

LAB MANUAL

of

Computer Networks

Prepared By
Assistant Professor
Prof. Salunke M. D.

Savitribai Phule Pune University Third Year of Computer Engineering (2015 Course) 310248: Computer Networks Lab		
Teaching Scheme: PR: 02 Hours/Week	Credit 01	Examination Scheme: TW: 25 Marks PR: 50 Marks

Course Objectives:

- To establish communication among the computing nodes in P2P and Client-Server architecture
- Configure the computing nodes with understanding of protocols and technologies
- Use different communicating modes and standards for communication
- Use modern tools for network traffic analysis
- To learn network programming

Course Outcomes:

On completion of the course, student will be able to

- Demonstrate LAN and WAN protocol behavior using Modern Tools
- Analyze data flow between peer to peer in an IP network using Application, Transport and Network Layer Protocols
- Demonstrate basic configuration of switches and routers.
- Develop Client- Server architectures and prototypes by the means of correct standards and technology

	1. Ethernet 2. IP 3. TCP 4. UDP	
5	<p>Lab Assignment on Unit II: (Use C/C++)</p> <p>Write a program for Prof. Salunke M.D. section for 7/8 bits ASCII codes using Hamming Codes or CRC. Demonstrate the packets captured traces using Wireshark Packet Analyzer Tool for peer to peer mode. (50% students will perform Hamming Code and others will perform CRC)</p>	
6	<p>Lab Assignment on Unit II: (Use JAVA/PYTHON)</p> <p>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.</p>	
7	<p>Lab Assignment on Unit IV: (Use JAVA/PYTHON)</p> <p>Write a program to demonstrate subnetting and find the subnet masks.</p>	
8	<p>Lab Assignment on Unit VI: (Use JAVA/PYTHON)</p> <p>Write a program for DNS lookup. Given an IP address input, it should return URL and vice-versa.</p>	
Group B		
9	<p>Lab Assignment on Unit IV and Unit V: (Mandatory Assignment)</p> <p>Use network simulator NS2 to implement:</p> <ol style="list-style-type: none"> Monitoring traffic for the given topology Analysis of CSMA and Ethernet protocols Network Routing: Shortest path routing, AODV. Analysis of congestion control (TCP and UDP). 	
10	<p>Lab Assignment on Unit IV: (Mandatory Assignment)</p> <p>Configure RIP/OSPF/BGP using packet Tracer.</p>	
11	<p>Lab Assignment on Unit IV:</p> <p>Study of any network simulation tools -To create a network with three</p>	

	nodes and establish a TCP connection between node 0 and node 1 such that node 0 will send TCP packet to node 2 via node 1	
12	Lab Assignment on Unit V: (Use JAVA/PYTHON) Write a program using TCP sockets for wired network to implement a. Peer to Peer Chat b. Multiuser Chat	

Prof.Salunke M.D.

Subject Incharge

Prof. M. D. Salunke

HOD COMP

Assignment Group-A_1

Problem Definition:

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

1. Apparatus (Components):

RJ-45 connector, Crimping Tool, Twisted pair Cable(Cat6), Line Tester, HTTP Server (Apache) with Website pages of your Institute, Four Client Nodes with Wi-Fi Support, Wireshark Protocol Analyzer tool on all nodes, Layer-II Switch, Layer-III IP Switch, Wi-Fi Access Point.

2. Prerequisite:

1. Networking Components: Switch, Router, etc.
2. Linux Command: Ping
3. Wireshark Tool
4. IP Addressing

- Students will able to setup wired and Wi-Fi network
- Learn to setup wired and Wi-Fi office/organization network

4. Theory

Cable Preparation

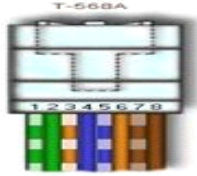
The cable will be constructed using either TIA/EIA T568A or T568B standards for Ethernet, which determines the color wire to be used on each pin.

Straight-through patch cables are normally used to connect a host directly to a hub or switch or to a wall plate in an office area. With a straight-through cable, the color of wire used by pin 1 on one end is the same color used by pin 1 on the other cable end, and similarly for the remaining seven pins.

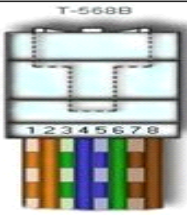
With a **crossover cable** the second and third pairs on the RJ-45 connector at one end of the cable are reversed at the other end. The pin-outs for the cable are the T568A standard on one end and the T568B standard on the other end. Crossover cables are normally used to connect hubs and switches or can be used to directly connect two hosts to create a simple network.

