Experiment No: 1

Aim:

Introduction to Local Area Network with its cables, connectors and topologies.

Theory:

<u>Local Area Network:</u>

A Local Area Network (LAN) is a group of computers and associated devices that share a common communications line or wireless link to a server. Typically, a LAN encompasses computers and peripherals connected to a server within a small geographic area such as an office building or home. Computers and other mobile devices can share resources such as a printer or network storage. A local area network may serve as few as two or three users (for example, in a home network) or as many hundreds of users. Ethernet is the most commonly used LAN technology. The different devices connected to the network are known as nodes and the edges are known as links.

<u>Links</u>

Telecommunication links can broadly be classified into guided and unguided media.
Guided Media:-It is the transmission media in which signals are confined to a specific path using wire or cable.

Unguided Media: Unguided or wireless media sends the data through air (or water), by freely releasing electromagnetic signals which are available to anyone who have a device capable of receiving them.

<u>Cables And Connectors Used In LAN</u>

<u>Twisted Pair Cable:</u>

It is further divided into two types namely shielded and unshielded twisted pair cables.

1. Unshielded Twisted Pair Cables:
   It is the most common type of telecommunication when compared with Shielded Twisted Pair Cable which consists of two conductors usually copper, each with its own colour plastic insulator. Identification is the reason behind coloured plastic insulation.
   UTP is most commonly connected to n/w devices via a snap-in plug like RJ45connector with 8 conductors.

2. Shielded twisted pair cables:
   This cable has a metal foil or braided-mesh covering which encases each pair of insulated conductors. Electromagnetic noise penetration is prevented by metal casing.STP Uses same connectors as UTP but shield must be connected to ground.

Coaxial Cable:

Coaxial is called by this name because it contains two conductors that are parallel to each other. Copper is used in this as centre conductor which can be a solid wire or a standard one. It is surrounded by PVC installation, a sheath which is encased in an outer conductor of metal foil, barid or both.
Outer metallic wrapping is used as a shield against noise and as the second conductor which completes the circuit. The outer conductor is also encased in an insulating sheath. The outermost part is the plastic cover which protects the whole cable.

There are two types of coaxial cables:

Baseband:
This is a 50 ohm (Ω) coaxial cable which is used for digital transmission. It is mostly used for LAN's. Baseband transmits a single signal at a time with very high speed. The major drawback is that it needs amplification after every 1000 feet.

Broadband:
This uses analog transmission on standard cable television cabling. It transmits several simultaneous signal using different frequencies. It covers large area when compared with Baseband Coaxial Cable.

Fiber Optic Cable:

These are similar to coaxial cable. It uses electric signals to transmit data. At the centre is the glass core through which light propagates.
In multimode fibres, the core is 50microns, and In single mode fibres, the thickness is 8 to 10 microns.
The core in fiber optic cable is surrounded by glass cladding with lower index of refraction as compared to core to keep all the light in core. This is covered with a thin plastic jacket to protect the cladding. The fibers are grouped together in bundles protected by an outer shield.
Fiber optic cable has bandwidth more than 2 gbps (Gigabytes per Second)

Topology:

The term topology refers to the way physically or logically arrangement of network.There are five basic topologies:
1) Bus topology
2) Ring topology
3) Star topology
4) Mesh topology
5) Tree Topology

Experiment No: 2

Aim:

To study about hubs and switches

Theory:

Switch-In a telecommunications network, a switch is a device that channels incoming data from any of multiple input ports to the specific output port that will take the data toward its intended destination. In the traditional circuit-switched telephone network, one or more switches are used to set up a dedicated though temporary connection or circuit for an exchange between two or more parties. On an Ethernet local area network (LAN), a switch determines from the physical device (Media Access Control or MAC) address in each incoming message frame which output port to forward it to and out of. In a wide area packet-switched network such as the Internet, a switch determines from the IP address in each packet which output port to use for the next part of its trip to the intended destination.

Hub- A hub is a common connection point for devices in a network. Hubs are commonly used to connect segments of a LAN. A hub contains multiple ports. When a packet arrives at one port, it is copied to the other ports so that all segments of the LAN can see all packets.

Advance switches technology issues-

Store and forward vs cut through

switch buffer limitation

layer 3 switching

ethernet base 10

network switching

operation of a switch

types of switch

Generally, Switches can be categorized as.

1) Mechanical Switches 2) Electrical/Electronic Switches

Both of these types of switches are widely used in Electrical and Electronics systems. Type of switch selection depends upon the system in which they are going to be incorporated. Switches can also be categories on many different bases. We will discuss them one by one later in this article.

Switches can also be categories on the basis of holding the current state.

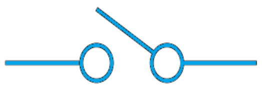
1) Late Switch

The latch switch holds its state whether ON or OFF until the new commands initiated.

2) Momentary Switch

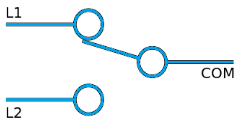
Mechanical Switches can be categories on the basis of their operation:

SPST (Single Pole Single Through)



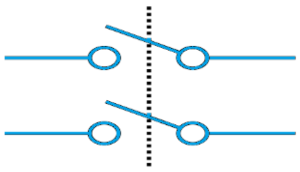
SPST (Single Pole Single Through) Switch

SPDT (Single Pole Double Throw)



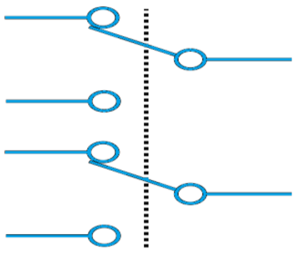
SPDT (Single Pole Double Throw) Switch

DPST (Double Pole, Single Throw)



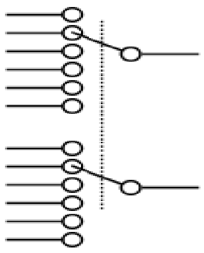
DPST (Double Pole, Single Throw) Switch

DPDT (Double Pole Double Throw)



DPDT (Double Pole Double Throw) switch

2P6T (Two Pole, Six Throw)



2P6T (Two Pole, Six Throw) Switch

Toggle switch:

CHARACTERISTICS OF A SWITCH

Experiment No:-3

Aim: To study about Wires, sockets and plugs.

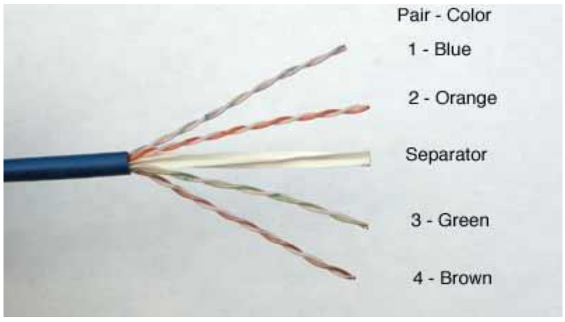
Theory:

- **CAT 5, CAR 5E AND CAT 6 WIRES, SOCKETS, MODULAR PLUGS**
- **HUB TO PLUG**
- **INSTALLATION OF UTP, COAXIAL CABLE,CROSS CABLE , PARALLEL CABLE**
- **UTP CABLE STANDARDS**

The section of the article (at the end, after External Links) that lists UTP and STP cable standards (Cat1, Cat2, etc.) states that Cat5 was used for 100BASE-TX and 1000BASE-T Ethernet. That should be 10BASE-T and 100BASE-TX. (I'll ignore the obsolete HP 100BASE-T4 "standard.") Cat5 was never appropriate, nor used, for 1000BASE-T, as that speed did not even exist when Cat5 cable was in vogue.64.192.2.189 (talk) 17:02, 10 November 2010 (UTC).1000BASE-T was designed to work on Cat5 because there was a lot of it sitting in cable plants at the time 1000BASE-T was developed. One thing that is possibly unclear is the fact that 10BASE-T can be run on Cat3 *and higher*. Should we add 10BASE-T to Cat5, Cat5e, etc? --Kvng (talk) 18:48, 10 November 2010 (UTC)

UTP TERMINATION

Connector Pinouts and Color Codes-In structured cabling as specified by TIA-568, there are two possible color codes or pinouts, called T568A and T568B which differ only in which color coded pairs are connected - pair 2 and 3 are reversed. Either work equally well, as long as you don't mix them! If you always use only one version, you're OK, but if you mix A and B in a cable run, you will get crossed pairs! While it makes no difference electrically which one you use, it is not wise to use both in one customer location. Always check to see which scheme has been used and continue using that one in new cabling installation.The cable pairs are color coded as shown:



Each pair consists of a colored wire and a white wire with a matching color stripe. The stripe wire is "tip" and the solid color wire is "ring," referring to the tip of the old 1/4" telephone plug and the ring around the shaft that makes the connections.

MTU

10BASE5, 10BASE2, 10BASET

Experiment No: 4

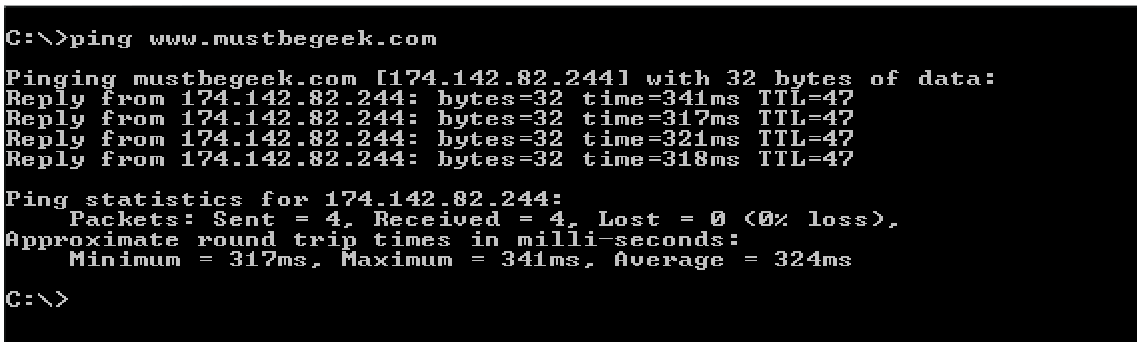
AIM:

To study about packet loss latency or delay, network throughput, collision, collision avoidance, difference between IPv6 and IPv4

THEORY:

Packet loss

Latency or Delay



```
C:\>ping www.mustbegeek.com

Pinging mustbegeek.com [174.142.82.244] with 32 bytes of data:
Reply from 174.142.82.244: bytes=32 time=341ms TTL=47
Reply from 174.142.82.244: bytes=32 time=317ms TTL=47
Reply from 174.142.82.244: bytes=32 time=321ms TTL=47
Reply from 174.142.82.244: bytes=32 time=318ms TTL=47

Ping statistics for 174.142.82.244:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 317ms, Maximum = 341ms, Average = 324ms

C:\>
```

The following useful information can be obtained from the output:-

1. The average round trip delay is 324ms
2. The delay time is between 317ms and 341ms (Min and Max round trip times)
3. Total of four packets was sent and received each of 32 bytes with no loss.

