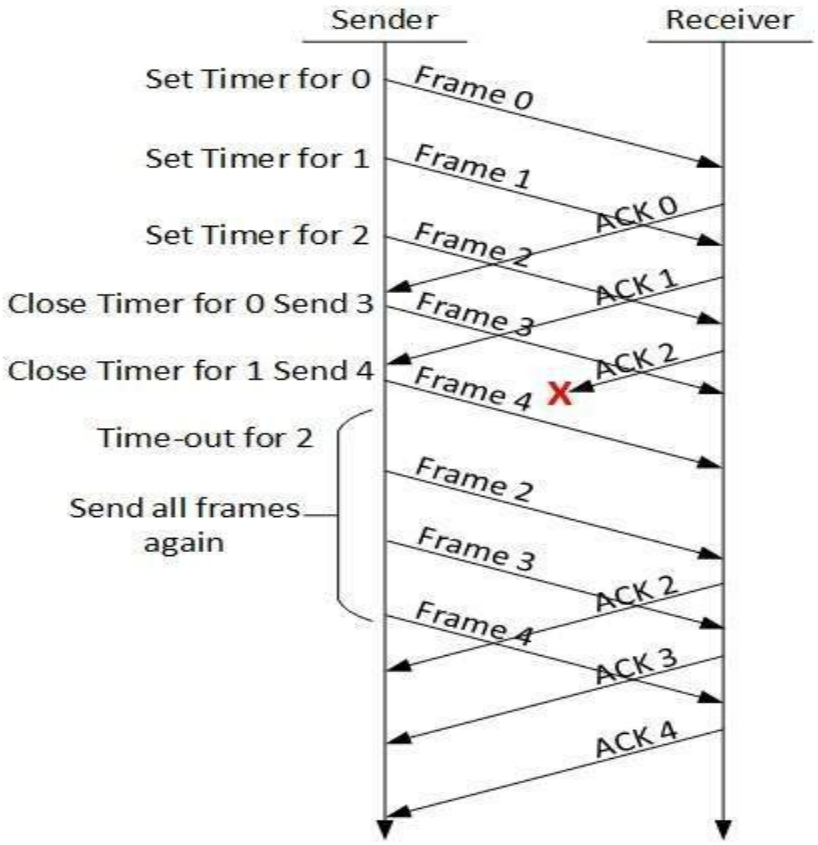


Fig: Go Back N Protocol

Selective Repeat ARQ

Selective Repeat ARQ is a specific instance of the Automatic Repeat-reQuest (ARQ) Protocol, in which the sending process continues to send a number of frames specified by a *window size* even after a frame loss. Unlike Go-Back-N ARQ, the receiving process will continue to accept and acknowledge frames sent after an initial error. The receiver process keeps track of the sequence number of the earliest frame it has not received, and sends that number with every ACK it sends. If a frame from the sender does not reach the receiver, the sender continues to send subsequent frames until it has emptied its *window*. The receiver continues to fill its receiving window with the subsequent frames, replying each time with an ACK containing the sequence number of the earliest missing frame. Once the sender has sent all the frames in its *window*, it resends the frame number given by the ACKs, and then continues where it left off.

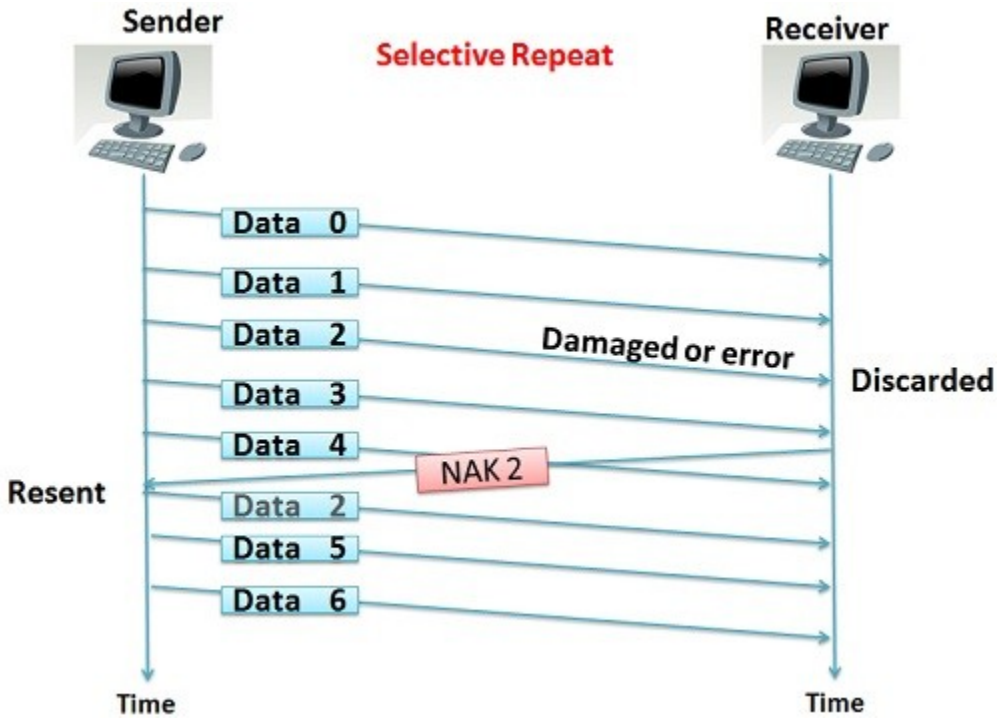


Fig: Selective Repeat Protocol

The size of the sending and recieving windows must be equal, and half the maximum sequence number (assuming that sequence numbers are numbered from 0 to n-1) to avoid miscommunication in all cases of packets being dropped. The sender moves its window for every packet that is acknowledged.

Algorithm:

GO Back N
N=Window Size
Sn = sequence number
Sb = sequence base
Sm = sequence max
ack = ack number
nack = first non acknowledged

Receiver:
Do the following forever:
Randomly accept or reject packet

If the packet received and the packet is error free
Accept packet
Send a positive ack for packet
Else
Refuse packet
Send a negative ack for packet

Sender:
Sb = 0
Sm = N − 1

```
ack = 0
Repeat the following steps forever:
  Send packet with ack
  If positively ack is recieved:
        ack++
        Transmit a packet where Sb <= ack <= Sm.
        packets are transmitted in order
  Else
         Enqueue the nack into the queue
  //check if last packet in the window is sent
  if(ack==Sm)
    if(queue is not empty)
         // start from the first nack packet
        nack = queue.front();
        empty the queue
        ack = nack

    Sm = Sm + (ack − Sb)
    Sb = ack
```

Selective Repeat

```
N  = window size
Sn = sequence number
Sb = sequence base
Sm = sequence max
ack = ack number
nack = first non acknowledged
```

Receiver:

```
Do the following forever:
  Randomly accept or reject packet

  If the packet received and the packet is error free
       Accept packet
       Send a positive ack for packet
  Else
       Refuse packet
       Send a negative ack for packet
```

Sender:

```
Sb = 0
Sm = N − 1
ack = 0
Repeat the following steps forever:
  If the packet was not already positively acknowledged by receiver
         Send packet with ack
         If positively ack is recieved:
             Transmit a packet where Sb <= ack <= Sm.
             packets are transmitted in order
         Else
             Enqueue the nack into the queue
         ack++
```

```
//check if last packet in the window is sent
if(ack==Sm)
  if(queue is not empty)
       // start from the first nack packet
      nack = queue.front();
      empty the queue
      ack = nack

  Sm = Sm + (ack − Sb)
  Sb = ack
```

Application:
The selective repeat is a more efficient protocol as it does not waste bandwidth for the frames which are properly received but, its complexity and expense favors the use of the go-back-n protocol.

Conclusion
Hence we studied how packet transmission takes place in Go Back-N and Selective Repeat ARQ.

Assignment No: 4

Title: Sub-netting and subnet mask

Learning Objectives:

- To understand the need of subnetting.
- To compute subnet mask .
- To compute Network Address, broadcast address and range of host addresses in a subnet.

Problem Statement:
Write a program to demonstrate sub-netting and find the subnet masks

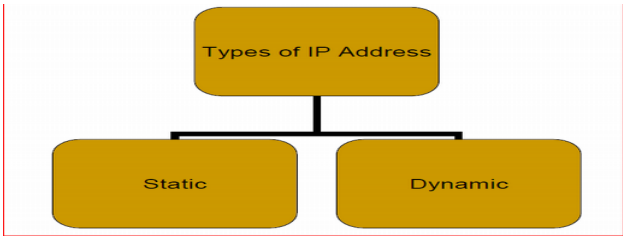
Learning Outcome: Students will able to

- Understand the need of subnetting
- Compute subnet mask
- Compute Network Address ,broadcast address and range of addresses in a subnet

Software and Hardware Requirement:

- PC with network
- Linux Operating system
- Netbeans/Eclipse

Theory:
A Unique, 32-bit address used by computers to communicate over a computer network is called IP address. A machine can have one of the types of IP address.

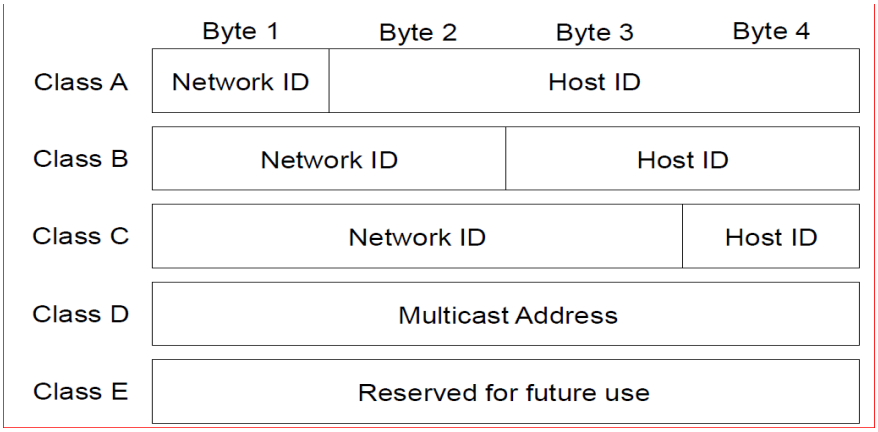


Classes of Address

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

Class	A	-	0.0.0.0	-	127.255.255.255
Class	B	-	128.0.0.0	-	191.255.255.255
Class	C	-	192.0.0.0	-	223.255.255.255
Class	D	-	224.0.0.0	-	239.255.255.255

Class E - 240.0.0.0 - 255.255.255.255



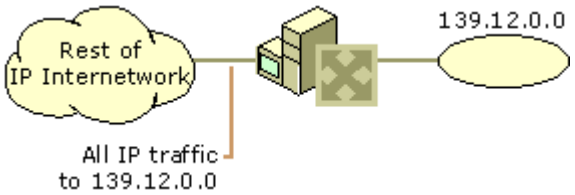
Need of subnetting?

The Internet Address Classes accommodate three scales of IP internetworks, where the 32-bits of the IP address are apportioned between network IDs and host IDs depending on how many networks and hosts per network are needed. However, consider the class A network ID, which has the possibility of over 16 million hosts on the same network. All the hosts on the same physical network bounded by IP routers share the same broadcast traffic; they are in the same broadcast domain. It is not practical to have 16 million nodes in the same broadcast domain. The result is that most of the 16 million host addresses are unassignable and are wasted. Even a class B network with 65 thousand hosts is impractical.