TIA/EIA 568A and 568B Wiring Standards

Pin Diagram TIA/EIA 568-A					
PIN	F()	Pair	Polarity	COLOR	A
1	Rx	3	Rx+	Green/White	G
2	Rx	3	Rx-	Green	G
3	Tx	2	Tx+	Orange/White	O
4	-	1	Not Used	Blue	B
5	-	1	Not Used	Blue/White	B
6	Tx	2	Tx-	Orange	O
7	-	4	Not Used	Brown/White	B
8	-	4	Not Used	Brown	B



Pin Diagram TIA/EIA 568-B					
PIN	F()	Pair	Polarity	COLOR	A
1	Tx	2	Tx+	Orange/White	O
2	Tx	2	Tx-	Orange	O
3	Rx	3	Rx+	Green/White	G
4	-	1	Not Used	Blue	B
5	-	1	Not Used	Blue/White	B
6	Rx	3	Rx-	Green	G
7	-	4	Not Used	Brown/White	B
8	-	4	Not Used	Brown	B



Prepare and test an Ethernet straight-through and Crossover patch cable

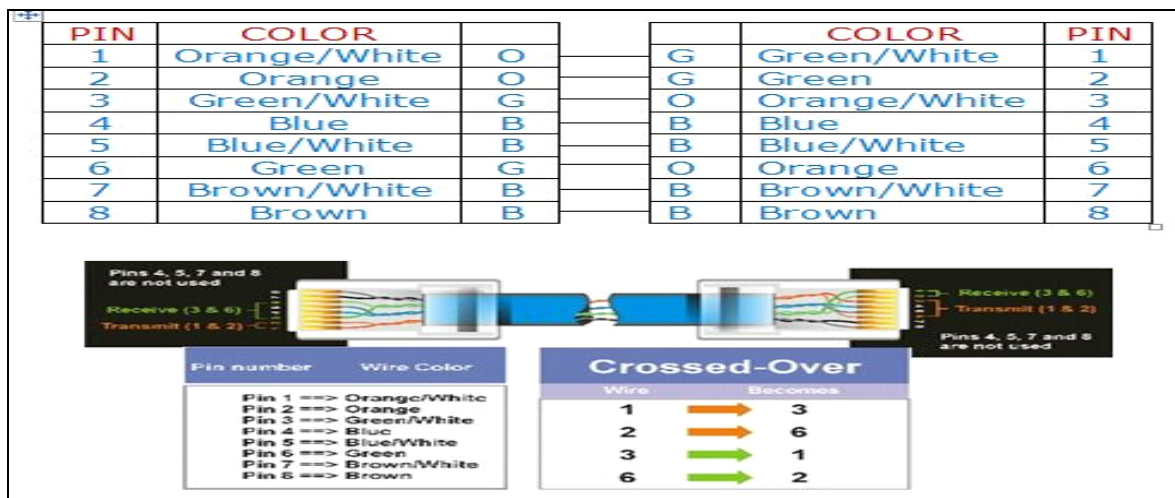
Step 1: Obtain and prepare the cable

- Determine the length of cable required. This could be the distance from a computer to a switch or between a device and an RJ-45 outlet jack.
- Using wire strippers, remove 5.08 cm (2 in.) of the cable jacket from both ends of the cable.

Pin Diagram TIA/EIA 568-B for Straight-Through Cabling



Pin Diagram EIA/TIA 568B for Cross-Over Cabling



Step 2: Prepare and insert the wires

- Determine which wiring standard will be used. Circle the standard. [T568A | T568B] and locate the correct table or figure from the “Wire Diagrams” based on the wiring standard used.
- Spread the cable pairs and arrange them roughly in the desired order based on the standard chosen.
- Untwist a short length of the pairs and arrange them in the exact order needed by the standard moving left to right starting with pin 1.

- It is very important to untwist as little as possible. The twists are important because they provide noise cancellation
- Straighten and flatten the wires between your thumb and forefinger. Ensure the cable wires are still in the correct order as the standard.
- Cut the cable in a straight line to within 1.25 to 1.9 cm (1/2 to 3/4 in.) from the edge of the cable jacket. If it is longer than this, the cable will be susceptible to crosstalk (the interference of bits from one wire with an adjacent wire).
- The key (the prong that sticks out from the RJ-45 connector) should be on the underside pointing downward when inserting the wires. Ensure the wires are in order from left to right starting with pin 1. Insert the wires firmly into the RJ-45 connector until all wires are pushed as far as possible into the connector

Step 3: Inspect, crimp, and re-inspect

- Visually inspect the cable and ensure the right color codes are connected to the correct pin numbers.
- Visually inspect the end of the connector. The eight wires should be pressed firmly against the end of the RJ-45 connector. Some of the cable jacket should be inside the first portion of the connector. This provides strain relief for the cable. If the cable jacket is not far enough inside the connector, it may eventually cause the cable to fail.
- If everything is correctly aligned and inserted properly, place the RJ-45 connector and cable into the crimper. The crimper will push two plungers down on the RJ-45 connector.
- Visually re-inspect the connector. If improperly installed, cut the end off and repeat the process.