Network Throughput

Collision and collision avoidance

Difference between IPv6 and IPv4

- IPv4 addresses are 32 bit length but IPv6 addresses are 128 bit length.
- IPv4 addresses are binary numbers represented in decimals but IPv6 addresses are binary numbers represented in hexadecimals.
- IPSec support is only optional in IPv4 whereas IPv6 has inbuilt IPSec support.
- Fragmentation is done by sender and forwarding routers in IPv4 but in IPv6 fragmentation is done only by sender.
- No packet flow identification in IPv4 but packet flow identification is available within the IPv6 header using the Flow Label field.

Difference between IPv6 and IPv4 IP Addresses

An IP address is binary numbers but can be stored as text for human readers. For example, a 32-bit numeric address (IPv4) is written in decimal as four numbers separated by periods. Each number can be zero to 255. For example, 1.160.10.240 could be an IP address.IPv6 addresses are 128-bit IP address written in hexadecimal and separated by colons. An example IPv6 address could be written like this: 3ffe:1900:4545:3:200:f8ff:fe21:67cf

Experiment No: 5

AIM: Explain commands with example.

<u>nslookup</u>: 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.

<u>ping</u>: Helps in determining TCP/IP networks IP address as well as determine issues with the network and assists in resolving them. See the ping definition for a full description.

Example:

1 .Increase Ping Time Interval

Example: Wait for 5 seconds before sending the next packet.

```
$ ping -i 5 IP
```

Decrease Ping Time Interval

Example: Wait 0.1 seconds before sending the next packet.

```
$ ping -i 0.1
```

2. Check whether the local network interface is up and running.

Ping local host using zero (0)

```
$ ping 0
PING 0 (127.0.0.1) 56(84) bytes of data.
```

<u>ipconfig</u>:

To display the basic TCP/IP configuration for all adapters, type.

```
ipconfig /all
```

To renew a DHCP-assigned IP address configuration for only the Local Area Connection adapter, type:

```
ipconfig /renew "Local Area Connection"
```

To flush the DNS resolver cache when troubleshooting DNS name resolution problems, type:

```
ipconfig /flushdns
```

To display the DHCP class ID for all adapters with names that start with Local, type:

```
ipconfig /showclassid Local*
```

<u>pathping</u>: Similar to the tracert command, pathping provides users with the ability of locating spots that have network latency and network loss.

<u>arp</u>: Displays, adds , and removes arp information from network devices.

```
arp -a
Interface 220.0.0.80
Internet Address        Physical address        Type
   220.0.0.160          00-50-04-62-F7-23       static
```

Experiment No: 6

Aim: Given a IP address in binary format, identify the class to which it belongs.

Theory:

Class A address: Class A addresses always have the first bit of their IP addresses set to “0”. Since Class A networks have an 8-bit network mask, the use of a leading zero leaves only 7 bits for the network portion of the address, allowing for a maximum of 128 possible network numbers, ranging from 0.0.0.0 – 127.255.255.255 Number 127.x.x.x is reserved for loopback, used for internal testing on the local machine.

Class B address: Class B addresses always have the first bit set to “1” and their second bit set to “0”. Since Class B addresses have a 16-bit network mask, the use of a leading “10” bit-pattern leaves 14 bits for the network portion of the address, allowing for a maximum of 16,384 networks, ranging from 128.0.0.0 – 191.255.255.255

Class C address: Class C addresses have their first two bits set to “1” and their third bit set to “0”. Since Class C addresses have a 24-bit network mask, this leaves 21 bits for the network portion of the address, allowing for a maximum of 2,097,152 network addresses, ranging from 192.0.0.0 – 223.255.255.255

Class D address: Class D addresses are used for multicasting applications. Class D addresses have their first three bits set to “1” and their fourth bit set to “0”. Class D addresses are 32-bit network addresses, meaning that all the values within the range of 224.0.0.0 – 239.255.255.255 are used to uniquely identify multicast groups.

Class E address: Class E addresses are defined as experimental and are reserved for future testing purposes. They have never been documented or utilized in a standard way. Class E has its highest bit order set to **1-1-1-1.**

PROGRAM:

Experiment No. 7

Aim: Implement client server program.

Description:

This Java Program

1. Make Use of Java Socket Programming

2. It starts a server which will be always running listening to a port 9999 (Server.java)

3. Client (Client.java) sends a number (message) to the server

4. Server receives this number and multiplies it by 2

5. Server (Server.java) sends back the result (message) to the client (Client.java)

6. In case the number sent by the client was not a proper number, server (Server.java) sends back

   the message “Please send a proper number” to the client (Client.java)

Program:

For client

For server

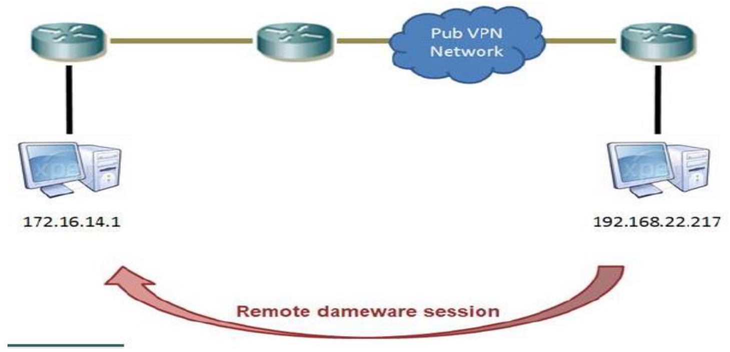
Output

Experiment No: 8

Aim: Installation and working of Telnet (Terminal Network).

Hardware Required:LAN Card, LAN drivers, 2-computers, Modem, Cables

Theory: It's an abbreviation for Terminal Network.



Telnet Usage

Telnet is a protocol that allows a user to log onto other computers. You use an IP address or domain name to log on. Bulletin boards are still available to play games, download files or read information. In addition, you can play games with your friends over this type of network. Telnet is not as common as it once was. Nevertheless, it is a simple method of connecting to different friends or online communities. Telnet is a user command and an underlying TCP/IP protocol for accessing remote computers. Through Telnet, an administrator or another user can access someone else's computer remotely. On the Web, HTTP and FTP protocols allow you to request specific files from remote computers, but not to actually be logged on as a user of that computer. With Telnet, you log on as a regular user with whatever privileges you may have been granted to the specific application and data on that computer. A Telnet command request looks like this (the computer name is made-up): telnet the.library at.whatis.edu. The result of this request would be an invitation to log on with a user-id and a prompt for a password. If accepted, you would be logged on like any user who used this computer every day. Telnet is most likely to be used by program developers and anyone who has a need to use specific applications or data located at a particular host computer.

Procedure:

1. 1 Go to ―My Computer right click and select properties.
2. Then go to Manage. In the opened window select ―Services and Application then select ―Telnet from right hand side of window.
3. In property window of telnet set ―start-up box to ―Automatic
4. Go to Start-1.All Program 2. Accessories 3. Command prompt
5. In ―C: prompt(C: \>) type telnet and type the IP of the 2nd computer after space. Eg : telnet 192.27.24.16.
6. Enter username and password of 2nd computer when prompted.
7. To quit type ―exit

Properties Of Telnet:

- Telnet is done via command prompt
- Telnet works on password protected system

Telnet service must be ―ON on both the system

Experiment No: 9

Aim: Installation and Configuration of FTP Server.

Requirements: The File Transfer Protocol (FTP) is used as one of the most common means of copying files between servers over the Internet.

Implementation Details

Configuration files of ftp server (vsftpd daemon).The examples of configuration files are as follows:

1. Package: vsftpd
2. Daemon: /usr/sbin/vsftpd
3. Script; /etc/init.d/vsftpd
4. Configuration: /etc/vsftpd.conf, /etc/pam.d/vsftpd
5. Log: /var/log/xferlog

Steps to install and configure the ftp server:

Perform the following steps to install and configure the FTP server (vsftpd daemon)

Installing the VSFTPD Daemon:

Go To Synaptic Package Manager and apply the VSFTPD daemon.