In an effort to create smaller broadcast domains and to better utilize the bits in the host ID, an IP network can be subdivided into smaller networks, each bounded by an IP router and assigned a new subnetted network ID, which is a subset of the original class-based network ID.

This creates *subnets*, subdivisions of an IP network each with their own unique subnetted network ID. Subnetted network IDs are created by using bits from the host ID portion of the original class-based network ID.

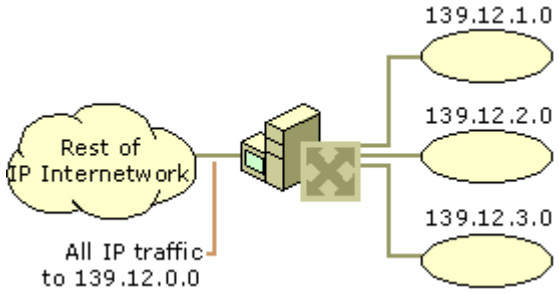
Consider the example in Fig. The class B network of 139.12.0.0 can have up to 65,534 nodes. This is far too many nodes, and in fact the current network is becoming saturated with broadcast traffic. The subnetting of network 139.12.0.0 should be done in such a way so that it does not impact nor require the reconfiguration of the rest of the IP internetwork.



Network 139.12.0.0 Before Subnetting

Network 139.12.0.0 is subnetted by utilizing the first 8 host bits (the third octet) for the new subnetted network ID. When 139.12.0.0 is subnetted, as shown in Figure, separate networks with their own subnetted network IDs (139.12.1.0, 139.12.2.0, 139.12.3.0) are created. The router is aware of the separate subnetted networks IDs and routes IP packets to the appropriate subnet.

Note that the rest of the IP internetwork still regards all the nodes on the three subnets as being on network 139.12.0.0. The other routers in the IP internetwork are unaware of the subnetting being done on network 139.12.0.0 and therefore require no reconfiguration.



Network 139.12.0.0 After Subnetting

Subnet Masks

With the advent of subnetting, one can no longer rely on the definition of the IP address classes to determine the network ID in the IP address. A new value is needed to define which part of the IP address is the network ID and which part is the host ID regardless of whether class-based or subnetted network IDs are being used.

RFC 950 defines the use of a *subnet mask* (also referred to as an address mask) as a 32-bit value that is used to distinguish the network ID from the host ID in an arbitrary IP address. The bits of the subnet mask are defined as follows:

- All bits that correspond to the network ID are set to 1.
- All bits that correspond to the host ID are set to 0.

Each host on a TCP/IP network requires a subnet mask even on a single segment network. Either a default subnet mask, which is used when using class-based network IDs, or a custom subnet mask, which is used when subnetting r supernetting, is configured on each TCP/IP node.

Address Class	Bits for Subnet Mask	Network Prefix
Class A	11111111 00000000 00000000 00000000	/8
Class B	11111111 11111111 00000000 00000000	/16
Class C	11111111 11111111 11111111 00000000	/24

Determining the Network ID

To extract the network ID from an arbitrary IP address using an arbitrary subnet mask, IP uses a mathematical operation called a logical AND comparison. In an AND comparison, the result of two items being compared is true only when both items being compared are true; otherwise, the result is false. Applying this principle to bits, the result is 1 when both bits being compared are 1, otherwise the result is 0.

IP performs a logical AND comparison with the 32-bit IP address and the 32-bit subnet mask. This operation is known as a bit-wise logical AND. The result of the bit-wise logical AND of the IP address and the subnet mask is the network ID.

For example, what is the network ID of the IP node 129.56.189.41 with a subnet mask of 255.255.240.0? To obtain the result, turn both numbers into their binary equivalents and line them up. Then perform the AND operation on each bit and write down the result.

```
10000001 00111000 10111101 00101001 IP Address
11111111 11111111 11110000 00000000 Subnet Mask
10000001 00111000 10110000 00000000 Network ID
```

The result of the bit-wise logical AND of the 32 bits of the IP address and the subnet mask is the network ID 129.56.176.0.

Note that every network has a network name (which is the FIRST address of the network, this is also your subnet number), and a broadcast address (which is the LAST address of the network). These two addresses are reserved; you cannot use them for hosts.

Algorithm:

1. Enter IP address and subnet mask
2. Perform AND operation on IP address and subnet mask and compute Network ID
3. Compute and print broadcast address
4. Compute and print range of subnet and no. of nodes available in subnet

Application:

The main purpose of subnetting is to help relieve network congestion and improve network performance and speed.

Conclusion:

Hence we understand the process of subnetting and computation of network ID of subnet.

Assignment No: 5

Title: TCP Client Server communication using Socket Programming of C

Learning Objectives:

- To understand socket programming in C
- To establish a communication between client and server using TCP

Problem Statement:

Write a program using TCP socket for wired network for following
a. Say Hello to Each other (For all students)
b. File transfer (For all students)
c. Calculator (Arithmetic) (50% students)
d. Calculator (Trigonometry) (50% students)
Demonstrate the packets captured traces using Wireshark Packet Analyzer Tool for peer to peer mode.

Learning Outcome: Students will able to

- Write client and server code to establish a connection using TCP
- Perform calculation in between client and server.
- Transfer data and files in between client and server

Software and Hardware Requirement:

- PC with network
- Linux Operating system
- C compiler

Theory:

TCP:

TCP provides a connection oriented service, since it is based on connections between clients and servers as shown in fig.1. TCP provides reliability. When TCP client send data to the server, it requires an acknowledgement in return. If an acknowledgement is not received, TCP automatically retransmit the data and waits for a longer period of time.

The socket() Function

The first step is to call the socket function, specifying the type of communication protocol (TCP based on IPv4, TCP based on IPv6, UDP).

The function is defined as follows:

```
#include <sys/socket.h>
int socket (int family, int type, int protocol);
```

where family specifies the protocol family (AF_INET for the IPv4 protocols), type is a constant described the type of socket (SOCK_STREAM for stream sockets and SOCK_DGRAM for datagram sockets.

The function returns a non-negative integer number, similar to a file descriptor, that we define *socket descriptor* or -1 on error.

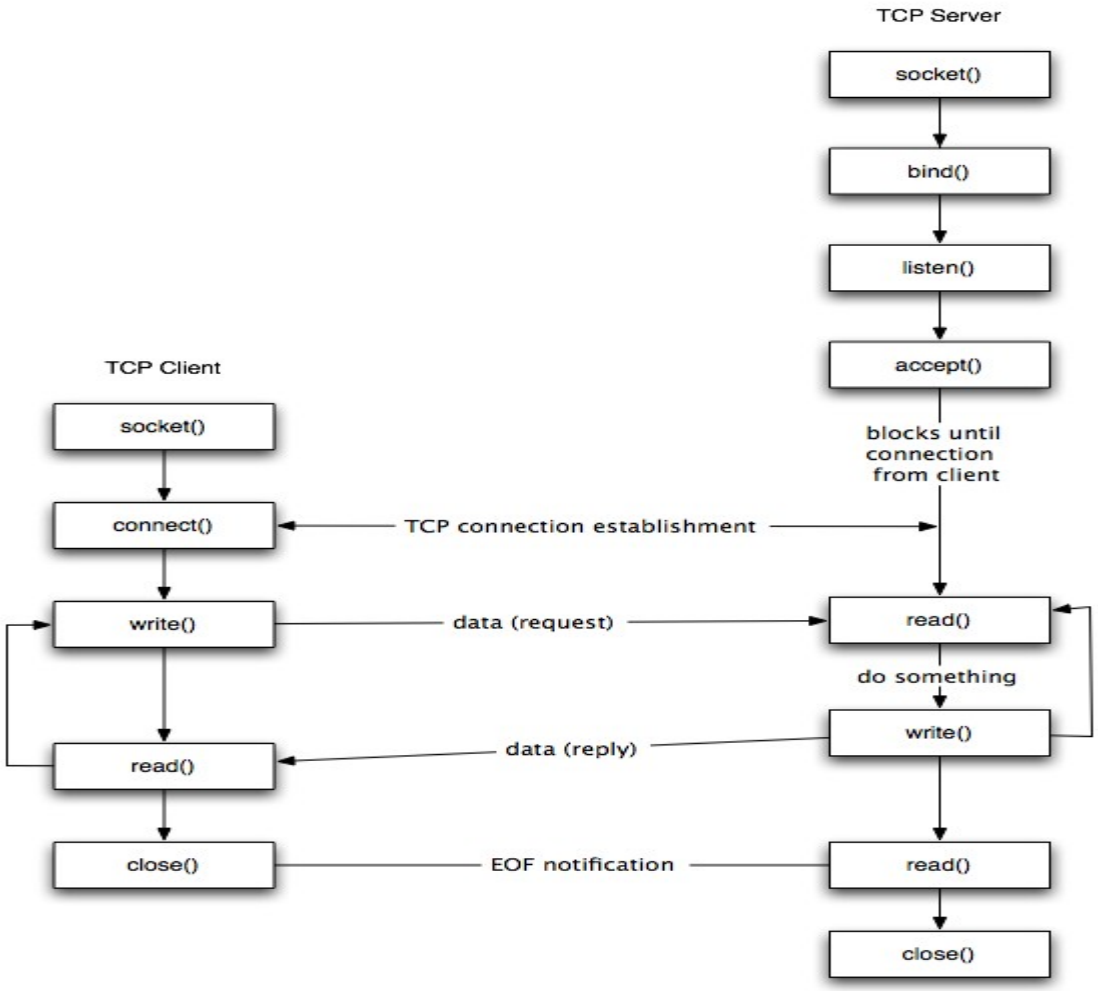


Fig. 1: TCP Client Server Communication

The connect() Function

The connect() function is used by a TCP client to establish a connection with a TCP server/

The function is defined as follows:

```
#include <sys/socket.h>
int connect (int sockfd, const struct sockaddr *servaddr, socklen_t addrlen);
```

where sockfd is the socket descriptor returned by the socket function.

The function returns 0 if establishing a connection (i.e., successful TCP three-way handshake, -1 otherwise.

The client does not have to call bind() in Section before calling this function: the kernel will choose both an ephemeral port and the source IP if necessary.

The bind() Function

The bind() assigns a local protocol address to a socket. With the Internet protocols, the address is the combination of an IPv4 or IPv6 address (32-bit or 128-bit) address along with a 16 bit TCP port number.

The function is defined as follows:

```
#include <sys/socket.h>
int bind(int sockfd, const struct sockaddr *servaddr, socklen_t addrlen);
```

where sockfd is the socket descriptor, servaddr is a pointer to a protocol-specific address and addrlen is the size of the address structure.

bind() returns 0 if it succeeds, -1 on error.

This use of the generic socket address sockaddr requires that any calls to these functions must cast the pointer to the protocol-specific address structure. For example for and IPv4 socket structure:

```
struct sockaddr_in serv; /* IPv4 socket address structure */
bind(sockfd, (struct sockaddr*) &serv, sizeof(serv))
```

A process can bind a specific IP address to its socket: for a TCP client, this assigns the source IP address that will be used for IP datagrams sent on the sockets. For a TCP server, this restricts the socket to receive incoming client connections destined only to that IP address.

The listen() Function

The listen() function converts an unconnected socket into a passive socket, indicating that the kernel should accept incoming connection requests directed to this socket. It is defined as follows:

```
#include <sys/socket.h>
int listen(int sockfd, int backlog);
```

where sockfd is the socket descriptor and backlog is the maximum number of connections the kernel should queue for this socket. The backlog argument provides an hint to the system of the number of outstanding connect requests that it should enqueue on behalf of the process. Once the queue is full, the system will reject additional connection requests. The backlog value must be chosen based on the expected load of the server.