Step 4: Terminate the other cable end

- Use the previously described steps to attach an RJ-45 connector to the other end of the cable.
- Visually re-inspect the connector. If improperly installed, cut the end off and repeat the process.

Step 5: Test the cable

- Use the cable to connect a PC to a network.
- Visually check the LED status lights on the NIC card. If they are on (usually green or amber) the cable is functional.
- On the PC, open the command prompt.
- Type `ifconfig`
- Write down the default gateway IP address.
- Or you can use line tester to test the prepared cable.

Network Devices

1. Repeater:

Functioning at Physical Layer. A repeater is an electronic device that receives a signal and retransmits it at a higher level and/or higher power, or onto the other side of an obstruction, so that the signal can cover longer distances. Repeater have two ports ,so cannot be use to connect for more than two devices.

2. Hub:

An Ethernet hub, active hub, network hub, repeater hub, hub or concentrator is a device for connecting multiple twisted pair or fiber optic Ethernet devices together and making them act as a single network segment. Hubs work at the physical layer (layer 1) of the OSI model. The device is a form of multiport repeater. Repeater hubs also participate in collision detection, forwarding a jam signal to all ports if it detects a collision.

3. Switch:

A network switch or switching hub is a computer networking device that connects network segments. The term commonly refers to a network bridge that processes and routes data at the data link layer (layer 2) of the OSI model. Switches that additionally process data at the network layer (layer 3 and above) are often referred to as Layer 3 switches or multilayer switches.

4. Bridge:

A network bridge connects multiple network segments at the data link layer (Layer 2) of the OSI model. In Ethernet networks, the term bridge formally means a device that behaves according to the IEEE 802.1D standard. A bridge and switch are very much alike; a switch being a bridge with numerous ports. Switch or Layer 2 switch is often used interchangeably with bridge. Bridges can analyze incoming data packets to determine if the bridge is able to send the given packet to another segment of the network.

5. Router:

A router is an electronic device that interconnects two or more computer networks, and selectively interchanges packets of data between them. Each data packet contains address information that a router can use to determine if the source and destination are on the same network, or if the data packet must be transferred from one network to another. Where multiple routers are used in a large collection of interconnected networks, the routers exchange information about target system addresses, so that each router can build up a table showing the preferred paths between any two systems on the interconnected networks.

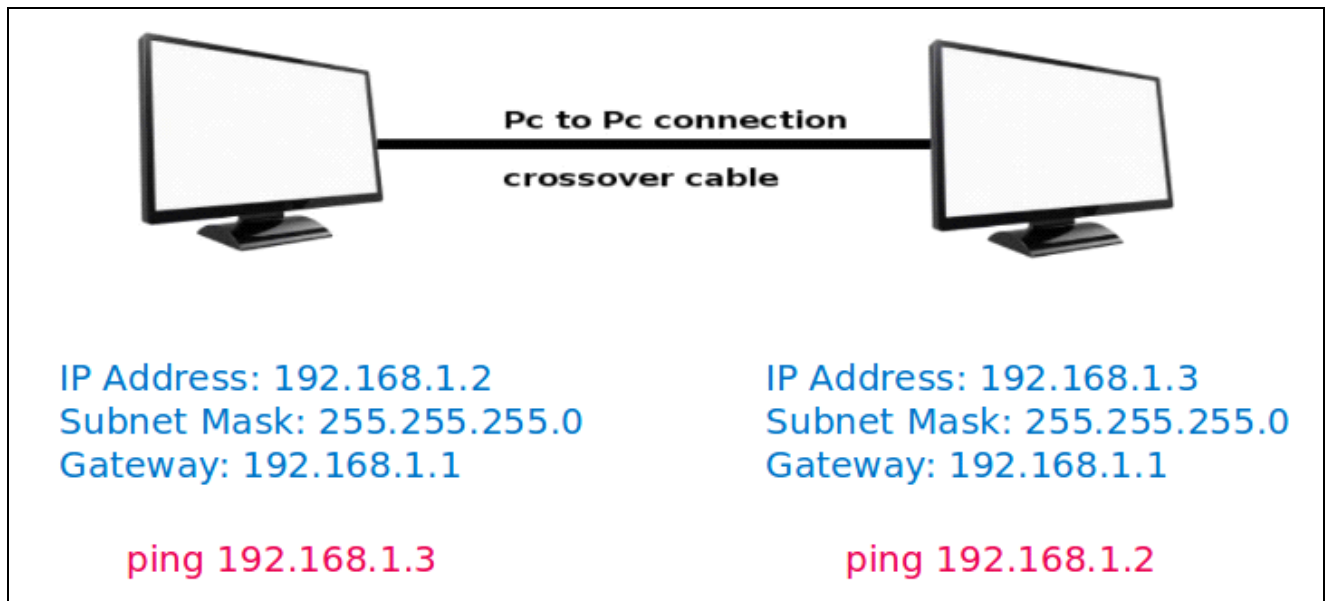
6. Gate Way:

In a communications network, a network node equipped for interfacing with another network that uses different protocols. A gateway may contain devices such as protocol translators, impedance matching devices, rate converters, fault isolators, or signal translators as necessary to provide system interoperability. It also requires the establishment of mutually acceptable administrative procedures between both networks. A protocol translation/mapping gateway

interconnects networks with different network protocol technologies by performing the required protocol.

Building and Testing of Wired Network

1. Crossover Cable



Connect two machines using crossover cable and configure it using ip address, subnet mask and gateway address as shown in figure. Ping from both the machines and capture ICMP packets in Wireshark tool.

2. Setting Up LAN using Straight-Through Cable

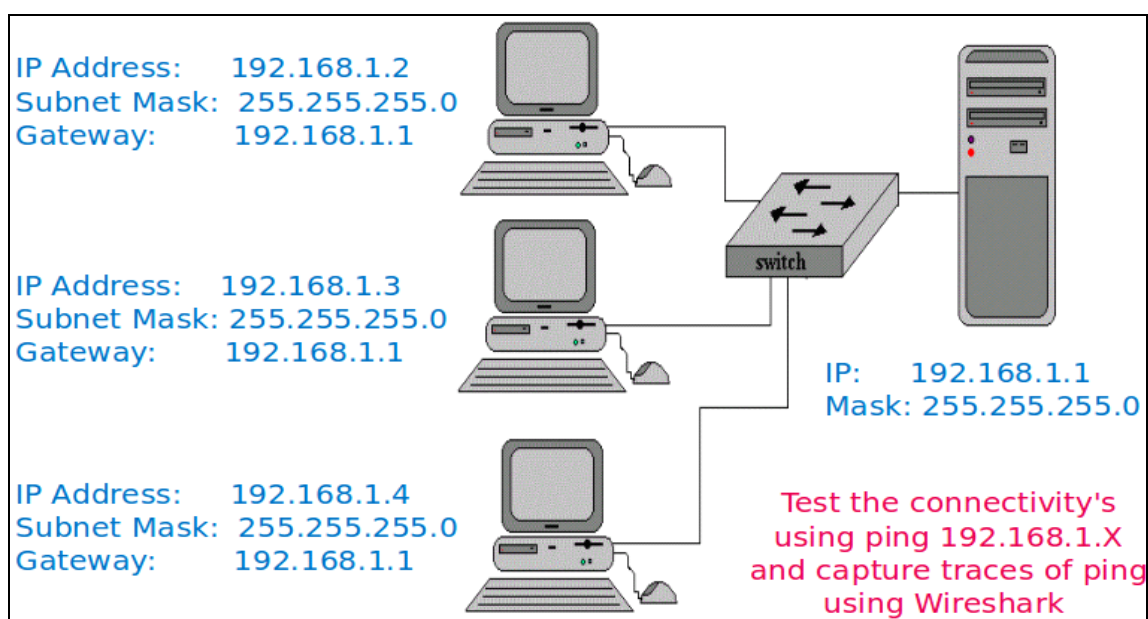


Figure 1

Connect four machines using Straight-Through cable to switch and router and then configure all using ip address, subnet mask and gateway address as shown in figure. Ping all the machines and capture ICMP packets in Wireshark tool.

3. Testing Web Server over LAN

- Installation of Web Server – Apache2 or Tomcat7
- Install the server – `sudo apt-get install apache`
- Start web server - `/etc/init.d/apache2 start`
- Create the web page and store in `/var/www/http`
- Access the web pages from client machines 1/2/3

Test the web server by accessing web pages stored on server and capture the traces of http ,tcp, ip and Ethernet-II using Wireshark.

Part B: Repeat the same process for wireless network. For that use wifi supported nodes and access point.

Conclusion:

Hence we have designed wired and wireless LAN using crossover and straight-through cable, and captured the ICMP, HTTP packets in Wireshark.