Type the following command to install the FTP server.

```
sudo apt-get install vsftpd
```

Starting the VSFTP Daemon:

Use the following commands to start, stop and restart VSFTPD after booting:

```
/etc/init.d/vsftpd start
/etc/ init.d/vsftpd stop
/etc/init.d/vsftpd restart
```

Testing the status of VSFTPD:You can always test whether the VSFTPD process is running by using the netstat command which lists all the TCP and UDP ports on which the server is listening for Traffic.Edit the vsftpd.conf file:The vsftpd.conf file uses a number of default settings that you need to know. You can check the vsftpd.conf file for the various parameters. Descriptions on this and more can be found in the vsftpd.conf man (manual) page.Change the FTP greeting banner:Change the default greeting banner in the vsftpd.conf file to make it harder for malicious Users to determine the type of system you have. The directive in this file is :ftpd_banner = welcome to F.R.C.R.C.E. site

FTP Client

Using Anonymous Access:

The following steps are performed to use anonymous access of FTP server:

1. Add the TEST File to the /home/ftp directory. Use the following commands to add the test file “hello.txt”

```
cd/home/ftp
nano Hello.txt (Save and exit from the nano)
```

2. Connect to the Server via FTP.

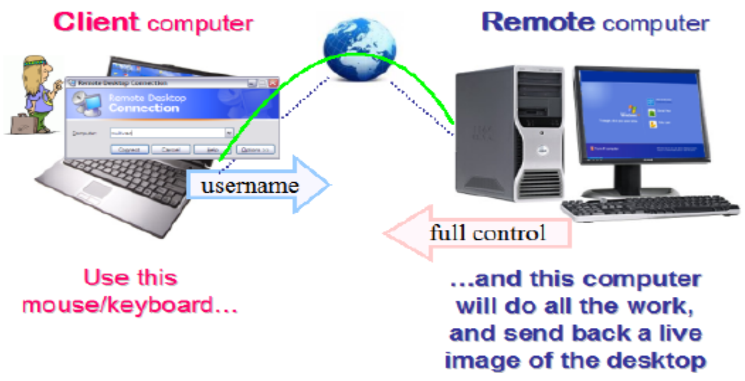
Type the following commands at the root mode to connect to the server using FTP:

```
# ftp 8.6..8.18 Connected to 8.6.8.18
ftp>bin (For binary mode data transfer)
ftp>ha
```

Experiment No: 10

Aim: Installation & working of Remote desktop.
Hardware Required:
LAN Card, LAN drivers, 2-computers, Modem, Cables

Theory:
Remote Desktop, a function included with Windows XP Professional, enables you to connect to your computer across the Internet from virtually any computer, Pocket PC, or Smartphone. Unlike a typical VPN connection (which will give a remote PC access to the company network) Remote Desktop will actually allow you to see and control your connected PC as though you were sitting directly in front of it. Remote desktop technology makes it possible to view another computer's desktop on your computer. This means you can open folders, move files, and even run programs on the remote computer, right from your own desktop. Both Windows and Macintosh computer support remote desktop connections, though they use different implementations.


Client computer
Remote computer
username
full control
Use this mouse/keyboard...
...and this computer will do all the work, and send back a live image of the desktop

PROCEDURE:
1. Go to My Computer properties window by right clicking and selecting properties from menu that appears
2. Select Remote* for successful connection we have to perform 3 things: -
   - Check the checkbox ―Allow users to connect remotely to this computer
   - Turn the Firewall off
   - Set password by: 1.Right clicking my computer and click Manage.
     2. Select --Local user & group‖
     3. Right click on ―Administrator

3. Set IP of 2nd computer in ―Remote Desktop Connection‖ window. To open ―Remote Desktop Connection Start
   - All programs
   - Accessories
   - Communication
   - Remote Desktop Connection
     4. Give username and password of 2ndcomputer. The 2ndcomputer will automatically Log off while 1st one is working remotely on it. The remote connection is last once the user of 2nd computer logs in again.

EXPERIMENT-11

Aim-: To study the network topology and IP Addressing scheme of Institute Network.
Apparatus required-: Computer systems with intranet connection, interfaces.
Theory-:

Network topology is the arrangement of the various elements (links, nodes, etc.) of a computer network Essentially, it is the topological structure of a network, and may be depicted physically or logically. *Physical* topology refers to the placement of the network's various components, including device location and cable installation, while *logical* topology shows how data flows within a network, regardless of its physical design. Distances between nodes, physical interconnections, transmission rates, and/or signal types may differ between two networks, yet their topologies may be identical.

A good example is a local area network (LAN): Any given node in the LAN has one or more physical links to other devices in the network; graphically mapping these links results in a geometric shape that can be used to describe the physical topology of the network. Conversely, mapping the data flow between the components determines the logical topology of the network.

IP Address:-An **Internet Protocol address** (**IP address**) is a numerical label assigned to each device (e.g., computer, printer) participating in a computer network that uses the Internet Protocol for communication. An IP address serves two principal functions: host or network interface identification and location addressing. Its role has been characterized as follows: "Aname indicates what we seek. An address indicates where it is. A route indicates how to get there.

The designers of the Internet Protocol defined an IP address as a 32-bit number consisting of 4 octetsand this system, known as Internet Protocol Version 4 (IPv4), is still in use today. However, due to the enormous growth of the Internet and the predicted depletion of available addresses, a new version of IP (IPv6), using 128 bits for the address, was developed in 1995. IPv6 was standardized as RFC 2460 in 1998, and its deployment has been ongoing since the mid-2000s.

IP addresses are binary numbers, but they are usually stored in text files and displayed in human-readable notations, such as 172.16.254.1 (for IPv4), and 2001:db8:0:1234:0:567:8:1 (for IPv6).

The Internet Assigned Numbers Authority (IANA) manages the IP address space allocations globally and delegates fiveregional Internet registries (RIRs) to allocate IP address blocks to local Internet registries (Internet service providers) and other entities.

INSTITUTION NETWORK TOPOLOGY AND IP ADDRESSING

In our institute, we have 7 Virtual LAN’s (VLAN). The VLAN numbers are:-

VLAN1 – 192.168.00.1For Science/ETC/EE Block

VLAN2 – 192.168.08.1For CIVIL /MECH/EEE Block

VLAN3 – 192.168.16.1For I.T/CSE Block

VLAN4 – 192.168.24.1For SSEC

VLAN5 – 192.168.32.1For WIFI

VLAN6 – 192.168.40.1For SSITM

VLAN7 – 192.168.47.1For SSITM

All the buildings including SSEC,SSITM,SHRISHANKARACHARYA MAHAVIDYALAYA are connected in one LAN.

The Main leased line of internet from service provider is connected to the SSCET main server which is situated at administration. Through optical fibre cable all the blocks of sscet, SSEC, SSITM, Shri ShankaracharyaMahavidyalaya are connected in a commom LAN.

Individual control rooms are situated in each area by which all other switches are connected.

In each labs there are many switches by which further system or computers are connected.

EXPERIMENT -12

Aim-: Aim to Implement OSI Model using C & its Execution.

Apparatus Required-: P4 Computer, Network Connectivity, Turbo C.

Theory-: The Open Systems Interconnection (OSI) model (ISO/IEC 7498-1) is a conceptual model that characterizes and standardizes the internal functions of a communication system by partitioning it into abstraction layers. The model is a product of the Open Systems Interconnection project at the International Organization for Standardization (ISO).The model groups' similar communication functions into one of seven logical layers. A layer serves the layer above it and is served by the layer below it. For example, a layer that provides error-free communications across a network provides the path needed by applications above it, while it calls the next lower layer to send and receive packets that make up the contents of that path. Two instances at one layer are connected by a horizontal connection on that layer.According to recommendation X.200; there are seven layers, labelled 1 to 7, with layer 1 at the bottom. Each layer is generically known as an N layer. An "N+1 entity" (at layer N+1) requests services from an "N entity" (at layer N).

OSI Model			
	Data unit	**Layer**	**Function**
Host layers	Data	7. Application	Network process to application
		6. Presentation	Data representation, encryption and decryption, convert machine dependent data to machine independent data
		5. Session	Interhost communication, managing sessions between applications
	Segments	4. Transport	Reliable delivery of packets between points on a network.
Media layers	Packet/Datagram	3. Network	Addressing, routing and (not necessarily reliable) delivery of datagrams between points on a network.
	Bit/Frame	2. Data link	A reliable direct point-to-point data connection.
	Bit	1. Physical	A (not necessarily reliable) direct point-to-point data connection.

Program code

EXPERIMENT-13

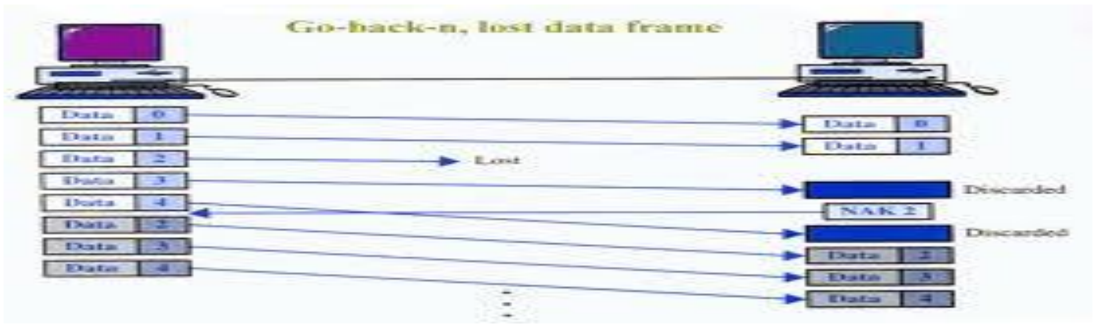
Aim-: Aim to Implement GO BACK N ARQ using C & its Execution.
Apparatus Required-: P4 Computer, Network Connectivity, Turbo C.
Theory-:

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.

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 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.

Diagram-:



Program Code:

EXPERIMENT-14

Aim-: Aim to Implement Bit stuffing using C & its Execution.
Apparatus Required-: P4 Computer, Network Connectivity, Turbo C.
Theory-:

Bit stuffing is the process of inserting noninformation bits into data to break up bit patterns to affect the synchronous transmission of information. It is widely used in network and communication protocols, in which bit stuffing is a required part of the transmission process. Bit stuffing is commonly used to bring bit streams up to a common transmission rate or to fill frames. Bit stuffing is also used for run-length limited coding.

In order to fill bit frames, the position where the new bits are stuffed is communicated to the receiving end of the data link. The receiver removes the extra bits to return the bit streams to their original bit rate. This is used when a communication protocol requires a fixed frame size. Bits are inserted to make the frame size equal to the defined frame size.

Bit stuffing also works to limit the number of consecutive bits of the same value included in the transmitted data for run-length limited coding. This procedure includes a bit of the opposite value after the maximum allowed number of consecutive bits of the same value. For instance, if a number of zero bits are transmitted consecutively, the receiving end loses synchronization because a lot of time has passed without voltage sensing. Using bit stuffing, sets of bits beginning with the number one are stuffed into streams of zeros at specific intervals. The receiver does not require any extra information regarding the bit location when the extra bits are removed. Such bit stuffing is done to ensure reliable data transmission and ensure that transmissions start and end at the right places, among other purposes.