The function listen() return 0 if it succeeds, -1 on error.

The accept() Function

The accept() is used to retrieve a connect request and convert that into a request. It is defined as follows:

```
#include <sys/socket.h>
int accept(int sockfd, struct sockaddr *cliaddr,socklen_t *addrlen);
```

where sockfd is a new file descriptor that is connected to the client that called the connect(). The cliaddr and addrlen arguments are used to return the protocol address of the client. The new socket descriptor has the same socket type and address family of the original socket. The original socket passed

to accept() is not associated with the connection, but instead remains available to receive additional connect requests. The kernel creates one connected socket for each client connection that is accepted.

If we don't care about the client's identity, we can set the cliaddr and addrlen to NULL. Otherwise, before calling the accept function, the cliaddr parameter has to be set to a buffer large enough to hold the address and set the interger pointed by addrlen to the size of the buffer.

The send() Function

Since a socket endpoint is represented as a file descriptor, we can use read and write to communicate with a socket as long as it is connected. However, if we want to specify options we need another set of functions.

For example, send() is similar to write() but allows to specify some options. send() is defined as follows:

```
#include <sys/socket.h>
ssize_t send(int sockfd, const void *buf, size_t nbytes, int flags);
```

where buf and nbytes have the same meaning as they have with write. The additional argument flags is used to specify how we want the data to be transmitted. We will not consider the possible options in this course. We will assume it equal to 0.

The function returns the number of bytes if it succeeds, -1 on error.

The receive() Function

The recv() function is similar to read(), but allows to specify some options to control how the data are received. We will not consider the possible options in this course. We will assume it equal to 0.

```
#include <sys/socket.h>
ssize_t recv(int sockfd, void *buf, size_t nbytes, int flags);
```

The function returns the length of the message in bytes, 0 if no messages are available and peer had done an orderly shutdown, or -1 on error.

The close() Function

The close() function is used to close a socket and terminate a TCP socket. It returns 0 if it succeeds, -1 on error. It is defined as follows:

```
#include <unistd.h>
int close(int sockfd);
```

Algorithm:

Client

1. Create a socket using the socket() function;
2. Connect the socket to the address of the server using the connect() function;

3. Send and receive data by means of the read() and write() functions.

4. Close the connection by means of the close() function.

Server

1. Create a socket with the socket() function;

2. Bind the socket to an address using the bind() function;

3. Listen for connections with the listen() function;

4. Accept a connection with the accept() function system call. This call typically blocks until a client connects with the server.

5. Send and receive data by means of send() and receive().

6. Close the connection by means of the close() function.

Application:
TCP communication between client and server is used to transfer data reliably.

Conclusion
Hence we studied the socket programming in C and establish a connection between client and server in order to transfer data.

Assignment No: 6

Title: UDP Client Server Communication using Socket Programming of C

Learning Objectives:

- To understand UDP socket programming in C
- To establish a communication between client and server using UDP

Problem Statement:

Write a program using UDP Sockets to enable file transfer (Script, Text, Audio and Video one file each) between two machines. Demonstrate the packets captured traces using Wireshark Packet Analyzer Tool for peer to peer mode

Learning Outcome: Students will able to

- Write client and server code to establish a connection using UDP
- Transfer data and files in between client and server

Software and Hardware Requirement:

- PC with network
- Linux Operating system
- C compiler

Theory:

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.

There are a few steps involved in using sockets:

1. Create the socket
2. Identify the socket (name it)
3. On the server, wait for a message
4. On the client, send a message
5. Send a response back to the client (optional)
6. Close the socket

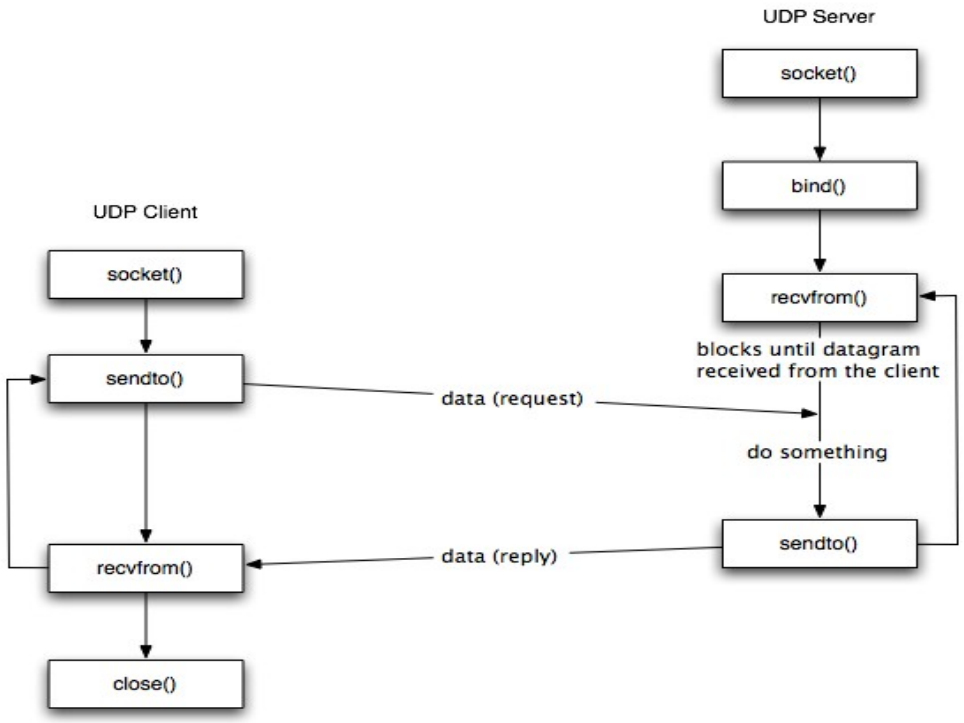


Fig.: UDP Client Server

Other than socket(), bind() , in UDP two new functions recvfrom() and sendto() are used.

The recvfrom() Function

This function is similar to the read() function, but three additional arguments are required. The recvfrom() function is defined as follows:

```
#include <sys/socket.h>
ssize_t recvfrom(int sockfd, void* buff, size_t nbytes, int flags, struct sockaddr* from,socklen_t *addrlen)
```

The first three arguments sockfd, buff, and nbytes, are identical to the first three arguments of read and write. sockfd is the socket descriptor, buff is the pointer to read into, and nbytes is number of bytes to read. In our examples we will set all the values of the flags argument to 0. The recvfrom function fills in the socket address structure pointed to by from with the protocol address of who sent the datagram. The number of bytes stored in the socket address structure is returned in the integer pointed by addrlen.

The function returns the number of bytes read if it succeeds, -1 on error.

The sendto() Function

This function is similar to the send() function, but three additional arguments are required. The sendto() function is defined as follows:

```
#include <sys/socket.h>
ssize_t sendto(int sockfd, const void *buff, size_t nbytes, int flags, const struct sockaddr *to, socklen_t ad
drlen);
```

The first three arguments sockfd, buff, and nbytes, are identical to the first three arguments of recv. sockfd is the socket descriptor, buff is the pointer to write from, and nbytes is number of bytes to write. In our examples we will set all the values of the flags argument to 0. The to argument is a socket address structure containing the protocol address (e.g., IP address and port number) of where the data is sent. addlen specified the size of this socket.

The function returns the number of bytes written if it succeeds, -1 on error.

Algorithm:

Client

- Create a socket using the socket() function;
- Send and receive data by means of the recvfrom() and sendto() functions

Server

- Create a socket with the socket() function;
- Bind the socket to an address using the bind() function;
- Send and receive data by means of recvfrom() and sendto().

Application:
UDP communication between client and server is used to transfer data.

Conclusion
Hence we studied the UDP socket programming in C and establish a UDP connection between client and server in order to transfer data.

Assignment No: 7

Title: Program to analyze file of Wireshark

Learning Objectives:

- Understand the capturing network traffic at Wireshark

Problem Statement:
Write a program to analyze following packet formats captured through Wireshark for wired network. 1. Ethernet 2. IP 3.TCP 4. UDP

Learning Outcome: Students will able to

- Capture network on Wireshark
- Write program to analyze packets on Wireshark file.

Software and Hardware Requirement:

- PC with network
- Linux Operating system
- Wireshark
- C Compiler
- Browser

Theory:

1.Wireshark

Wireshark is a very powerful and popular network analyzer for Windows, Mac and Linux. It's a tool that is used to inspect data passing through a network interface which could be your ethernet, LAN and WiFi.

After downloading and installing Wireshark, you can launch it and double-click the name of a network interface under Capture to start capturing packets on that interface. For example, if you want to capture traffic on your wireless network, click your wireless interface. You can configure advanced features by clicking Capture > Options, but this isn't necessary for now.

2. Viewing and Analyzing Packet Contents

After capturing data , take a look at the captured packets. As shown in the screenshot above, the captured data interface contains three main sections: The packet list pane, the packet details pane, and the packet bytes pane.

Packet List

The packet list pane, located at the top of the window, shows all packets found in the active capture file. Each packet has its own row and corresponding number assigned to it, along with each of these data points.

- **Time:** The timestamp of when the packet was captured is displayed in this column, with the default format being the number of seconds (or partial seconds) since this specific capture file was first created. To modify this format to something that may be a bit more useful, such as the actual time of day, select the *Time Display Format* option from Wireshark's *View* menu - located at the top of the main interface.

- **Source:** This column contains the address (IP or other) where the packet originated.

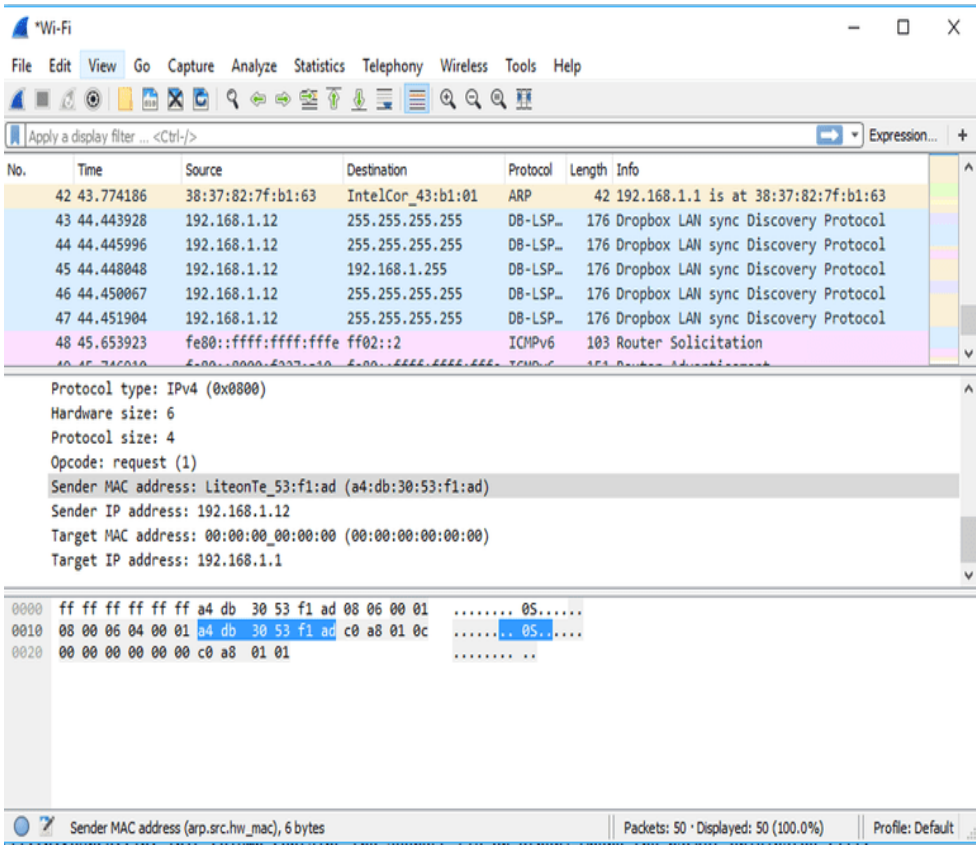
- **Destination:** This column contains the address that the packet is being sent to.