Assignment Group-A_2

Problem Definition:

Write a program using TCP socket for wired network for following (Use C/C++)

- 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.

1. Prerequisite:

1. Transport Layer: Roles, Protocols(TCP,UDP)
2. C/C++ Programming Syntax
3. Wireshark Tool

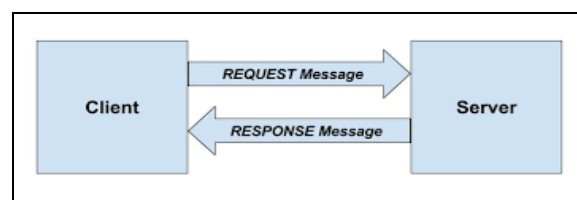
2. Learning Objectives:

- Students will able to understand socket programming.
- Students will able to design networking applications using TCP protocol.

3. Theory

Client-Server Model

Network applications can be divided into two process: a Client and a Server, with a communication link joining the two processes.



Normally, from Client side it is one-one connection. From the Server Side, it is many-one connection. The standard model for network applications is the Client-Server model. A

Server is a process that is waiting to be contacted by a Client process so that server can do something for the client. Typical BSD Sockets applications consist of two separate application level processes; one process (the client) requests a connection and the other process (the server) accepts it.

Client-Server Model Using TCP

TCP Clients sends request to server and server will receives the request and response with acknowledgement. Every time client communicates with server and receive response from it.

Algorithm to create client and server process is as below.

ALGORITHM:

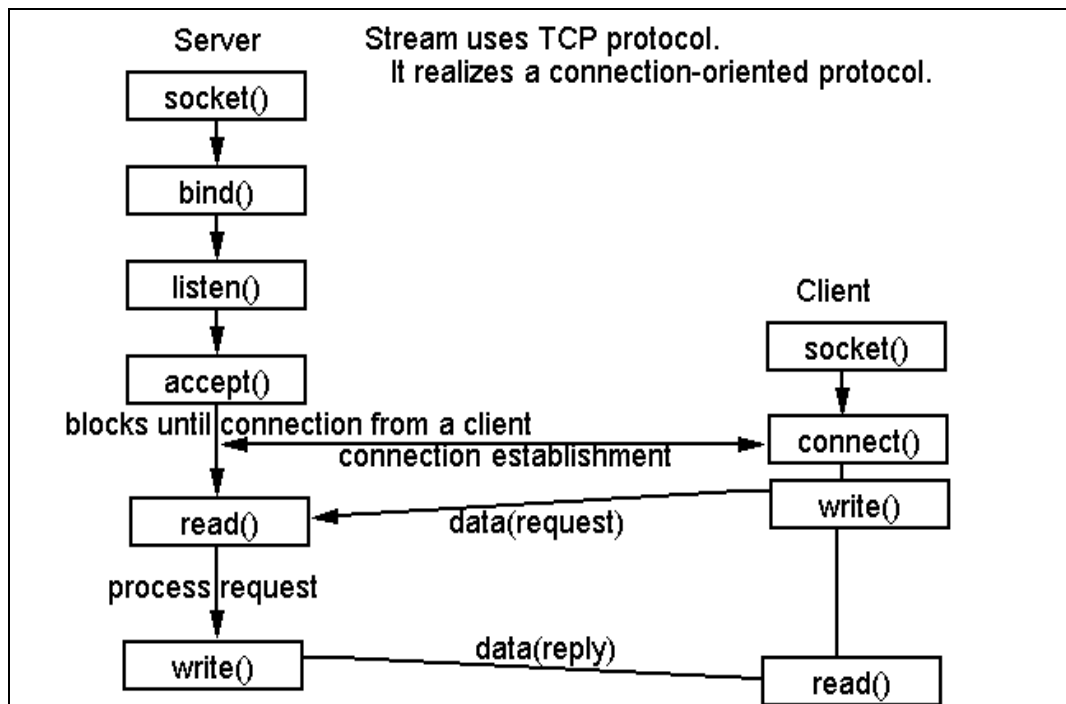
Server

1. Create a server socket and bind it to port
2. Listen for new connection and when a connection arrives, accept it
3. Read Client's message and display it
4. Get a message from user and send it to client
5. Close the server socket
6. Stop

Client

1. Create a client socket and connect it to the server's port number
2. Get a message from user and send it to server
3. Read server's response and display it
4. Close the client socket
5. Stop