//C PROGRAM TO IMPLEMEMENT BIT STUFFING//

EXPERIMENT-15

Aim-: To implement CRC program.

Apparatus Required-: P4 Computer, Network Connectivity, Turbo C.

Theory-:

A cyclic redundancy check (CRC) is an error-detecting code commonly used in digital networks and storage devices to detect accidental changes to raw data. Blocks of data entering these systems get a short *check value* attached, based on the remainder of a polynomial division of their contents; on retrieval the calculation is repeated, and corrective action can be taken against presumed data corruption if the check values do not match.

CRCs are so called because the *check* (data verification) value is a *redundancy* (it expands the message without adding information) and the algorithm is based on *cyclic* codes. CRCs are popular because they are simple to implement in binary hardware, easy to analyze mathematically, and particularly good at detecting common errors caused by noise in transmission channels. Because the check value has a fixed length, the function that generates it is occasionally used as a hash function.

CRCs are based on the theory of cyclic error-correcting codes. The use of systematic cyclic codes, which encode messages by adding a fixed length check value, for the purpose of error detection in communication networks, was first proposed by W. Wesley Peterson during 1961. Cyclic codes are not only simple to implement but have the benefit of being particularly well suited for the detection of burst errors, contiguous sequences of erroneous data symbols in messages. This is important because burst errors are common transmission errors in many communication channels,including magnetic and optical storage devices. Typically an *n*-bit CRC applied to a data block of arbitrary length will detect any single error burst not longer than *n* bits and will detect a fraction $1 - 2{-n}$ of all longer error bursts.

Specification of a CRC code requires definition of a so-called generator polynomial. This polynomial becomes the divisor in a polynomial long division, which takes the message as the dividend and in which the quotient is discarded and the remainder becomes the result. The important caveat that the polynomial coefficients are calculated according to the arithmetic of a finite field, so the addition operation can always be performed bitwise parallel (there is no carry between digits). The length of the remainder is always less than the length of the generator polynomial, which therefore determines how long the result can be.
To compute an *n*-bit binary CRC, line the bits representing the input in a row, and position the (n + 1)-bit pattern representing the CRC's divisor (called a "polynomial") underneath the left-hand end of the row.

In this example, we shall encode 14 bits of message with a 3-bit CRC, with a polynomial x^3+x+1. The polynomial is written in binary as the coefficients; a 3rd order polynomial has 4 coefficients. In this case, the coefficients are 1,0, 1 and 1. The result of the calculation is 3 bits long.

Start with the message to be encoded:

```
11010011101100
```

This is first padded with zeroes corresponding to the bit length *n* of the CRC. Here is the first calculation for computing a 3-bit CRC:

```
11010011101100 000 <--- input right padded by 3 bits
1011 <--- divisor (4 bits) = x³+x+1
 ------------------
01100011101100 000 <--- result
```

The algorithm acts on the bits directly above the divisor in each step. The result for that iteration is the bitwise XOR of the polynomial divisor with the bits above it. The bits not above the divisor are simply copied directly below for that step. The divisor is then shifted one bit to the right, and the process is repeated until the divisor reaches the right-hand end of the input row. Here is the entire calculation:

```
11010011101100 000 <--- input right padded by 3 bits
1011 <--- divisor
01100011101100 000 <--- result (note the first four bits are the XOR with the divisor
beneath, the rest of the bits are unchanged)
1011 <--- divisor ...
00111011101100 000
1011
00010111101100 000
1011
00000001101100 000 <--- note that the divisor moves over to align with the next 1 in
the dividend (since quotient for that step was zero)
1011 (in other words, it doesn't necessarily move one bit per iteration)
00000000110100 000
```

```
1011
00000000011000 000
1011
00000000001110 000
1011
00000000000101 000
101 1
-----------------
00000000000000 100 <--- remainder (3 bits). Division algorithm stops here as
quotient is equal to zero.
```

Since the leftmost divisor bit zeroed every input bit it touched, when this process ends the only bits in the input row that can be nonzero are the n bits at the right-hand end of the row. These *n* bits are the remainder of the division step, and will also be the value of the CRC function (unless the chosen CRC specification calls for some postprocessing).

The validity of a received message can easily be verified by performing the above calculation again, this time with the check value added instead of zeroes. The remainder should equal zero if there are no detectable errors.

```
11010011101100 100 <--- input with check value
1011 <--- divisor
01100011101100 100 <--- result
1011 <--- divisor ...
00111011101100 100
......
00000000001110 100
1011
00000000000101 100
101 1
------------------
0 <--- remainder
```

// C PROGRAM TO IMPLEMENT CRC.//

EXPERIMENT-16

Aim-: Aim to Implement STOP & WAIT ARQ using C & its Execution.
Apparatus Required-: P4 Computer, Network Connectivity, Turbo C.
Theory-:

Stop-and-wait ARQ also known as Alternating-Bit-Protocol is a method used in telecommunications to send information between two connected devices. It ensures that information is not lost due to dropped packets and that packets are received in the correct order. It is the simplest kind of automatic repeat-request (ARQ) method. A stop-and-wait ARQ sender sends one frame at a time; it is a special case of the general sliding window protocol with both transmit and receive window sizes equal to 1. After sending each frame, the sender doesn't send any further frames until it receives an acknowledgement (ACK) signal. After receiving a good frame, the receiver sends an ACK. If the ACK does not reach the sender before a certain time, known as the timeout, the sender sends the same frame again.

The above behavior is the simplest Stop-and-Wait implementation. However, in a real life implementation there are problems to be addressed.

Typically the transmitter adds a redundancy check number to the end of each frame. The receiver uses the redundancy check number to check for possible damage. If the receiver sees that the frame is good, it sends an ACK. If the receiver sees that the frame is damaged, the receiver discards it and does not send an ACK—pretending that the frame was completely lost, not merely damaged.

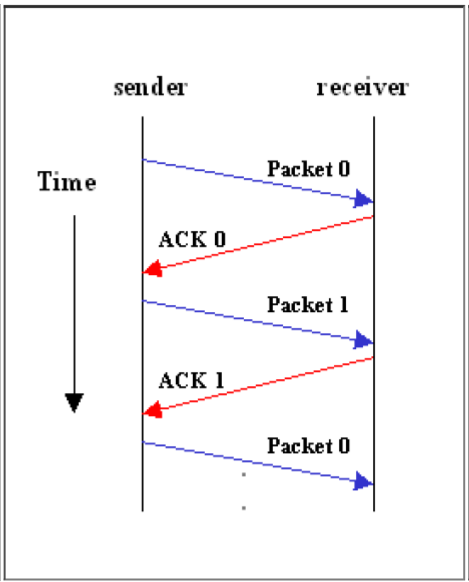
One problem is where the ACK sent by the receiver is damaged or lost. In this case, the sender doesn't receive the ACK, times out, and sends the frame again. Now the receiver has two copies of the same frame, and doesn't know if the second one is a duplicate frame or the next frame of the sequence carrying identical data.

Another problem is when the transmission medium has such a long latency that the sender's timeout runs out before the frame reaches the receiver. In this case the sender resends the same packet. Eventually the receiver gets two copies of the same frame, and sends an ACK for each one. The sender, waiting for a single ACK, receives two ACKs, which may cause problems if it assumes that the second ACK is for the next frame in the sequence.

To avoid these problems, the most common solution is to define a 1 bit *sequence number* in the header of the frame. This sequence number alternates (from 0 to 1) in subsequent frames. When the receiver sends an ACK, it includes the sequence number of the next packet it expects. This way, the receiver can detect duplicated frames by checking if the frame sequence numbers alternate. If two subsequent frames have the same sequence number, they are duplicates, and the second frame is discarded. Similarly, if two subsequent ACKs reference the same sequence number, they are acknowledging the same frame.

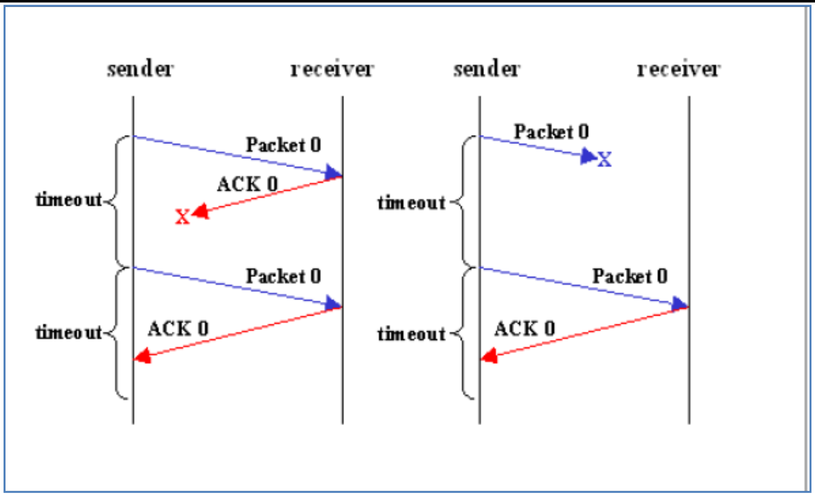
Stop-and-wait ARQ is inefficient compared to other ARQs, because the time between packets, if the ACK and the data are received successfully, is twice the transit time (assuming the turnaround time can be zero). The throughput on the channel is a fraction of what it could be. To solve this problem, one can send more than one packet at a time with a larger sequence number and use one ACK for a set. This is what is done in Go-Back-N ARQ and the Selective Repeat ARQ.

Diagram-:



After transmitting one packet, the sender waits for an ***acknowledgment (ACK)*** from the receiver before transmitting the next one. In this way, the sender can recognize that the previous packet is transmitted successfuly and we could say "stop-n-wait" guarantees reliable transfer between nodes.

To support this feature, the sender keeps a record of each packet it sends.
Also, to avoid confusion caused by delayed or duplicated ACKs, "stop-n-wait" sends each packets with unique sequence numbers and receives that numbers in each ACKs.