- **Protocol:** The packet's protocol name (i.e., TCP) can be found in this column.

- **Length:** The packet length, in bytes, is displayed in this column.

- **Info:** Additional details about the packet are presented here. The contents of this column can vary greatly depending on packet contents.

When a packet is selected in the top pane, you may notice one or more symbols appear in the first column. Open and/or closed brackets, as well as a straight horizontal line, can indicate whether or not a packet or group of packets are all part of the same back-and-forth conversation on the network. A broken horizontal line signifies that a packet is not part of said conversation.



Packet Details

The details pane, found in the middle, presents the protocols and protocol fields of the selected packet in a collapsible format. In addition to expanding each selection, you can also apply individual filters based on specific details as well as follow streams of data based on protocol type via the details context menu -- accessible by right-clicking your mouse on the desired item within this pane.

Packet Bytes

At the bottom is the packet bytes pane, which displays the raw data of the selected packet in a hexadecimal view. This hex dump contains 16 hexadecimal bytes and 16 ASCII bytes alongside the data offset.

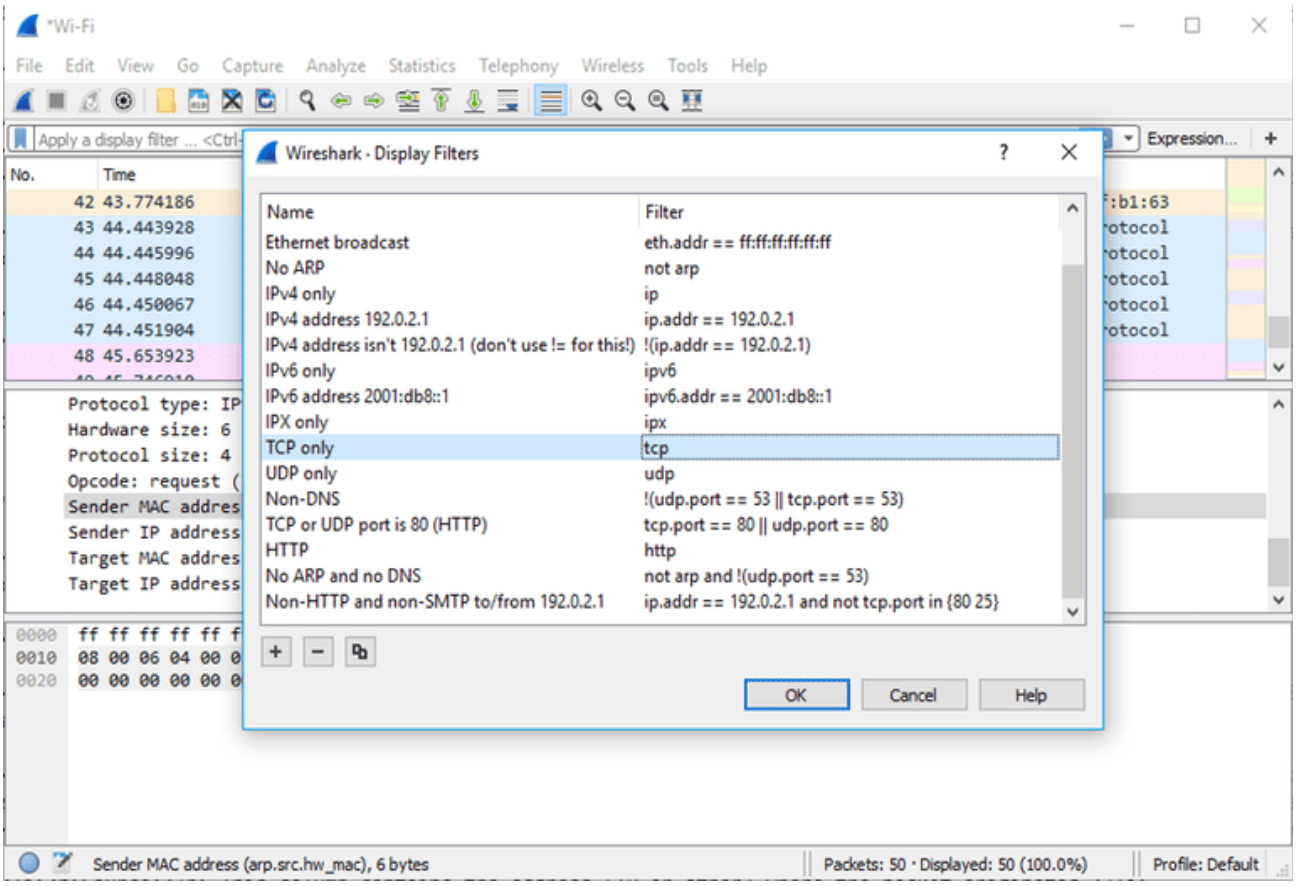
Selecting a specific portion of this data automatically highlights its corresponding section in the packet details pane, and vice versa. Any bytes that cannot be printed are instead represented by a period.

You can choose to show this data in bit format as opposed to hexadecimal by right-clicking anywhere within the pane and selecting the appropriate option from the context menu.

3. Using Filters

One of the most important feature sets in Wireshark is its filter capabilities, especially when you're dealing with files that are significant in size. Capture filters can be set before the fact, instructing Wireshark to only record those packets that meet your specified criteria.

Filters can also be applied to a capture file that has already been created so that only certain packets are shown. These are referred to as display filters.



Wireshark provides a large number of predefined filters by default, letting you narrow down the number of visible packets with just a few keystrokes or mouse clicks. To use one of these existing filters, place its name in the *Apply a display filter* entry field (located directly below the Wireshark toolbar) or in the *Enter a capture filter* entry field (located in the center of the welcome screen).

There are multiple ways to achieve this. If you already know the name of your filter, simply type it into the appropriate field. For example, if you only wanted to display TCP packets you would type *tcp*. Wireshark's autocomplete feature will show suggested names as you begin typing, making it easier to find the correct moniker for the filter you're seeking.

Once set, capture filters will be applied as soon as you begin recording network traffic. To apply a display filter, however, you'll need to click on the right arrow button found on the far-right hand side of the entry field.

Algorithm:

1. Open Wireshark
2. Capture some network traffic by browsing some data on browser.
3. Save a file of network traffic.

4. Use this file in code to count number of ICMP,TCP call made by system.

5. Print the count for different protocols.

Application:
With help of Wireshark we can analyze the network traffic, Packet Structure and can apply filtering. Wireshark tool can be use to troubleshoot network related issues.

Conclusion
Hence we studied the Wireshark packet analyzer tool and analyze the different protocols used during client server communication.

Assignment No: 8

Title: DNS Lookup

Learning Objectives:
- To understand the Process of DNS.

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

Learning Outcome: Students will able to

- Describe process of DNS.
- Get domain name for IP address and vice versa

Software and Hardware Requirement:

- PC with network
- Linux Operating system
- Netbeans/Eclipse
-

Theory:

1. DNS:

A DNS server is a computer server that contains a database of public IP addresses and their associated hostnames, and in most cases, serves to resolve, or translate, those common names to IP addresses as requested.DNS servers run special software and communicate with each other using special protocols.

In more easy to understand terms: a DNS server on the internet is the device that translates that *www.domainname.com* to its corresponding IP address e.g. *151.101.129.121* .The DNS domain namespace, as shown in the following figure, is based on the concept of a tree of named domains. Each level of the tree can represent either a branch or a leaf of the tree. A branch is a level where more than one name is used to identify a collection of named resources. A leaf represents a single name used once at that level to indicate a specific resource.

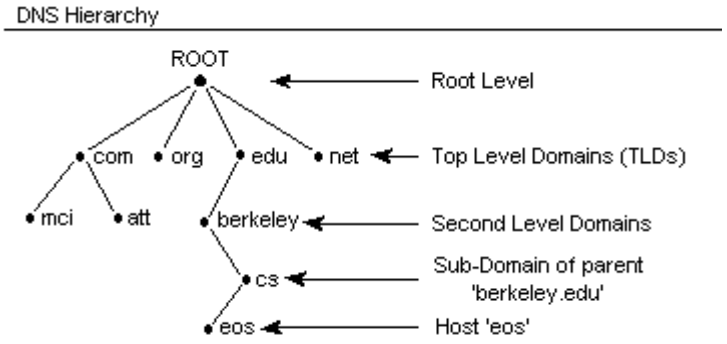


Fig: DNS Hierarchy

2. Why Do We Have DNS Servers?

When you enter *www.google.com* into a web browser, all you have to understand and remember is the URL *http://www.google.com*. The same is true for any other website like *yahoo.com*, *Amazon.com*, etc.The opposite is true too, that while we as humans can understand the words in the URL much easier than the IP address numbers, other computers and network devices understand the IP address.

Therefore, we have DNS servers because we not only want to use human-readable names to access websites, but the computers need to use IP addresses to access websites. The DNS server is that translator between the hostname and IP address.

3. DNS LOOKUP

A DNS lookup, in a general sense, is the process by which a DNS record is returned from a DNS server. This is like looking up a phone number in a phone book - that is why it is referred to as a "lookup". Interconnected computers, servers and smart phones need to know how to translate the email addresses and domain names people use into meaningful numerical addresses. A DNS lookup performs this function

The basic idea of DNS is that humans can't easily remember long strings of digits like machines can, but can much more easily remember words. So, when you type in a domain name like www.techopedia.com, the request is forwarded to a DNS server (whether locally or at an ISP), which returns the corresponding IP address. This address is then used by all the computers and routers to channel the request and responses of a user's session. The result is the user sees web pages as expected or has email show up in an in-box. The two types of DNS lookups are forward DNS lookups and reverse DNS lookups.

Forward DNS lookup is using an Internet domain name to find an IP address. Reverse DNS lookup is using an Internet IP address to find a domain name. When you enter the address for a Web site at your browser (the address is formally called the Uniform Resource Locator, or URL), the address is transmitted to a nearby router which does a forward DNS lookup in a routing table to locate the IP address. Forward DNS (which stands for domain name system) lookup is the more common lookup since most users think in terms of domain names rather than IP addresses. However, occasionally you may see a Web page with a URL in which the domain name part is expressed as an IP address (sometimes called a dot address) and want to be able to see its domain name. An Internet facility that lets you do either forward or reverse DNS lookup yourself is called nslookup. It comes with some operating systems or you can download the program and install it in your computer.

Algorithm:

1. Enter a domain Name
2. Using Java /Python API extract IP Address
3. Return IP address corresponding to domain name

   Or

1. Enter IP address
2. Using Java /Python API extract IP Address host name
3. Return domain name for corresponding IP address

Application:
The computers need to use IP addresses to access websites. The DNS server is that translator between the hostname and IP address

Conclusion
Hence we studied the process of DNS lookup.