Socket functions for TCP client/server in Connection-oriented Scenario



Elementary Socket System Calls

1. socket()	<p>socket() System Call: Creates an end point for communication and returns a descriptor.</p> <pre>#include <sys/socket.h> #include <sys/types.h> int socket (int Address Family, int <u>Type</u>, int <u>Protocol</u>);</pre> <p><u>Return Values:</u> Upon successful completion, the socket subroutine returns an integer (the socket descriptor). It returns -1 on error.</p>
2. bind()	<p>bind() System call: Binds a name to a socket.</p> <p>Description: The bind subroutine assigns a Name parameter to an unnamed socket. It assigns a local protocol address to a socket.</p> <pre>#include <sys/socket.h> int bind (int sockfd, struct sockaddr *myaddr, int addrlen);</pre>

	<p><u>Return Values:</u> Upon successful completion, the bind subroutine returns a value of 0. Otherwise, it returns a value of -1 to the calling program.</p>
3. connect()	<p>connect() System call: The connect function is used by a TCP client to establish a connection with a TCP server.</p> <p>#include <sys/socket.h></p> <p>int connect(int sockfd, struct sockaddr *servaddr, int addrlen);</p> <p><u>Return Values:</u> Upon successful completion, the connect subroutine returns a value of 0. Otherwise, it returns a value of -1 to the calling program.</p>
4. listen()	<p>listen() System call: This system call is used by a connection-oriented server to indicate that it is willing to receive connections.</p> <p>#include <sys/socket.h></p> <p>int listen (int sockfd, int backlog);</p> <p><u>Return values:</u> Returns 0 if OK, -1 on error</p>
5. accept()	<p>accept() System call: The actual connection from some client process is waited for by having the server execute the accept system call.</p> <p>#include <sys/socket.h></p> <p>int accept (int sockfd, struct sockaddr *cliaddr, int *addrlen);</p> <p><u>Return Values:</u> This system call returns up to three values: an integer return code that is either a new socket descriptor or an error indication, the protocol address of the client process (through the cliaddr pointer), and the size of this address (through the addrlen pointer).</p>
6. read() and write()	<p>read() and write() System call: - read/write from a file descriptor</p> <p>#include <unistd.h></p> <p>ssize_t read(int fd, void *buf, size_t count);</p> <p>ssize_t write(int fd, const void *buf, size_t count);</p> <p><u>Return Values:</u> On success, the number of bytes read or written is returned (zero indicates nothing was written/read). On error, -1 is returned.</p>
7. Send(), sendto(), recv() and recvfrom()	<p>Send(), sendto(), recv() and recvfrom() system calls:</p> <p>These system calls are similar to the standard read and write functions, but one additional argument is required.</p>

and recvfrom()	<pre>#include <sys/socket.h> int send(int sockfd, char *buff, int nbytes, int flags); int sendto(int sockfd, char void *buff, int nbytes, int flags, struct sockaddr *to, int addrlen); int recv(int sockfd, char *buff, int nbytes, int flags); int recvfrom(int sockfd, char *buff, int nbytes, int flags, struct sockaddr *from, int *addrlen);</pre> <p>The first three arguments, sockfd, buff and nbytes are the same as the first three arguments to read and write. The flags argument is either 0 or is formed by logically OR'ing one or more of the constants.</p> <p><u>Return Values:</u> All four system calls return the length of the data that was written or read as the value of the function. Otherwise it returns, -1 on error.</p>
8. Close()	<p>Close() system call: The normal Unix close function is also used to close a socket and terminate a TCP connection.</p> <pre>#include <unistd.h> int close (int sockfd);</pre>

Testing

1. Run Wireshark tool
2. Run client and server program
3. Exchange message
4. Capture TCP packets in Wireshark

Conclusion:

Hence we have studied TCP Socket Programming in C for echo message, file transfer, arithmetic and trigonometric calculator and captured TCP packets in Wireshark tool.

Assignment Group-A_3

Problem Definition:

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.

1. Prerequisite:

1. Transport Layer: Roles, Protocols(TCP,UDP)
2. C/C++ Programming Syntax
3. Wireshark Tool

2. Learning Objectives:

- Students will able to understand socket programming.
- Students will able to design networking applications using UDP protocol.