Program code:

EXPERIMENT-17

Aim-: Aim to Implement Selective –N-Reject ARQ using C & its Execution.
Apparatus Required-: P4 Computer, Network Connectivity, Turbo C.
Theory-:
Selective Repeat is one of the automatic repeat-request (ARQ) techniques. With selective repeat, the sender sends a number of frames specified by a window size even without the need to wait for individual ACK from the receiver as in Go-back N ARQ. However, the receiver sends ACK for each frame individually, which is not like cumulative ACK as used with go-back-n. The receiver accepts out-of-order frames and buffers them. The sender individually retransmits frames that have timed out.

It may be used as a protocol for the delivery and acknowledgement of message units, or it may be used as a protocol for the delivery of subdivided message sub-units.

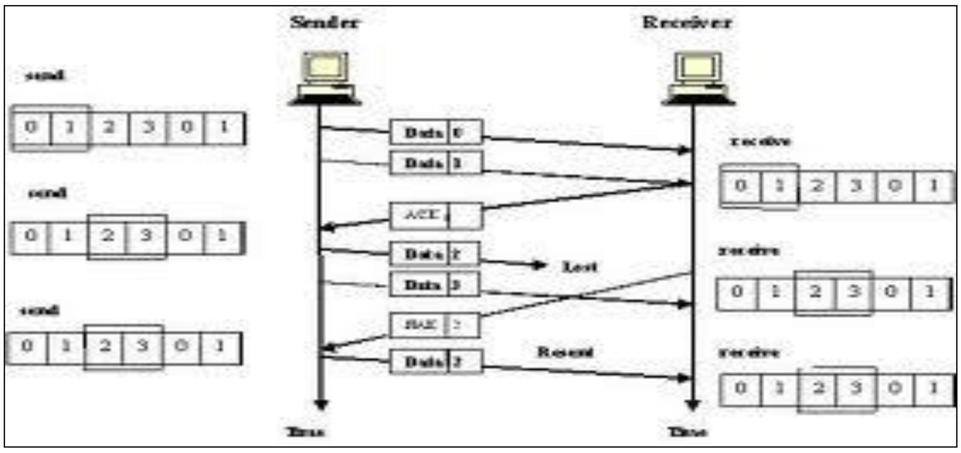
When used as the protocol for the delivery of messages, the sending process continues to send a number of frames specified by a window size even after a frame loss. Unlike Go-Back-N ARQ, the receiving process will continue to accept and acknowledge frames sent after an initial error; this is the general case of the sliding window protocol with both transmit and receive window sizes greater than 1.

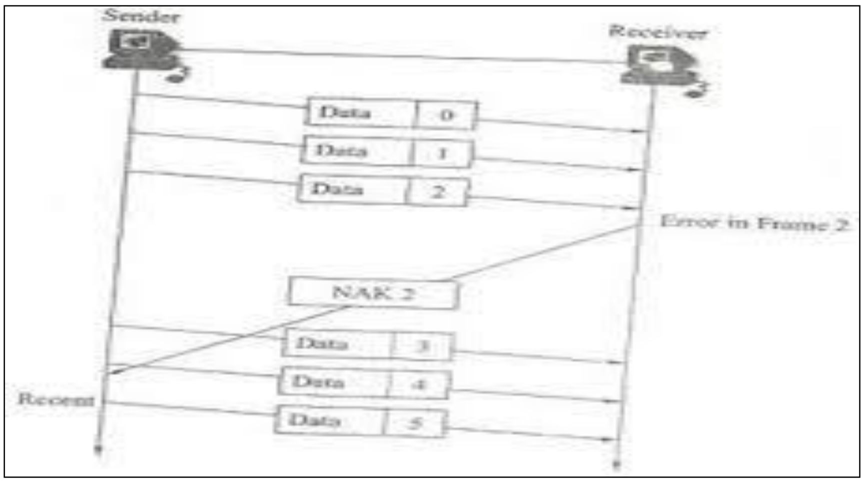
The receiver process keeps track of the sequence number of the earliest frame it has not received, and sends that number with every acknowledgement (ACK) it sends. If a frame from the sender does not reach the receiver, the sender continues to send subsequent frames until it has emptied its window. The 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 re-sends the frame number given by the ACKs, and then continues where it left off.

The size of the sending and receiving 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. To understand this, consider the case when all ACKs are destroyed. If the receiving window is larger than half the maximum sequence number, some, possibly even all, of the packages that are resent after timeouts are duplicates that are not recognized as such. The sender moves its window for every packet that is acknowledged.[1]

When used as the protocol for the delivery of subdivided messages it works somewhat differently. In non-continuous channels where messages may be variable in length, standard ARQ or Hybrid ARQ protocols may treat the message as a single unit. Alternately selective retransmission may be employed in conjunction with the basic ARQ mechanism where the message is first subdivided into sub-blocks (typically of fixed length) in a process called packet segmentation. The original variable length message is thus represented as a concatenation of a variable number of sub-blocks. While in standard ARQ the message as a whole is either acknowledged (ACKed) or negatively acknowledged (NAKed), in ARQ with selective transmission the NAKed response would additionally carry a bit flag indicating the identity of each sub-block successfully received. In ARQ with selective retransmission of sub-divided messages each retransmission diminishes in length, needing to only contain the sub-blocks that were NAKed.

In most channel models with variable length messages, the probability of error-free reception diminishes in inverse proportion with increasing message length. In other words it's easier to receive a short message than a longer message. Therefore standard ARQ techniques involving variable length messages have increased difficulty delivering longer messages, as each repeat is the full length. Selective re-transmission applied to variable length messages completely eliminates the difficulty in delivering longer messages, as successfully delivered sub-blocks are retained after each transmission, and the number of outstanding sub-blocks in following transmissions diminishes.





Program Code:

Experiment 17

AIM: Installation and working with FTP (File Transfer Protocol). Write a program to simulate FTP Server.

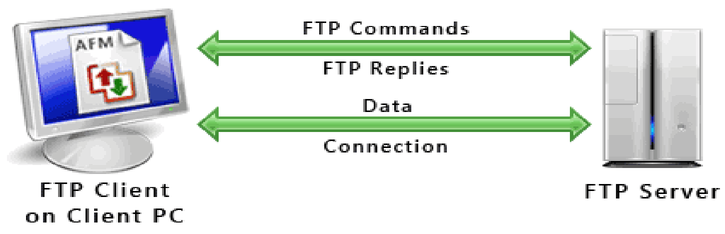
Theory: Understanding How FTP Works Data exchange has been important from the early days of computing. A popular means of data exchange is connecting computers to one another. The File Transfer Protocol (FTP) is used to transfer files between two computers over a network and Internet. In this article we will look at how to work with a FTP client. Auto FTP Manager is an advanced FTP client that automates file transfers between your computer and the FTP server.

What is FTP?

When you want to copy files between two computers that are on the same local network, often you can simply "share" a drive or folder, and copy the files the same way you would copy files from one place to another on your own PC.

What if you want to copy files from one computer to another that is halfway around the world? You would probably use your Internet connection. However, for security reasons, it is very uncommon to share folders over the Internet. File transfers over the Internet use special techniques, of which one of the oldest and most widely-used is FTP. **FTP,** short for**"File Transfer Protocol,"** can transfer files between any computers that have an Internet connection, and also works between computers using totally different operating systems.

Transferring files from a client computer to a server computer is called **"uploading"** and transferring from a server to a client is **"downloading".**



Requirements for using FTP

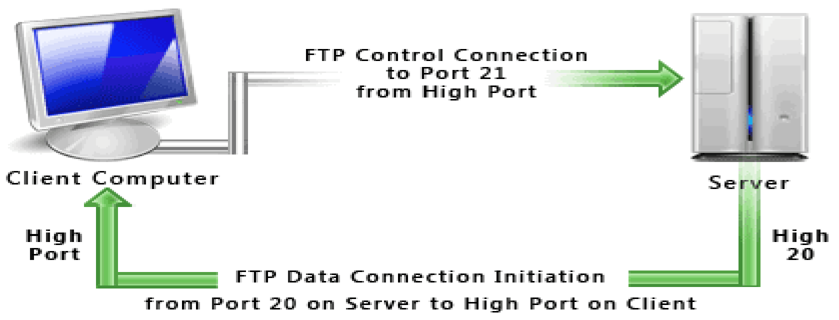
1. An FTP client like Auto FTP Manager installed on your computer
2. Certain information about the FTP server you want to connect to:
   a. The **FTP server address.** This looks a lot like the addresses you type to browse web sites.
   Example : Server address is "ftp.videodesk.net".
   Sometimes the server address will be given as a numeric address, like "64.185.225.87".
   b. A user name and password. Some FTP servers let you connect to them anonymously.
   For anonymous connections, you do not need a user name and password.

To transfer files, provide your client software (Auto FTP Manager) with the server address, user name, and password. After connecting to the FTP server, you can use Auto FTP Manager's **File Manager** to upload, download and delete files. Using the File Manager is a lot like working with Windows Explorer.

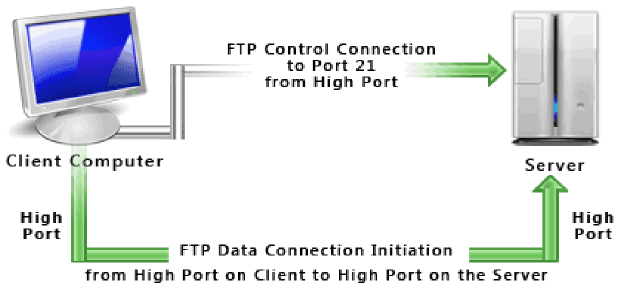
FTP and Internet Connections FTP uses one connection for commands and the other for sending and receiving data. FTP has a standard port number on which the FTP server "listens" for connections. A port is a "logical connection point" for communicating using the Internet Protocol (IP). The standard port number used by FTP servers is 21 and is used only for sending commands. Since port 21 is used exclusively for sending commands, this port is referred to as a **command port.** For example, to get a list of folders and files present on the FTP server, the FTP Client issues a "LIST" command. The FTP server then sends a list of all folders and files back to the FTP Client. So what about the internet connection used to send and receive data? The port that is used for transferring data is referred to as a **data port.** The number of the data port will vary depending on the "mode" of the connection. (See below for Active and Passive modes.)