Assignment No: 9

Title: Install and Configure DHCP server

Learning Objectives:

- Install and configure DHCP server on Ubuntu
- Write program to install software on remote machine

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

Learning Outcome: Students will able to

- Install and configure DHCP server on Ubuntu

Software and Hardware Requirement:

- PC with network
- Linux Operating system

Theory:

1. DHCP:
The Dynamic Host Configuration Protocol (DHCP) is a network service that enables host computers to be automatically assigned settings from a server as opposed to manually configuring each network host. Computers configured to be DHCP clients have no control over the settings they receive from the DHCP server, and the configuration is transparent to the computer's user.

The most common settings provided by a DHCP server to DHCP clients include:

- IP address and netmask
- IP address of the default-gateway to use
- IP adresses of the DNS servers to use

However, a DHCP server can also supply configuration properties such as:

- Host Name
- Domain Name
- Time Server
- Print Server

The advantage of using DHCP is that changes to the network, for example a change in the address of the DNS server, need only be changed at the DHCP server, and all network hosts will be reconfigured the next time their DHCP clients poll the DHCP server. As an added advantage, it is also easier to integrate new computers into the network, as there is no need to check for the availability of an IP address. Conflicts in IP address allocation are also reduced.

A DHCP server can provide configuration settings using the following methods:

Manual allocation (MAC address)

This method entails using DHCP to identify the unique hardware address of each network card connected to the network and then continually supplying a constant configuration each time the DHCP client makes a request to the DHCP server using that network device. This ensures that a particular address is assigned automatically to that network card, based on it's MAC address.

Dynamic allocation (address pool)

In this method, the DHCP server will assign an IP address from a pool of addresses (sometimes also called a range or scope) for a period of time or lease, that is configured on the server or until the client informs the server that it doesn't need the address anymore. This way, the clients will be receiving their configuration properties dynamically and on a "first come, first served" basis. When a DHCP client is no longer on the network for a specified period, the configuration is expired and released back to the address pool for use by other DHCP Clients. This way, an address can be leased or used for a period of time. After this period, the client has to renegociate the lease with the server to maintain use of the address.

Automatic allocation

Using this method, the DHCP automatically assigns an IP address permanently to a device, selecting it from a pool of available addresses. Usually DHCP is used to assign a temporary address to a client, but a DHCP server can allow an infinite lease time.

The last two methods can be considered "automatic" because in each case the DHCP server assigns an address with no extra intervention needed. The only difference between them is in how long the IP address is leased, in other words whether a client's address varies over time. Ubuntu is shipped with both DHCP server and client. The server is dhcpd (dynamic host configuration protocol daemon). The client provided with Ubuntu is dhclient and should be installed on all computers required to be automatically configured. Both programs are easy to install and configure and will be automatically started at system boot.

Advantages of DHCP:
DHCP has a number of advantages:
• There is no need to manually configure each client with an IP address.
• You don't need to keep a record of the IP addresses that you have assigned.
• You can automatically assign a new IP address if you move a client to a different subnet.
• You can release the IP address of a computer that is offline and reassign the address to another computer.
• Address duplication is eliminated as DHCP automatically tracks IP address assignments.
• The DHCP server can detect unauthorised DHCP servers on the network.

Algorithm:

Installation and configuration

At a terminal prompt, enter the following command to install dhcp:

Step1: sudo apt install isc-dhcp-server

Step2: gedit /etc/dhcp/dhcpd.conf and make change in this file as given below:

```
# minimal sample /etc/dhcp/dhcpd.conf
default-lease-time 600;
max-lease-time 7200;

subnet 192.168.1.0 netmask 255.255.255.0 {
 range 192.168.1.150   192.168.1.200;
 option routers 192.168.1.254;
```

```
 option domain-name-servers 192.168.1.1, 192.168.1.2;
 option domain-name "mydomain.example";
}
```

This will result in the DHCP server giving clients an IP address from the range 192.168.1.150-192.168.1.200. It will lease an IP address for 600 seconds if the client doesn't ask for a specific time frame. Otherwise the maximum (allowed) lease will be 7200 seconds. The server will also "advise" the client to use 192.168.1.254 as the default-gateway and 192.168.1.1 and 192.168.1.2 as its DNS servers.

After changing the config file you have to restart the dhcpd:

Step3: sudo service isc-dhcp-server restart

Install software on other machine

1.Use scp command to copy any ".deb" file from your computer to remote computer.

```
scp SourceFile user@host:directory/TargetFile
Example: scp abc.txt comp1@172.20.54.29:/home/comp1/
```

2. Log into remote computer by using ssh. You will need remote computer's password.

```
 ssh [-l login_name ] hostname | user@hostname [command ]
Examples: ssh ccompl01-19@172.20.54.29
```

3.Your will get remote computer's shell. Now you can execute commands which will install package on remote computer.

```
example :sudo dpkg -i abc.deb
```

The code for above mentioned step in Python as follows:

```
 #! /user/bin
import os
print "\n Copying .deb file from our computer to remote computer \n"
os.system("scp libqt5gui5_5.2.1+dfsg-1ubuntu14.2_amd64.deb comp1@172.20.54.22:/home/comp1/ ")
print "\n Copy completed \n"
os.system("ssh -t comp1@172.20.54.22 sudo dpkg -i libqt5gui5_5.2.1+dfsg-1ubuntu14.2_amd64.deb \n")
print "\n ********** Installation completed..!! **********\n"
```

Application:
DHCP enables host computers to be automatically assigned settings from a server as opposed to manually configuring each network host

Conclusion

Hence we have studied the DHCP protocol concepts and we have installed and configured DHCP server.

Assignment No: 10

Title: TCP Client Server Communication using Socket Programming of Java

Learning Objectives:

- To understand TCP socket programming in Java
- To establish a communication between client and server using TCP

Problem Statement:
Write a program using TCP sockets for wired network to implement
a. Peer to Peer Chat
b. Multiuser Chat
Demonstrate the packets captured traces using Wireshark Packet Analyzer Tool for peer to peer mode

Learning Outcome: Students will able to
- Write client and server code to establish a connection using TCP Socket Programming of Java
- Transfer data and files in between client and server

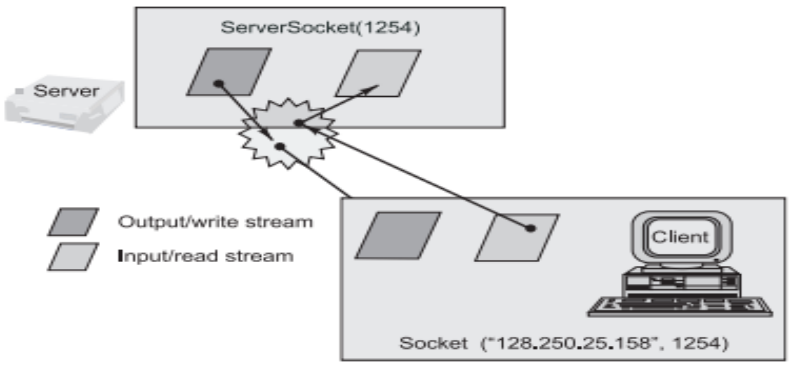
Software and Hardware Requirement:
- PC with network
- Linux Operating system
  Eclipse/Netbeans

Theory:

1.Socket Programming in java

The two key classes from the java.net package used in creation of server and client programs are:
- ServerSocket
- Socket

A server program creates a specific type of socket that is used to listen for client requests (server socket), In the case of a connection request, the program creates a new socket through which it will exchange data with the client using input and output streams. The socket abstraction is very similar to the file concept: developers have to open a socket, perform I/O, and close it. Figure illustrates key steps involved in creating socket-based server and client programs.



In order to connect to a server over the internet (via TCP/IP) in Java, you need to create a java.net.Socket and connect it to the server.

The Socket class has five constructors that a client uses to connect to a server:

SN	Methods	Description

1	public Socket(String host, int port) throws UnknownHostException, IOException.	This method attempts to connect to the specified server at the specified port. If this constructor does not throw an exception, the connection is successful and the client is connected to the server.
2	public Socket(InetAddress host, int port) throws IOException	This method is identical to the previous constructor, except that the host is denoted by an InetAddress object.
3	public Socket(String host, int port, InetAddress localAddress, int localPort) throws IOException.	Connects to the specified host and port, creating a socket on the local host at the specified address and port.
4	public Socket(InetAddress host, int port, InetAddress localAddress, int localPort) throws IOException.	This method is identical to the previous constructor, except that the host is denoted by an InetAddress object instead of a String
5	public Socket()	Creates an unconnected socket. Use the connect() method to connect this socket to a server.

Here are some of the common methods of the Socket class:

SN	**Methods**	**Description**
1	public void connect(SocketAddress host, int timeout) throws IOException	This method connects the socket to the specified host. This method is needed only when you instantiated the Socket using the no-argument constructor.
2	public InetAddress getInetAddress()	This method returns the address of the other computer that this socket is connected to.
3	public int getPort()	Returns the port the socket is bound to on the remote machine.
4	public int getLocalPort()	Returns the port the socket is bound to on the local machine.
5	public SocketAddress getRemoteSocketAddress()	Returns the address of the remote socket.
6	public InputStream getInputStream() throws IOException	Returns the input stream of the socket. The input stream is connected to the output stream of the remote socket.
7	public OutputStream getOutputStream() throws IOException	Returns the output stream of the socket. The output stream is connected to the input stream of the remote socket
8	public void close() throws IOException	Closes the socket, which makes this Socket object no longer capable of connecting again to any server

ServerSocket Class

In order to implement a Java server that listens for incoming connections from clients via TCP/IP, you need to use a `java.net.ServerSocket` .

The ServerSocket class has four constructors:

SN	Methods	Description
1	public ServerSocket(int port) throws IOException	Attempts to create a server socket bound to the specified port. An exception occurs if the port is already bound by another application.
2	public ServerSocket(int port, int backlog) throws IOException	Similar to the previous constructor, the backlog parameter specifies how many incoming clients to store in a wait queue.
3	public ServerSocket(int port, int backlog, InetAddress address) throws IOException	Similar to the previous constructor, the InetAddress parameter specifies the local IP address to bind to. The InetAddress is used for servers that may have multiple IP addresses, allowing the server to specify which of its IP addresses to accept client requests on
4	public ServerSocket() throws IOException	Creates an unbound server socket. When using this constructor, use the bind() method when you are ready to bind the server socket

Here are some of the common methods of the ServerSocket class:

SN	Methods	Description
1	public int getLocalPort()	Returns the port that the server socket is listening on. This method is useful if you passed in 0 as the port number in a constructor and let the server find a port for you.
2	public Socket accept() throws IOException	Waits for an incoming client. This method blocks until either a client connects to the server on the specified port or the socket times out, assuming that the time-out value has been set using the setSoTimeout() method. Otherwise, this method blocks indefinitely
3	public void setSoTimeout(int timeout)	Sets the time-out value for how long the server socket waits for a client during the accept().
4	public void bind(SocketAddress host, int	Binds the socket to the specified server and port

	backlog)	in the SocketAddress object. Use this method if you instantiated the ServerSocket using the no-argument constructor.

Algorithm:

Server

1. Open the Server Socket:

   ServerSocket server = new ServerSocket(PORT);

2. Wait for the Client Request:

   Socket client = server.accept();

3. Create I/O streams for communicating to the client

   DataInputStream is = new DataInputStream(client.getInputStream());

   DataOutputStream os = new DataOutputStream(client.getOutputStream());

4. Perform communication with client Receive from client:

   String line = is.readLine(); Send to client: os.writeBytes("Hello\n");

5. Close socket:

   client.close();

Client

1. Create a Socket Object:

   Socket client = new Socket(server, port_id);

2. Create I/O streams for communicating with the server.

   is = new DataInputStream(client.getInputStream());

   os = new DataOutputStream(client.getOutputStream());

3. Perform I/O or communication with the server: Receive data from the server:

   String line = is.readLine();

4. Send data to the server:

os.writeBytes(“Hello\n”);

5. Close the socket when done:

client.close();

Application:

TCP client server communication is essential in order to make a interactive application over the network/

Conclusion
Hence we studied socket programming in java to establish a communication between client and server using TCP.

Assignment No: 11

Title: UDP Client Server Communication using Socket Programming of Java
Learning Objectives:

- To understand UDP socket programming in Java
- To establish a communication between client and server using UDP

Problem Statement:
Write a program using UDP sockets for wired network to implement
a. Peer to Peer Chat
b. Multiuser Chat
Demonstrate the packets captured traces using Wireshark Packet Analyzer Tool for peer to peer mode

Learning Outcome: Students will able to

- Write client and server code to establish a connection using UDP
- Transfer data and files in between client and server

Software and Hardware Requirement:

- PC with network
- Linux Operating system
- Eclipse/Netbeans

Theory:

Sometimes in Socket Programming rather than reliable delivery. It is required to use a lighter transport protocol. This kind of service is accomplished by the UDP protocol which conveys datagram packet s. Datagram packets are used to implement a connectionless packet delivery service supported by the UDP protocol. Each message is transferred from source machine to destination based on information contained within that packet. That means, each packet needs to have destination address and each packet might be routed differently, and might arrive in any order. Packet delivery is not guaranteed. The format of datagram packet is:

Msg | length | Host | serverPort

Java supports datagram communication through the following classes:

- DatagramPacket
- DatagramSocket: Datagram Socket is used to send or receive Datagrams .

Constructors of DatagramPacket

DatagramPacket (byte[ ] buff , int len)	Constructs a DatagramPacket for receiving packets of length len.
DatagramPacket (byte[] buf, int off, int len)	Constructs a DatagramPacket for receiving packets of length len, specifying an offset of off bytes into the buffer.
DatagramPacket ((byte[] buf, int len, InetAddress addr, int port)	Constructs a datagram packet for sending packets of length len to the specified port number on the specified host.
DatagramPacker (byte[] buf, int off, int len, InetAddress addr, int port)	Constructs a datagram packet for sending packets of length len with offset off to the specified port number on the specified host.
DatagramPacket (byte[] buf, int off, int len, SocketAddress addr)	Constructs a datagram packet for sending packets of length len with offset off to the specified port number on the specified host.

Constructors of DatagramSocket

DatagramSocket ()	Constructs a datagram socket and binds it to any available port on the local host.
DatagramSocket (DatagramSocketImpl impl)	Creates an unbound datagram socket with the specified DatagramSocketImpl.
DatagramSocket (int port)	Constructs a datagram socket and binds it to the specified port on the local host.
DatagramSocket (int port, InetAddress iaddr)	Creates a datagram socket, bound to the specified local address.
DatagramSocket (SocketAddress bindaddr)	Creates a datagram socket, bound to the specified local socket address.

Algorithm:

Server

1. Open the Datagram Socket object.
2. Wait for the Client Request.
3. Create Datagram Packet for communicating to the client.
4. Perform communication with client: Receive from client.
5. Send some data if required
6. Close socket.

Client

1. Create a Datagram Socket object.
2. Create Datagram Packet for communicating with the server.
3. Perform I/O or communication with the server: Send data to the server:
4. Receive some data from server if required
5. Close the socket when done:

Application:
UDP client server communication is essential in order to make an interactive application over the network.

Conclusion
Hence we studied socket programming in java to establish a communication between client and server using UDP.

Assignment No: 12

Title: Simulation of network traffic using Network Simulator 2

Learning Objectives:

- To understand the purpose of NS2
- Write simple code in NS2
- Analyze the network over NS2

Problem Statement:
Use network simulator NS2 to implement:
a. Monitoring traffic for the given topology
b. Analysis of CSMA and Ethernet protocols
c. Network Routing: Shortest path routing, AODV.
d. Analysis of congestion control (TCP and UDP).

Learning Outcome: Students will able to

- Describe the purpose of NS2
- Write simple code in NS2
- Analyze the network traffic over NS2

Software and Hardware Requirement:

- PC with network
- Linux Operating system
- NS2

Theory:

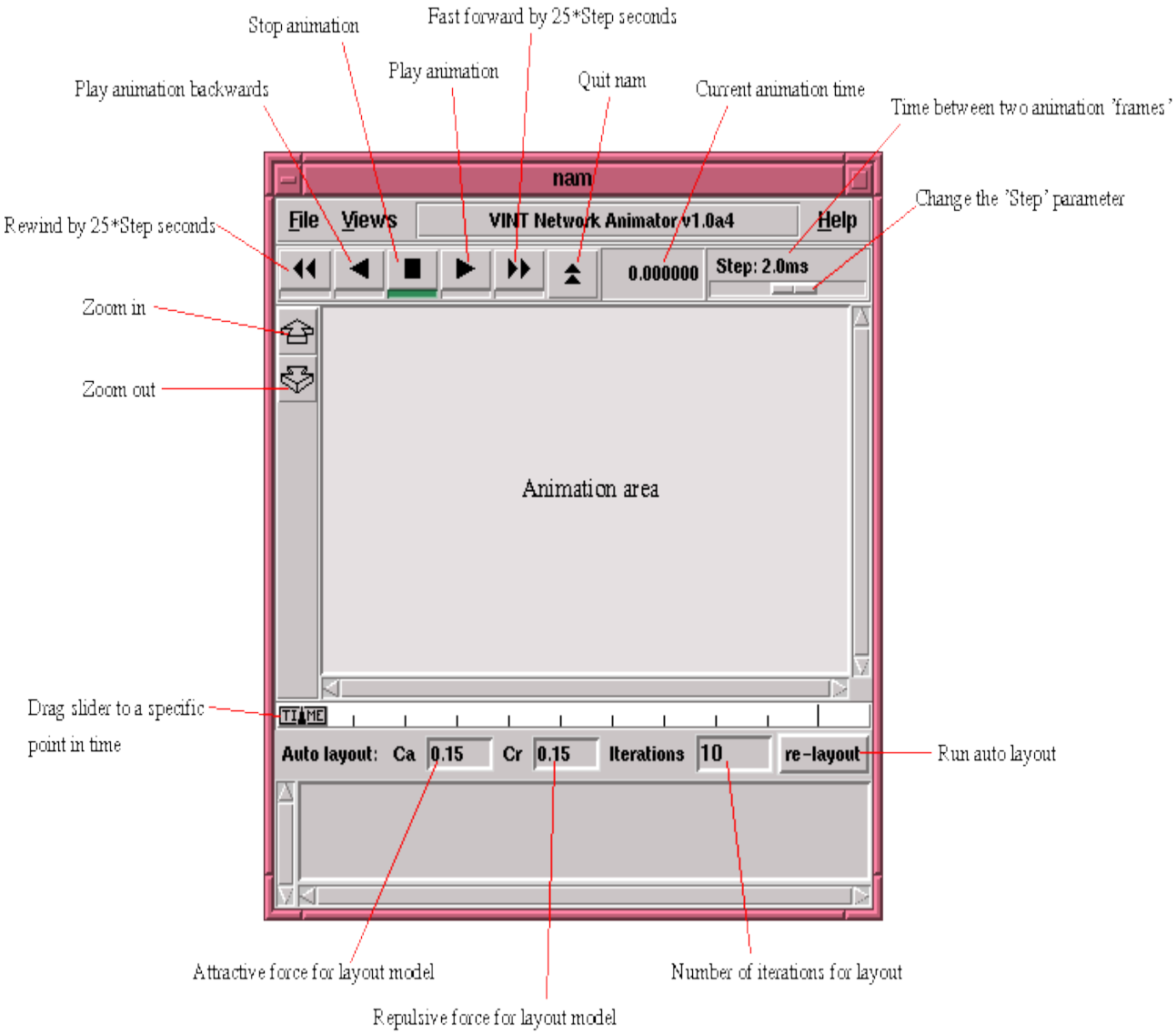
1. NS2:

Network simulator is a package of tools that simulates behavior of networks such as creating network topologies, log events that happen under any load,analyze the events and understand the network.
Network Simulator is mainly based on two languages.They are C++ and OTcl. OTcl is the object oriented version of Tool Command language.The network simulator is a bank of different network and protocol objects. C++ helps in the following way:

- It helps to increase the efficiency of simulation.
- Its used to provide details of the protocols and their operation.
- It is used to reduce packet and event processing time.
- OTcl helps in the following way:
    - With the help of OTcl we can describe different network topologies
    - It helps us to specify the protocols and their applications
    - It allows fast development
    - Tcl is compatible with many platforms and it is flexible for integration
    - Tcl is very easy to use and it is available in free

2.Starting ns
You start ns with the command 'ns <tclscript>' (assuming that you are in the directory with the ns executable, or that your path points to that directory), where '<tclscript>' is the name of a Tcl script file which defines the simulation scenario (i.e. the topology and the events).

3.Starting nam
You can either start nam with the command 'nam <nam-file>' where '<nam-file>' is the name of a nam trace file that was generated by ns, or you can execute it directly .



Writing first TCL file:

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

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

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

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

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

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

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

}
```

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

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

You probably understand what this line does just by looking at it. ns provides you with a very simple way to schedule events with the 'at' command.

The last line finally starts the simulation.

```
$ns run
```

Two nodes, one link
In this section we are going to define a very simple topology with two nodes that are connected by a link. The following two lines define the two nodes. (Note: You have to insert the code in this section **before** the line '$ns run', or even better, before the line '$ns at 5.0 "finish"').

```
set n0 [$ns node]

set n1 [$ns node]
```

A new node object is created with the command '$ns node'. The above code creates two nodes and assigns them to the handles 'n0' and 'n1'.

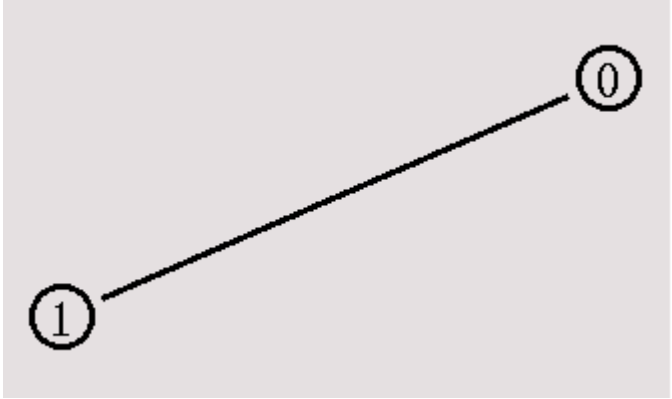
The next line connects the two nodes.