Active and Passive Connection Mode

The FTP server may support **Active** or **Passive** connections, or both. In an Active FTP connection, the client opens a port and listens and the server actively connects to it. In a Passive FTP connection, the server opens a port and listens (passively) and the client connects to it. You must grant Auto FTP Manager access to the Internet and to choose the right type of FTP Connection Mode.Most FTP client programs select passive connection mode by default because server administrators prefer it as a safety measure. Firewalls generally block connections that are "initiated" from the outside. Using passive mode, the FTP client (like Auto FTP Manager) is "reaching out" to the server to make the connection. The firewall will allow these outgoing connections, meaning that no special adjustments to firewall settings are required. If you are connecting to the FTP server using **Active mode** of connection you must set your firewall to accept connections to the port that your FTP client will open. However, many Internet service providers block incoming connections to all ports above 1024. Active FTP servers generally use port 20 as their data port.



It's a good idea to use **Passive mode** to connect to an FTP server. Most FTP servers support the Passive mode. For Passive FTP connection to succeed, the FTP server administrator must set his / her firewall to accept all connections to any ports that the FTP server may open. However, this is the server administrator's problem (and standard practice for servers). You can go ahead, make and use FTP connections.



Once the FTP Client manages to open the internet connections, one for command and one for data, it starts communicating with the FTP server. You are now ready to transfer your files and folders between the two connected computers with Auto FTP Manager.

Step-by-step guide: How to install an FTP server on Windows

the FTP server to share files with other computers on your home network, or you can make the FTP server available to people on the Internet.

FTP is the file transfer protocol. Anyone with an FTP client application (such as Internet Explorer, CuteFTP, WS-FTP and others) can connect to your FTP server. Here's how you install it:
Step 1: Log in as a member of the Administrators group on your Windows XP computer.
Step 2: Click Start, and then click Control Panel.
Step 3: With the Control Panel in category view, double click on the Add or Remove Programs icon.
Step 4: In the Add or Remove Programs dialog box, click on the Add/Remove Windows Components button on the left side of the dialog box.
Step 5: In the Windows Components dialog box, scroll down to find the Internet Information Servers (IIS) entry. Click on that entry and then click the Details button.
Step 6: Put a checkmark in the following check boxes: Common Files, Documentation, File Transfer Protocol (FTP) Service, Internet Information Services, Snap-In and SMTP Service. Click OK, and then click Next in the Windows Component Wizard dialog box. The Wizard will complete the installation process.
Step 7: After the FTP Service is installed, click Start and point to All Programs. Point to Administrative Tools and click on Internet Information Services. This opens the Internet Information Services console.
Step 8: To put files into your FTP directory, just open Windows Explorer and navigate to your InetPubftproot folder. Copy your files into the ftproot folder. Those files will then be available to users who connect to your FTP folder.
Step 9: After you put files into your FTP folder, open Internet Explorer and type ftp://computername. Computername is the name of the computer that you installed the FTP onto. You'll see a list of files and folders in your FTP directory. If you put your FTP server on the Internet, users will have to use an IP address instead of a computer name, since your home network computer names are not available to Internet users.

Experiment 18

AIM: Write a program to encrypt and decrypt a file using a key.

Theory: Encryption is the process of encoding messages or information in such a way that only authorized parties can read it. Encryption does not of itself prevent interception, but denies the message content to the interceptor. In an encryption scheme, the intended communication information or message, referred to as plaintext, is encrypted using an encryption algorithm, generating ciphertext that can only be read if decrypted. For technical reasons, an encryption scheme usually uses a pseudo-random encryption key generated by an algorithm. It is in principle possible to decrypt the message without possessing the key, but, for a well-designed encryption scheme, large computational resources and skill are required. An authorized recipient can easily decrypt the message with the key provided by the originator to recipients, but not to unauthorized interceptors.

Encryption has widely been used to protect data in numerous areas, such as e-commerce, online banking, cloud storage, online communication and so forth.

Decryption :

Decryption is the process of transforming data that has been rendered unreadable through encryption back to its unencrypted form. In decryption, the system extracts and converts the garbled data and transforms it to texts and images that are easily understandable not only by the reader but also by the system. Decryption may be accomplished manually or automatically. It may also be performed with a set of keys or passwords.

Programming

OUTPUT :

EXPERIMENT-19

AIM- Introduction to Network Simulator Version 2 (NS2).

THEORY-

Network Simulator (Version 2), widely known as NS2, is simply an event driven simulation tool that has proved useful in studying the dynamic nature of communication networks. Simulation of wired as well as wireless network functions and protocols (e.g., routing algorithms, TCP, UDP) can be done using NS2. In general, NS2 provides users with a way of specifying such network protocols and simulating their corresponding behaviors.

WRITING TCL SCRIPT-

Create a simulator object. This is done with the command

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

Opening 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 the simulator object is told that above is created 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"
```

The last line finally starts the simulation.

```
$ns run
```

SETTING UP OF NODES AND LINKS-

It is a very simple topology with two nodes that are connected by a link. The following two lines define the two nodes.

```
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
```

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. This looks like the fig. 2.1 below.

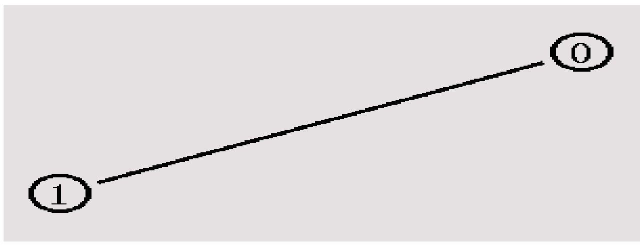


Fig 2.1

SENDING DATA-

The next step is to send some data from node n0 to node n1. In ns, data is always being sent from one 'agent' to another. So the next step is to create an agent object that sends data from node n0, and another agent object that receives the data on node n1.

```
#Create a UDP agent and attach it to node n0
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
# Create a CBR traffic source and attach it to udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0
```

These lines create a UDP agent and attach it to the node n0, then attach a CBR traffic generatot to the UDP agent. CBR stands for 'constant bit rate'. Line 7 and 8 should be self-explaining. The packetSize is being set to 500 bytes and a packet will be sent every 0.005 seconds (i.e. 200 packets per second).

The next lines create a Null agent which acts as traffic sink and attach it to node n1.

```
set null0 [new Agent/Null]
$ns attach-agent $n1 $null0
```

Now the two agents have to be connected with each other.

```
$ns connect $udp0 $null0
```

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"
```

Now we can save the file and start the simulation again. On clicking 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. It looks like the fig. 2.2 below.

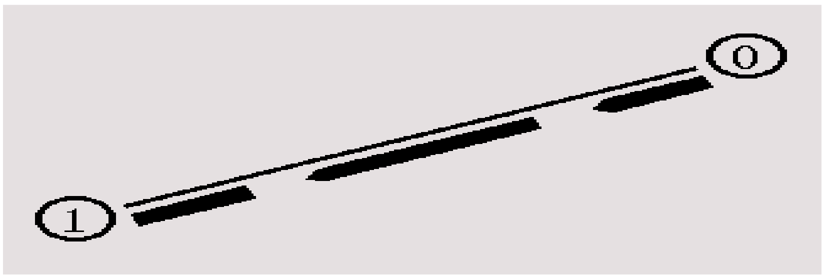


Fig. 2.2

CREATING DIFFERENT TOPOLOGIES-

A SIMPLE TOPOLOGY-#Create four nodes

```
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
```

The following piece of Tcl code creates three duplex links between the nodes.

```
$ns duplex-link $n0 $n2 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n3 $n2 1Mb 10ms DropTail
```

To have some more control over the layout. Add the next three lines.

```
$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link-op $n2 $n3 orient right
```

It looks like fig. 2.3 below.

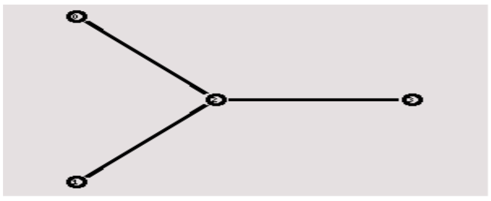
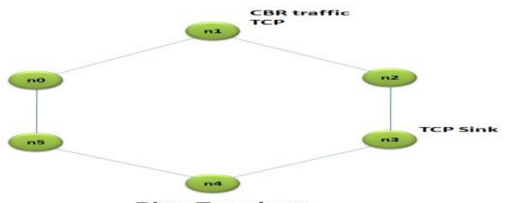


Fig. 2.3

RING TOPOLOGY-#Create four nodes

```
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
#Create links between the nodes $ns duplex-link $n0 $n1 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n2 $n3 1Mb 10ms DropTail
$ns duplex-link $n3 $n4 1Mb 10ms DropTail
$ns duplex-link $n4 $n5 1Mb 10ms DropTail
$ns duplex-link $n5 $n0 1Mb 10ms DropTail
$ns duplex-link-op $n0 $n1 orient right-up
$ns duplex-link-op $n1 $n2 orient right-down
$ns duplex-link-op $n2 $n3 orient down
$ns duplex-link-op $n3 $n4 orient left-down
$ns duplex-link-op $n4 $n5 orient left-up
$ns duplex-link-op $n5 $n0 orient up
```



Ring Topology

EXPERIMENT-20

AIM: Given a IP address in binary format, identify the class to which it belongs.

THEORY:

Class A address: Class A addresses always have the first bit of their IP addresses set to "0". Since Class A networks have an 8-bit network mask, the use of a leading zero leaves only 7 bits for the network portion of the address, allowing for a maximum of 128 possible network numbers, ranging from 0.0.0.0 – 127.0.0.0. Number 127.x.x.x is reserved for loopback, used for internal testing on the local machine.