```
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
```

Drop Tai is a simple queue mechanism that is used by the routers that when packets should to be drop. In this mechanism each packet is treated identically and when queue filled to its maximum capacity the newly incoming packets are dropped until queue have sufficient space to accept incoming traffic.

This line tells the simulator object to connect the nodes n0 and n1 with a duplex link with the bandwidth 1Megabit, a delay of 10ms and a DropTail queue.

Now you can save your file and start the script with 'ns example1.tcl'. nam will be started automatically and you should see an output that resembles the picture below.



Sending data

Now we will show how to set up tcp connection between two nodes

```
1.  set tcp [new Agent/TCP]
2. $ns attach-agent $n0 $tcp
3. set sink [new Agent/TCPSink]
4.  $ns attach-agent $n4 $sink
5. $ns connect $tcp $sink
6.  $tcp set fid_1
7. $tcp set packetSize_552
```

The command 'set tcp [new Agent/TCP]' gives a pointer called 'tcp' which indicates the tcp agent which is a object of ns. Then the command '$ns attach-agent $n0 $tcp' defines the source node of tcp connection. Next the command 'set sink [new Agent/TCPSink]' defines the destination of tcp by a pointer called sink. The next command '$ns attach-agent $n4 $sink' defines the destination node as n4.Next, the command '$ns connect $tcp $sink' makes the TCP connection between the source and the destination.i.e n0 and n4.When we have several flows such as TCP, UDP etc in a network. So, to identify these flows we mark these flows by using the command '$tcp set fid_1'. In the last line we set the packet size of tcp as 552 while the default packet size of tcp is 1000.

Now we will learn how to create a UDP connection in network simulator.

```
1.  set udp [new Agent/UDP]
2. $ns attach-agent $n1 $udp
3.  $set null [new Agent/Null]
4.  $ns attach-agent $n5 $null
5.  $ns connect $udp $null
6.  $udp set fid_2
```

Similarly,the command 'set udp [new Agent/UDP]' gives a pointer called 'udp' which indicates the udp agent which is a object of ns.Then the command '$ns attach-agent $n1 $udp' defines the source node of udp connection. Next the command 'set null [new Agent/Null]' defines the destination of udp by a pointer called null. The next command '$ns attach-agent $n5 $null' defines the destination node as n5.Next, the command '$ns connect $udp $null' makes the UDP connection between the source and the destination.i.e n1 and n5.When we have several flows such as TCP,UDP etc in a network. So, to identify these flows we mark these flows by using the command '$udp set fid_2

CBR over UDP Connection

Constant Bit Rate (CBR) is a term used in telecommunications, relating to the quality of service.When referring to codecs, constant bit rate encoding means that the rate at which a codec's output data should be consumed is constant. CBR is useful for streaming multimedia content on limited capacity channels since it is the maximum bit rate that matters, not the average, so CBR would be used to take advantage of all of the capacity.

1. set cbr [new Application/Traffic/CBR]
2. $cbr attach-agent $udp
3. $cbr set packetSize_1000
4. $cbr set rate_0.01Mb
5. $cbr set random _false

In the above we define a CBR connection over a UDP one. Well we have already defined the UDP source and UDP agent as same as TCP. Instead of defining the rate we define the time interval between the transmission of packets in the command '$cbr set rate_0.01Mb'. Next, with the help of the command '$cbr set random _false' we can set random noise in cbr traffic.we can keep the noise by setting it to 'false' or we can set the noise on by the command '$cbr set random _1'. We can set by packet size by using the command '$cbr set packetSize_(packetsize).We can set the packet size up to sum value in bytes.

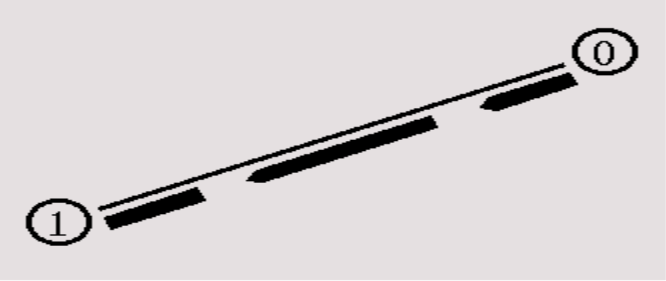
And now we have to tell the CBR agent when to send data and when to stop sending. Note: It's probably best to put the following lines just before the line '$ns at 5.0 "finish"'.

```
$ns at 0.5 "$cbr0 start"

$ns at 4.5 "$cbr0 stop"
```

This code should be self-explaining again.

Now you can save the file and start the simulation again. When you click on the 'play' button in the nam window, you will see that after 0.5 simulation seconds, node 0 starts sending data packets to node 1. You might want to slow nam down then with the 'Step' slider.



Analyzing trace file

In ns2 simulation result stored in trace files. These file formats need to be understood for analysing result. Here we will discuss general trace file and nam trace file.

Here we will discuss format of trace file.

event	time	from node	to node	pkt type	pkt size	flags	flow_id	src addr	Dst addr	seq num	pkt id

1. The first field is event.It gives you four possible symbols '+' '-' 'r' 'd'.These four symbols correspond respectively to enqueued, dequeued, received and dropped.
2. The second field gives the time at which the event occurs
3. The third field gives you the input node of the link at which the event occurs
4. The fourth field gives you the the output node at which the event occurs
5. The fifth field shows the information about the packet type.i.e whether the packet is UDP or TCP
6. The sixth field gives the packet size
7. The seventh field give information about some flags
8. The eight field is the flow id(fid) for IPv6 that a user can set for each flow in a tcl script.It is also used for specifying the color of flow in NAM display
9. The ninth field is the source address
10. The tenth field is the destination address
11. The eleventh field is the network layer protocol's packet sequence number
12. The last field shows the unique id of packet

Following are trace of two events:

r 1.84471 2 1 cbr 210 ------- 1 3.0 1.0 195 600
r 1.84566 2 0 ack 40 ------- 2 3.2 0.1 82 602

Algorithm:

1. Create a simulator object
2. Open a file for writing the trace
3. Add a 'finish' procedure that closes the trace file and starts nam
4. Add nodes and links to create star topology
5. Transmit data either by creating TCP connection/ UDP connection
6. Run the simulation and observe the network traffic/congestion etc
7. Analyze the trace file

Application:

NS2 simulates behavior of networks such as creating network topologies, log events that happen under any load, analyze the events and understand the network.

Conclusion
Hence we learn how to use NS2 tool and simulate the network over it.

Assignment No: 13

Title: Configure RIP/OSPF/BGP using Packet Tracer

Learning Objectives:
- To learn Packet Tracer tool
- To configure RIP/OSPF/BGP on Packet Tracer

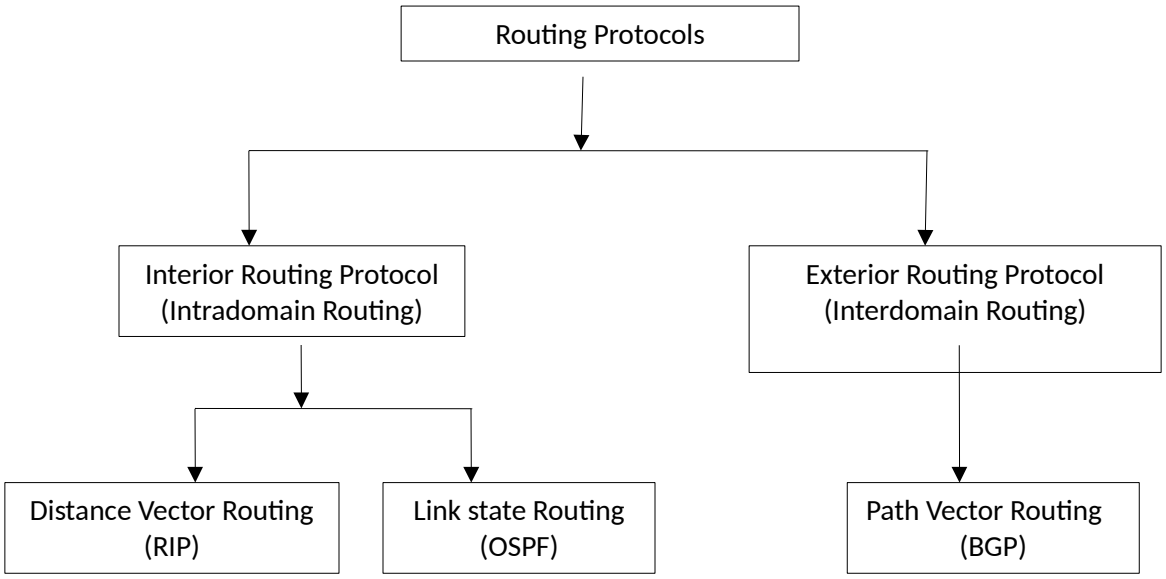
Problem Statement:
Configure RIP/OSPF/BGP using Packet Tracer

Learning Outcome: Students will able to
- To learn Packet Tracer tool
- To configure RIP/OSPF/BGP on Packet Tracer

Software and Hardware Requirement:
- PC with network
- Linux Operating system
- Packet Tracer

Theory:

1. Routing Protocols:
A routing protocol specifies how routers communicate with each other, distributing information that enables them to select routes between any two nodes on a computer network. Routing algorithms determine the specific choice of route. Each router has a priori knowledge only of networks attached to it directly. A routing protocol shares this information first among immediate neighbors, and then throughout the network. This way, routers gain knowledge of the topology of the network.Following fig. shows the types of routing Protocols.



Distance vector protocols

As the name implies, distance vector routing protocols use distance to determine the best path to a remote network. The distance is usually the number of hops (routers) to the destination network.

Distance vector protocols send complete routing table to each neighbor (a neighbor is directly connected router that runs the same routing protocol). They usually use some version of Bellman-Ford algorithm to calculate the best routes. Compared with link state routing protocols, distance vector protocols are simpler to configure and require little management, but are susceptible to routing loops and converge slower than link state routing protocols. Distance vector protocols also use more bandwidth because they send complete routing table, while link state procotols sends specific updates only when topology changes occur.RIP and EIGRP are examples of distance vector routing protocols.

Link state protocols

Unlike distance vector protocols, link state protocols don't advertise the entire routing table. Instead, they advertise information about a network toplogy (directly connected links, neighboring routers…), so that in the end all routers running a link state protocol have the same topology database. Link state routing protocols converge much faster than distance vector routing protocols, support classless routing, send updates using multicast addresses and use triggered routing updates. They also require more router CPU and memory usage than distance-vector routing protocols and can be harder to configure.

Each router running a link state routing protocol creates three different tables:

1. neighbor table – the table of neighboring routers running the same link state routing protocol
2. topology table – the table that stores the topology of the entire network
3. routing table – the table that stores the best routes

Shortest Path First algorithm is used to calculate the best route. OSPF and IS-IS are examples of link state routing protocols.

Path vector protocol

A path vector protocol is a network routing protocol which maintains the path information that gets updated dynamically. Updates which have looped through the network and returned to the same node are easily detected and discarded. This algorithm is sometimes used in Bellman–Ford routing algorithms to avoid "Count to Infinity" problems.

It is different from the distance vector routing and link state routing. Each entry in the routing table contains the destination network, the next router and the path to reach the destination.

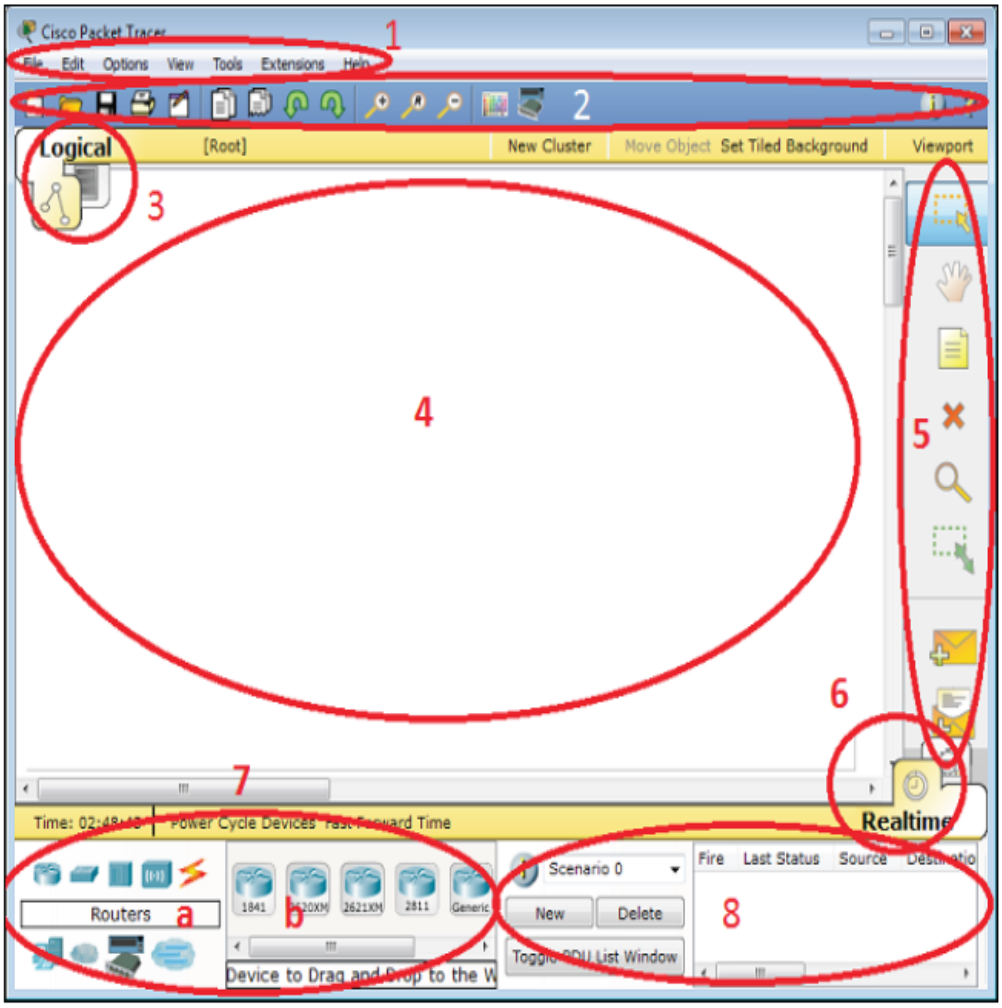
Path Vector Messages in Border Gateway Protocol (BGP): The autonomous system boundary routers (ASBR), which participate in path vector routing, advertise the reachability of networks. Each router that receives a path vector message must verify that the advertised path is according to its policy. If the messages comply with the policy, the ASBR modifies its routing table and the message before sending it to the next neighbor. In the modified message it sends its own AS number and replaces the next router entry with its own identification.

Border Gateway Protocol is an example of a path vector protocol. In BGP, the routing table maintains the autonomous systems that are traversed in order to reach the destination system.

2. Packet Tracer:

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

The layout of Packet Tracer is divided into several components similar to a photo editor. Match the numbering in the following screenshot with the explanations given after it:



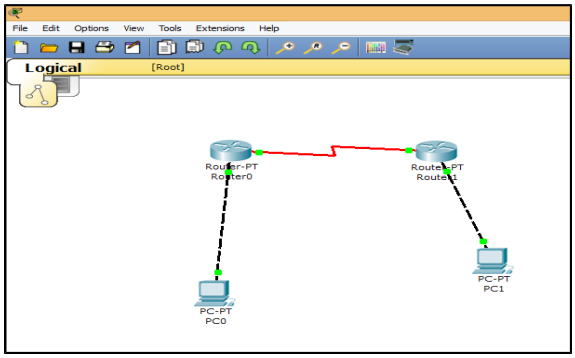
The components of the Packet Tracer interface are as follows:

1. Menu bar – This is a common menu found in all software applications; it is used to open, save, print, change preferences, and so on.
2. Main toolbar – This bar provides shortcut icons to menu options that are commonly accessed, such as open, save, zoom, undo, and redo, and on the right-hand side is an icon for entering network information for the current network.
3. Logical/Physical workspace tabs – These tabs allow you to toggle between the Logical and Physical work areas.
4. Workspace – This is the area where topologies are created and simulations are displayed.
5. Common tools bar – This toolbar provides controls for manipulating topologies, such as select, move layout, place note, delete, inspect, resize shape, and add simple/complex PDU.
6. Realtime/Simulation tabs – These tabs are used to toggle between the real and simulation modes. Buttons are also provided to control the time, and to capture the packets.
7. Network component box – This component contains all of the network and end devices available with Packet Tracer, and is further divided into two areas:
   7a: Device-type selection box – This area contains device categories
   7b: Device-specific selection box – When a device category is selected, this selection box displays the different device models within that category

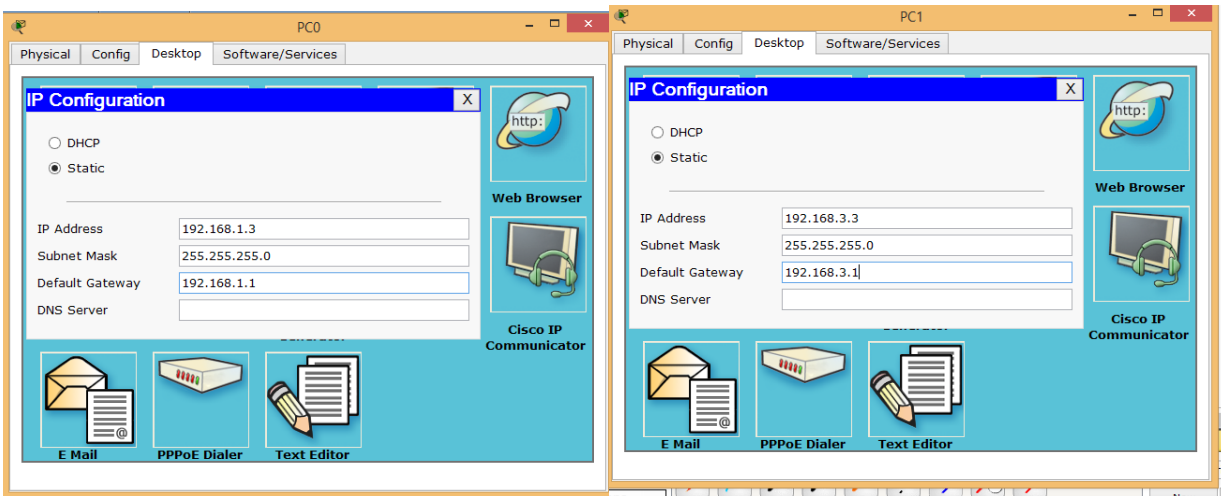
8. User-created packet box – Users can create highly-customized packets to test their topology from this area, and the results are displayed as a list.

3. Configure RIP

Step1: Create a topology as shown in fig 1. to apply RIP protocol .



Step2: Assigning IP address to PC0 and PC1



Step3: Assign IP addresses to all the fast Ethernet and serial interfaces and change the state of the interfaces from down to UP.