Class B address: Class B addresses always have the first bit set to "1" and their second bit set to "0". Since Class B addresses have a 16-bit network mask, the use of a leading "10" bit-pattern leaves 14 bits for the network portion of the address, allowing for a maximum of 16,384 networks, ranging from 128.0.0.0 – 181.255.0.0.

Class C address: Class C addresses have their first two bits set to "1" and their third bit set to "0". Since Class C addresses have a 24-bit network mask, this leaves 21 bits for the network portion of the address, allowing for a maximum of 2,097,152 network addresses, ranging from 192.0.0.0 – 223.255.255.0.

Class D address: Class D addresses are used for multicasting applications. Class D addresses have their first three bits set to "1" and their fourth bit set to "0". Class D addresses are 32-bit network addresses, meaning that all the values within the range of 224.0.0.0 – 239.255.255.255 are used to uniquely identify multicast groups.

Class E address: Class E addresses are defined as experimental and are reserved for future testing purposes. They have never been documented or utilized in a standard way. Class E has its highest bit order set to **1-1-1-1.**
PROGRAM:

Experiment 21

Aim: Implementation of CSMA/CD in C.

Theory:- Carrier sense multiple access with collision detection (CSMA/CD) is a media access control method used most notably in early Ethernet technology for local area networking. It uses a carrier sensing scheme in which a transmitting station detects collisions by sensing transmissions from other stations while transmitting a frame. When this collision condition is detected, the station stops transmitting that frame, transmits a jam signal, and then waits for a random time interval before trying to resend the frame
Code:-

Experiment 22

Aim:-To implement Shortest path routing- Dijkstra's algorithm.

Theory:- Dijkstra's algorithm is an algorithm for finding the shortest paths between nodes in a graph, which may represent, for example, road networks. The detailed steps used in Dijkstra's algorithm to find the shortest path from a single source vertex to all other vertices in the given graph.

Algorithm

1) Create a set *sptSet* (shortest path tree set) that keeps track of vertices included in shortest path tree, i.e., whose minimum distance from source is calculated and finalized. Initially, this set is empty.

2) Assign a distance value to all vertices in the input graph. Initialize all distance values as INFINITE. Assign distance value as 0 for the source vertex so that it is picked first.

3) While *sptSet* doesn't include all vertices

….**a)** Pick a vertex u which is not there in *sptSet*and has minimum distance value.

….**b)** Include u to *sptSet*.

….**c)** Update distance value of all adjacent vertices of u. To update the distance values, iterate through all adjacent vertices. For every adjacent vertex v, if sum of distance value of u (from source) and weight of edge u-v, is less than the distance value of v, then update the distance value of v.

C Code:-

Experiment 23

Aim:-Implement Distance Vector Routing.

Theory: - Distance-vector routing protocol is one of the two major classes of intra domain routing protocols, the other major class being the link-state protocol. Distance-vector routing protocols use the Bellman–Ford algorithm, Ford–Fulkerson algorithm to calculate paths.

C Code:-

Experiment 24

AIM:- Introduction to TCP network programming with sockets.

Theory:- /*

In this program we illustrate the use of Berkeley sockets for interprocess communication across the network. We show the communication between a server process and a client process.

Since many server processes may be running in a system, we identify the desired server process by a "port number". Standard server processes have a worldwide unique port number associated with it. For example, the port number of SMTP (the sendmail process) is 25. To see a list of server processes and their port numbers see the file /etc/services

In this program, we choose port number 6000 for our server process. Here we shall demonstrate TCP connections only. For details and for other types of connections see:

Unix Network Programming

-- W. Richard Stevens, Prentice Hall India.

To create a TCP server process, we first need to open a "socket" using the socket() system call. This is similar to opening a file, and returns a socket descriptor. The socket is then bound to the desired port number. After this the process waits to "accept" client connections.

```
#include <stdio.h>
#include <sys/types.h>
/* The following three files must be included for network programming */
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
/* THE SERVER PROCESS */
/* Compile this program with cc server.c -o server and then execute it as ./server */
main()
{
int sockfd, newsockfd ; /* Socket descriptors */
int clilen;
struct sockaddr_in cli_addr, serv_addr;
int i;
char buf[100]; /* We will use this buffer for communication */
/* The following system call opens a socket. The first parameter
indicates the family of the protocol to be followed. For internet
protocols we use AF_INET. For TCP sockets the second parameter
is SOCK_STREAM. The third parameter is set to 0 for user
applications.
*/
```

```
if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
printf("Cannot create socket\n");
exit(0);
}
/* The structure "sockaddr_in" is defined in <netinet/in.h> for the
internet family of protocols. This has three main fields. The
field "sin_family" specifies the family and is therefore AF_INET
for the internet family. The field "sin_addr" specifies the
internet address of the server. This field is set to INADDR_ANY
for machines having a single IP address. The field "sin_port"
specifies the port number of the server.
*/
serv_addr.sin_family = AF_INET;
serv_addr.sin_addr.s_addr = INADDR_ANY;
serv_addr.sin_port = htons(6000);
/* With the information provided in serv_addr, we associate the server
with its port using the bind() system call.
*/
if (bind(sockfd, (struct sockaddr *) &serv_addr,
sizeof(serv_addr)) < 0) {
printf("Unable to bind local address\n");
exit(0);
}
listen(sockfd, 5); /* This specifies that up to 5 concurrent client
requests will be queued up while the system is
executing the "accept" system call below.
*/
/* In this program we are illustrating an iterative server -- one
which handles client connections one by one.i.e., no concurrency.
The accept() system call returns a new socket descriptor
which is used for communication with the server. After the
communication is over, the process comes back to wait again on
the original socket descriptor.
*/
while (1) {
/* The accept() system call accepts a client connection.
It blocks the server until a client request comes.
The accept() system call fills up the client's details
in a struct sockaddr which is passed as a parameter.
The length of the structure is noted in clilen. Note
that the new socket descriptor returned by the accept()
system call is stored in "newsockfd".
*/
clilen = sizeof(cli_addr);
newsockfd = accept(sockfd, (struct sockaddr *) &cli_addr,
&clilen) ;
if (newsockfd < 0) {
printf("Accept error\n");
exit(0);
}
/* We initialize the buffer, copy the message to it,
and send the message to the client.
*/
strcpy(buf,"Message from server");
send(newsockfd, buf, strlen(buf) + 1, 0);
/* We again initialize the buffer, and receive a
message from the client.
*/
for(i=0; i < 100; i++) buf[i] = '\0';
recv(newsockfd, buf, 100, 0);
printf("%s\n", buf);
close(newsockfd);
}
}

*******************************************************************
****************
/* THE CLIENT PROCESS

Please read the file server.c before you read this file. To run this,
you must first change the IP address specified in the line:
serv_addr.sin_addr.s_addr = inet_addr("144.16.202.221");
to the IP-address of the machine where you are running the server.
*/

#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
main()
{
int sockfd ;
struct sockaddr_in serv_addr;
int i;
char buf[100];
/* Opening a socket is exactly similar to the server process */
if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
printf("Unable to create socket\n");
exit(0);
}
/* Recall that we specified INADDR_ANY when we specified the server
address in the server. Since the client can run on a different
machine, we must specify the IP address of the server.

TO RUN THIS CLIENT, YOU MUST CHANGE THE IP ADDRESS SPECIFIED
BELOW TO THE IP ADDRESS OF THE MACHINE WHERE YOU ARE
RUNNING
THE SERVER.   */
serv_addr.sin_family = AF_INET;
serv_addr.sin_addr.s_addr = inet_addr("144.16.202.221");
serv_addr.sin_port = htons(6000);
/* With the information specified in serv_addr, the connect()
system call establishes a connection with the server process.
*/
if ((connect(sockfd, (struct sockaddr *) &serv_addr,
sizeof(serv_addr))) < 0) {
printf("Unable to connect to server\n");
exit(0);
}
/* After connection, the client can send or receive messages.
However, please note that recv() will block when the
server is not sending and vice versa. Similarly send() will
block when the server is not receiving and vice versa. For
non-blocking modes, refer to the online man pages.
*/
for(i=0; i < 100; i++) buf[i] = '\0';
recv(sockfd, buf, 100, 0);
printf("%s\n", buf);
strcpy(buf,"Message from client");
send(sockfd, buf, strlen(buf) + 1, 0);
close(sockfd);
}
```

Experiment 25

Aim:- Introduction to UDP network programming with sockets.

Code:-

```
/****************************************
* UDP Iterative server: server_udp.c
****************************************/
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h> // Internet family of protocols
#include <arpa/inet.h>
#include <unistd.h>
#define PORT 50000
#define BUFFSIZE 1024
int main(){
int socDes ;
struct sockaddr_in thisAddr, thatAddr;
int dataBytes, thatAddrLen ;
char buff[BUFFSIZE] ;
if((socDes = socket(PF_INET, SOCK_DGRAM, 0)) == -1) {
perror("cannot create socket") ;
return 0 ;
}
thisAddr.sin_family = PF_INET ;
thisAddr.sin_addr.s_addr = INADDR_ANY ; // inet_addr("127.0.0.1") ;
// Converts to 32-bit number
thisAddr.sin_port = htons(PORT) ; // Port number - byte order
if((bind(socDes, (struct sockaddr *)&thisAddr, sizeof(thisAddr))) < 0) {
perror("cannot bind") ;
return 0 ;
}
thatAddrLen = sizeof(thatAddr) ;
if((dataBytes=recvfrom(socDes, buff, BUFFSIZE-1, 0,
(struct sockaddr *)&thatAddr, &thatAddrLen)) < 0) {
perror("cannot receive") ;
return 0 ;
}
buff[dataBytes] = '\0' ;
printf("Data received: %s\n", buff) ;
return 0 ;
}


/****************************************
* UDP Client : client_udp.c
****************************************/
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h> // Internet family of protocols
```

```
#include <arpa/inet.h>
#include <unistd.h>
#define SERVERIP "127.0.0.1"
#define PORT 50000
#define BUFFSIZE 1024
int main(){
int socDes ;
struct sockaddr_in thatAddr ;
char buff[BUFFSIZE] = "NIT DURGAPUR" ;
int dataBytes = 8, dataSent ;
if((socDes = socket(PF_INET, SOCK_DGRAM, 0)) == -1) {
perror("cannot create socket") ;
return 0 ;
}
thatAddr.sin_family = PF_INET ;
thatAddr.sin_addr.s_addr = inet_addr(SERVERIP) ; // Server IP
// Converts to 32-bit number
thatAddr.sin_port = htons(PORT) ; // Port number - byte order
if((dataSent = sendto(socDes, buff, dataBytes, 0,
(struct sockaddr *)&thatAddr, sizeof(thatAddr))) < 0) {
perror("cannot send") ;
return 0 ;
}
close(socDes) ;
return 0 ;
}
```

Experiment 26

Aim:- Design TCP iterative Client and Server application to reverse the given input sentence.