```
Router(config-if)#
Router(config-if)#exit
Router(config)#interface FastEthernet0/0
Router(config-if)#
Router(config-if)#exit
Router(config)#interface Serial2/0
Router(config-if)#ip address 192.168.2.1 255.255.255.0
Router(config-if)#clock rate 64000
Router(config-if)#no shutdown
Router(config-if)#
```

Fig.: Configuration of Router 0 i.e. configuring both serial and fast ethernet interfaces.

```
Router(config-if)#
Router(config-if)#exit
Router(config)#interface Serial2/0
Router(config-if)#ip address 192.168.2.3 255.255.255.0
Router(config-if)#
%LINK-5-CHANGED: Interface Serial2/0, changed state to up
no shutdown
Router(config-if)#
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial2/0, changed state to up
```

Fig.: Configuration of Router 1 i.e. configuring both serial and fast ethernet interfaces.

Step 4: We will apply RIP protocol commands on both routers

R1: In order to apply protocol RIP, we will write the following set of commands.
Router(config)# router rip
Router(config-router)# network 192.168.1.0
Router(config-router)# network 192.168.2.0
Router(config-router)# network 192.168.3.0
Router(config-router)#exit

R2: In order to apply protocol RIP, we will write the following set of commands on R2 as well.
Router(config)# router rip
Router(config-router)# network 192.168.1.0
Router(config-router)# network 192.168.2.0
Router(config-router)# network 192.168.3.0
Router(config-router)#exit

Step5: Now, you can check it. Traffic is enabled and you can easily send data from PC0 to PC1.

Algorithm:

1. Open Packet Tracer
2. Build the topology with help of devices and connections.
3. Configuring IP Addresses and Subnet Masks on the Hosts
4. Configure router with help of RIP/OSPF/BGP
5. Run the simulation and observer packet traversing

Application:
Packet tracer is useful to create network topologies using different network devices and connection and can simulate the packet traversing.

Conclusion
Hence we learn how to use packet tracer tool and simulate the network over it.

Assignment No: 14

Title: Network based Linux commands

Learning Objectives:
- To learn network based Linux commands

Problem Statement:
Basic Linux commands for networking

Learning Outcome: Students will able to
- Describe the use of different network based Linux commands

Software and Hardware Requirement:
- PC with network
- Linux Operating system

Theory:

1. **ipconfig:**

ipconfig Displays all current TCP/IP network configuration values and refreshes Dynamic Host Configuration Protocol (DHCP) and Domain Name System (DNS) settings. Used without parameters, ipconfig displays the IP address, subnet mask, and default gateway for all adapters.

2. **Pathping:**

Pathping provides information about network latency and network loss at intermediate hops between a source and destination .Pathping sends multiple Echo Request messages to each router between a source and destination over a period of time and then computes results based on the packets returned from each router. Because pathping displays the degree of packet loss at any given router or link, you can determine which routers or subnets might be having network problems.

3. **ARP:**

ARP command allows you to display and modify the Address Resolution Protocol (ARP)cache. An ARP cache is a simple mapping of IP addresses to MAC addresses. ARP (Address Resolution Protocol) is useful to view / add the contents of the kernel's ARP tables. To see default table use the command as.

4. **Netstat:**

Netstat is a useful tool for checking network and Internet connections. Some useful applications for the average PC user are considered, including checking for malware connections.

5. nbtstat:

nbtstat is designed to help troubleshoot NetBIOS name resolution problems. When a network is functioning normally, NetBIOS over TCP/IP (NetBT) resolves NetBIOS names to IP addresses. It does this through several options for NetBIOS name resolution, including local cache lookup, WINS server query, broadcast, LMHOSTS lookup, Hosts lookup, and DNS server query. The nbtstat command removes and corrects preloaded entries using a number of case-sensitive switches.

6. nslookup

nslookup is a network administration command-line tool available for many computer operating systems for querying the Domain Name System (DNS) to obtain domain name or IP address mapping or for any other specific DNS record.

7. Route:

Route displays and modifies the entries in the logical IP routing table.

8. tracert:

A utility that traces a packet from your computer to an Internet host, showing how many hops the packet requires to reach the host and how long each hop takes. If you're visiting a Web site and pages are appearing slowly, you can use traceroute to figure out where the longest delays are occurring.

9. **host:**

Host command to find name to IP or IP to name in IPv4 or IPv6 and also query DNS records.

Algorithm

Not applicable

Application:

Network commands are useful to configure a network and troubleshoot network problems.

Conclusion

Hence we learn different network based Linux commands.