Pseudo code:

```
START
Client sends message to server using sent functions.
Server receives all the messages, server ignores all the consonants in the message.
All the vowels in the message are converted into upper case.
Server returns the entire message to clients (with toggled vowel cases).
END
```

For example: "This is a test and sample message." to server will be sent back to client as "ThIs Is A tEst And sAmplE mEssAgE."

When client closes the connection server should close the communication with that client (socket). And once again wait for new clients to connect. Server program never exits. Using fork function rewrite the programs, such that this server can handle multiple client .

PROGRAM

Experiment 27

AIM: Design TCP client and server application to transfer file

Pseudo code:

Server side Filer Transfer TCP Pseudo code:

```
START
Start the program.
Declare the variables and structure for the socket.
Create a socket using socket functions The socket is binded at the specified port.
 Using the object the port and address are declared.
After the binding is executed the file is specified.
Then the file is specified.
Execute the client program.
END
```

Client side File Transfer TCP Pseudo code:

```
 START
Start the program.
 Declare the variables and structure.
Socket is created and connects function is executed.
If the connection is successful then server sends the message.
The file name that is to be transferred is specified in the client side.
The content of the file is verified from the server side.
End
```

SERVER PROGRAM

Experiment 28

AIM: Design UDP Client and server application to reverse the given input sentence

Description: UDP provides a connectionless service as there need not be any long-term relationship between a UDP client and server.

The User Datagram Protocol

The TCP/IP protocol suite provides two transport protocols, the *User Datagram Protocol* (UDP) described in this chapter, and the *Transmission Control Protocol* (TCP).There are some fundamental differences between applications written using TCP versus those that use UDP. These are because of the differences in the two transport layers:

UDP is a connectionless, unreliable, datagram protocol, quite unlike the connection-oriented, reliable byte stream provided by TCP. UDP is less complex and easier to understand. The characteristics of UDP are given below.

End-to-end: UDP can identify a specific process running on a computer.

Connectionless: UDP follows the connectionless paradigm (see below). *Message-oriented:* Processes using UDP send and receive individual messages called *segments*.

Best-effort: UDP offers the same best-effort delivery as IP.

Arbitrary interaction: UDP allows processes to send to and receive from as many other processes as it chooses. *Operating system independent:* UDP identifies processes independently of the local operating system.

Pseudo code for SERVER

```
START
Define LOCAL_SERVER_PORT 1500
Define MAX_MSG 3000
 Declare structure variables for Server socket data
take character buffers to store data
create IPV4 socket by using socket system call
Initialize server socket
 if socket system call
 return -1
then perror
 socket
exit Call
memeset system call to set the no of bytes to the value cin the destination
Set server_addr.sin_family=AF_INET
 Set server_addr.sin_port=htons(50000)
Set server_addr.sin_addr.s_addr=htonl(INADDR_ANY)
 Call bzero system call to set the specified no of bytes to 0
 If bind system call returns -1
Then Perror unable to bind
 Exit
 End if
 bind local server port
 server infinite loop
receive message
reading file contents
reading data to msg
```

closing stream
print received message
Send data received from client again to client by reversing it
Close connection
end of server infinite loop

PROGRAM
SERVER PROGRAM

CLIENT PROGRAM

Experiment 29

AIM: Design UDP Client server to transfer a file
DESCRIPTION:
UDP Client and Server The UDP client and server are created with the help of **DatagramSocket** and **Datagram packet** classes. If the UDP protocol is used at transport, then the unit of data at the transport layer is called a **datagram** and and not a segment. In UDP, no connection is established. It is the responsibility of an application to encapsulate data in datagrams (using Datagram classes) before sending it. If TCP is used for sending data, then the data is written directly to the socket (client or server) and reaches there as a connection exists between them. The datagram sent by the application using UDP may or may not reach the UDP receiver.

PROGRAM
CLIENT PROGRAM
SERVER PROGRAM

Advanced Network Technologies

Experiment No 30

Aim - Basics of Network Simulation: Defining nodes, links, queues and topology in NS2

Experiment No 31

Aim - Simulating a LAN using Network Simulator 2

Theory: Nodes in LAN are interconnected using one of four basic configurations: Bus topology, Star topology, Ring topology, Mesh topology.The network simulator simulates the three levels related to local area network. They are:

Link layer protocols such as ARQ
MAC protocol : MAC protocols are of two types namely:
Contention based protocols.
Round robin protocols.
Physical Channel

Experiment 32

Aim - Measuring Network Performance

Theory-Network Performance Evaluation using NS2

How to evaluate performance of a network by simulating it with ns2.
Choose and generate a network topology to be used throughout the simulation. This could be a wired network, in which case the topology remains fixed. However, for a wireless network with mobile nodes the topology would change with time, or randomly.
Once the topology has been generated, traffic source(s) and destination(s) are fixed. Assign suitable traffic sources to the source nodes, and traffic sinks to the destination nodes.
Some of the parameters that can be used for comparative study of performance of the network are: link bandwidth, propagation delay, node queue type.

Experiment 33

Aim - Simulating a Mobile Adhoc Network

Theory - Simulating the MANET protocols using AODV and DSR with the network simulator NS2

Mobile Adhoc NETwork (MANET)

MANET is a self-configuring network of mobile devices connected by wireless links. Each devices moves indepedently in any direction. Each node acts as a router.Some devices will detach from some devices in that area and attached or make link with other devices.'
A typical MANET is shown in the figure-01 below

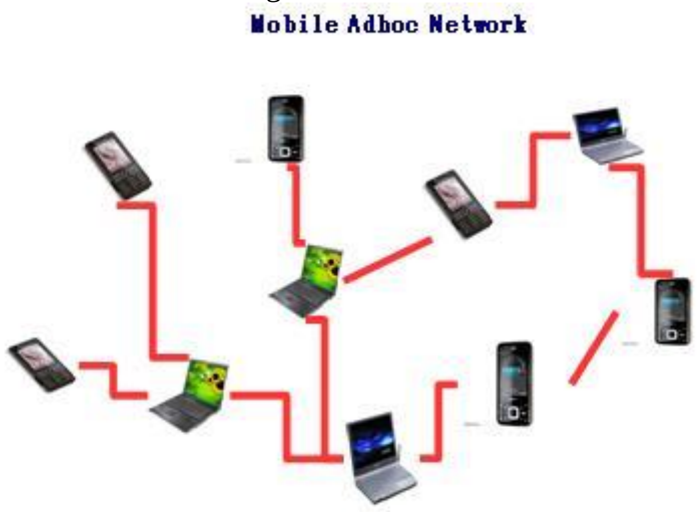


Figure-01: Mobile Adhoc Network MANETs

Routing

Routing is selecting a path or route in a network for forwarding packets. The objective of routing packets in a network is to determine the best possible path in terms of minimizing the number of hops (path length), delay, packet loss, cost etc.

Routing in MANET

MANETs are formed dynamically by collecting arbitrary wireless mobile nodes, without use of existing network infrastructure. So routing in MANET is different from traditional routing. In MANET each node acts as both host and router. The nodes transmit and receive their own packets. The nodes also take part in forwarding packets for other nodes. Therefore MANET provides limited physical security as compared to the traditional network.

Experiment 34

Aim - Simulating Routing protocols for MANET in NS2.

Theory-Routing protocols for a MANET can be classified as:

Proactive (table-driven) : DSDV, OLSR etc.
Reactive (on-demand): AODV, DSR etc.

Hybrid: ZRP

Proactive routing protocols determine path in advance and periodically exchange routing data to maintain the path. Reactive routing protocols, on the other hand, determine a route to some destination node only when it is required to send some data to that node. If at any time a path fails, an alternative path is determined again. Hybrid routing takes the advantages of both table driven and on-demand algorithms.

Experiment 35

Aim - Simulating Destination-Sequenced Distance-Vector(DSDV) algorithm in NS2.
Theory-The procedure for DSDV [1] is **:**

Each mobile node maintains a routing table with an entry of routing information from all its neighbors.
Each routing information in a routing table specifies

a)the destination identifier
b)the next hop on the route to the destination
c)the distance(in terms of hops) to the destination
d)a sequence number by monotonically increasing each time the node sends an update message to its neighbors. A route will be replaced only when the destination sequence number is less than the new one or two routes have the same sequence number but one has a lower metric.After generating a new routing table, each node broadcasts this table to all its neighbors.Based on the recieved tables, each mobile node update thier tables, until routing information is stable.

Experiment 36

Aim - Simulating Dynamic source routing (DSR) in NS2

Theory-The DSR [v] protocol is composed of two mechanisms that work together to allow the discovery and maintenance of source routes in the ad hoc network:

1.**Route Discovery:** Route discovery is used only when source wants to send a packet to destination and does not know a route to destination.A mobile node A wants to send a packet to a destination node B, then obtain a source route to B.

2.**Route Maintenance** : Route Maintenance is the mechanism by which a source node A is able to detect, while using a source route to B. If the network topology has changed and the route is broken then the source route attempts to use any other route to destination if it exist or can invoke route discovery again to find a new route. Route Maintenance is used only when source is actually sending packets to destination.

Both Route Discovery and Route Maintenance each operate entirely on demand. When the destination node is reached, it returns a reply containing the route to the source node. The reply then travels in the reverse direction of the discovered route or on a path already known by the destination, to the source. The source node, on receiving the reply, will place the route in its route cache.