3. Theory

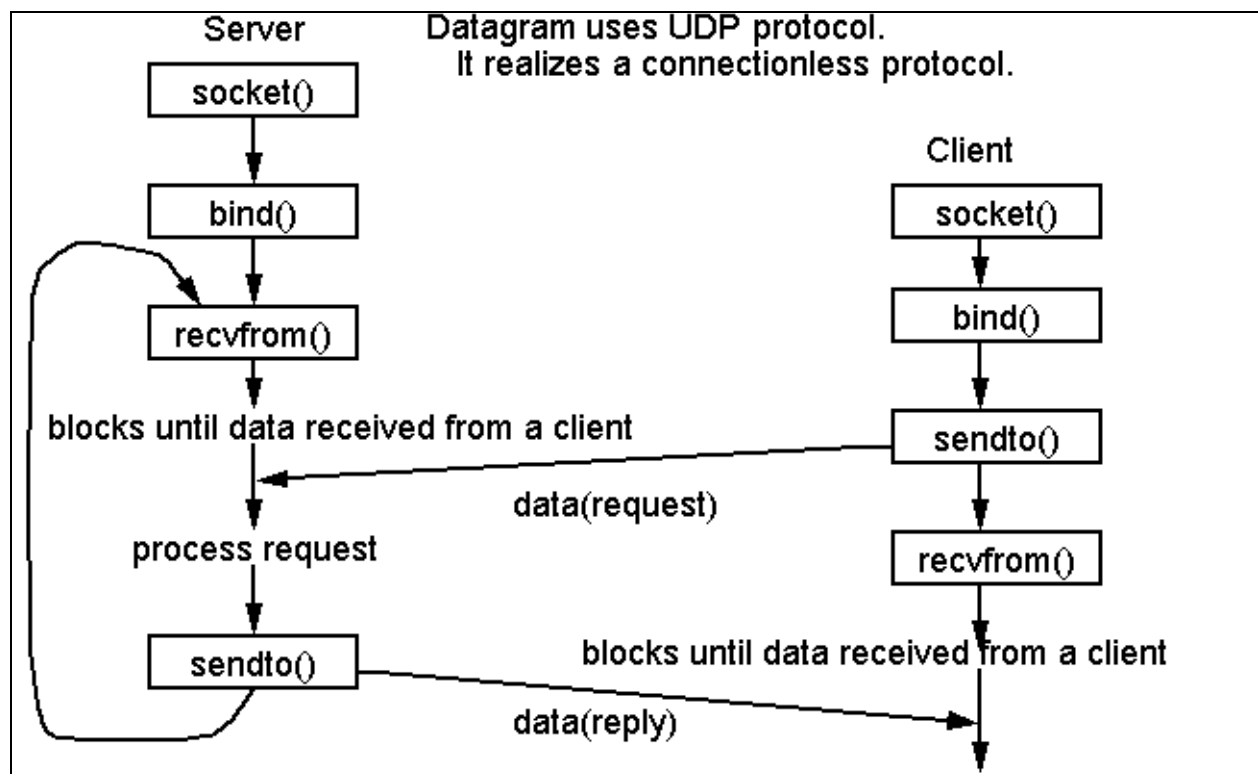
Server-Client file transfer using UDP

Server:

- Include appropriate header files.
- Create a UDP Socket using socket() system call.
- Fill in the socket address structure (with server information)
- Specify the port where the service will be defined to be used by client.
- Bind the address and port using bind() system call
- Receive a file name of text, audio or video from the Client using recvfrom() system call.
- Sends file to client using sendto() system call.
- Close the server socket
- Stop

Client:

- Include appropriate header files.
- Create a UDP Socket.
- Fill in the socket address structure (with server information)
- Specify the port of the Server, where it is providing service
- Send a file name to the server using sendto() system call.
- Receive a file from the Server using recvfrom() system call.
- Close the client socket
- Stop

Socket functions for UDP client/server in Connectionless Scenario

Testing

1. Run Wireshark tool
2. Run client and server program
3. Send and receive file
4. Capture UDP packets in Wireshark

Conclusion:

Hence we have studied UDP Socket Programming in C for file transmission of text, audio and video and captured UDP packets in Wireshark tool.

Assignment Group-A_4

Problem Definition:

Write a program to analyze following packet formats captured through Wireshark for wired network.

1. Ethernet 2. IP 3. TCP 4. UDP

1. Prerequisite:

1. Transport Layer: Roles, Protocols(TCP,UDP)
2. Network Layer: Roles, Protocols(IP)
3. C/C++ Programming Syntax
4. Wireshark Tool

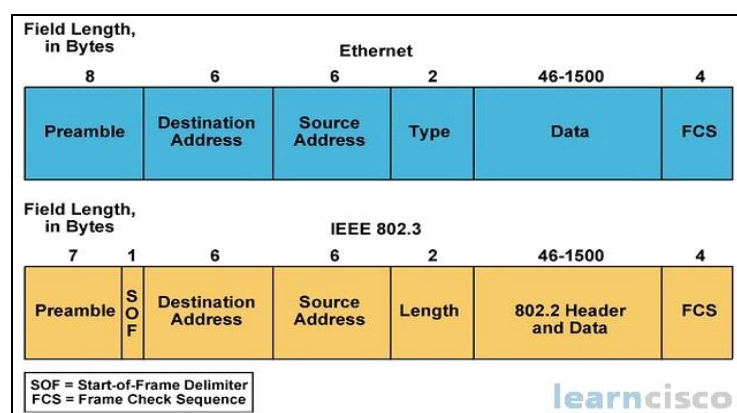
2. Learning Objectives:

- Students will able to understand Ethernet, IP, TCP and UDP packet formats
- Students will able to analyze packet formats of Ethernet, IP, TCP and UDP protocols

3. Theory

Header Formats:

1. Ethernet Header Format:



- The Preamble consists of seven bytes all of the form 10101010, and is used by the receiver to allow it to establish bit synchronization.
- The Start frame delimiter is a single byte, 10101011, which is a frame flag, indicating

the start of a frame.

- The MAC addresses used in 802.3 are always 48 bits long, although older versions of Ethernet used 16 bits. By convention, Ethernet addresses are usually quoted as a sequence of 6 bytes (in hexadecimal) with each byte quoted with its bits in reverse order (this curious arrangement is driven by the transmission order).
- The Length/EtherType field is the only one which differs between 802.3 and Ethernet II. In 802.3 it indicates the number of bytes of data in the frames payload, and can be anything from 0 to 1500 bytes. Frames must be at least 64 bytes long, not including the preamble, so, if the data field is shorter than 46 bytes, it must be compensated by the Pad field.
- The frame check sequence (FCS) is a four-octet cyclic redundancy check (CRC) that allows detection of corrupted data within the entire frame as received on the receiver side. The FCS value is computed as a function of the protected MAC frame fields: source and destination address, length/type field, MAC client data and padding.

2. IP Header Format:

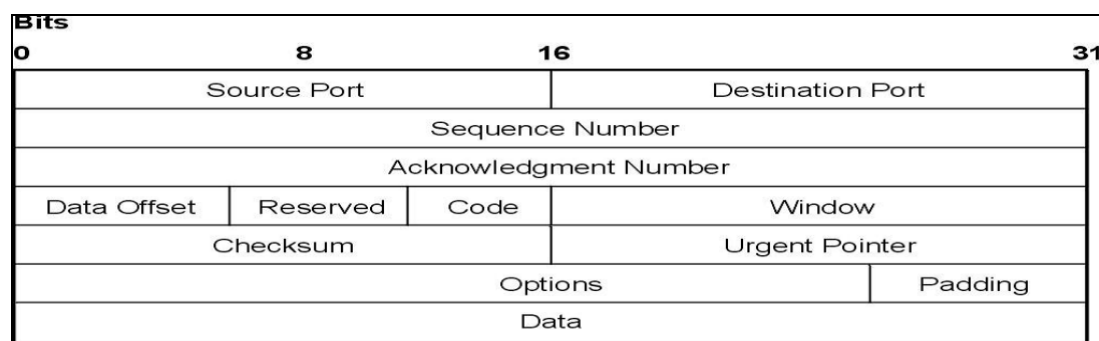
Bits					
0	4	8	16	19	31
Version	Length	Type of Service	Total Length		
Identification			Flags	Fragment Offset	
Time to Live	Protocol		Header Checksum		
Source Address					
Destination Address					
Options					
Data					

The fields in the IP header and their descriptions are

- **Version**—A 4-bit field that identifies the IP version being used. The current version is 4, and this version is referred to as IPv4.
- **Length**—A 4-bit field containing the length of the IP header in 32-bit increments. The minimum length of an IP header is 20 bytes, or five 32-bit increments. The maximum length of an IP header is 24 bytes, or six 32-bit increments. Therefore, the header length field should contain either 5 or 6.
- **Type of Service (ToS)**—The 8-bit ToS uses 3 bits for IP Precedence, 4 bits for ToS with the last bit not being used. The 4-bit ToS field, although defined, has never been used.
- **IP Precedence**— A 3-bit field used to identify the level of service a packet receives in the network.
- **Differentiated Services Code Point (DSCP)**—A 6-bit field used to identify the level of service a packet receives in the network. DSCP is a 3-bit expansion of IP precedence with the elimination of the ToS bits.

- **Total Length**—Specifies the length of the IP packet that includes the IP header and the user data. The length field is 2 bytes, so the maximum size of an IP packet is $2^{16} - 1$ or 65,535 bytes.
- **Identifier, Flags, and Fragment Offset**—As an IP packet moves through the Internet, it might need to cross a route that cannot handle the size of the packet. The packet will be divided, or fragmented, into smaller packets and reassembled later. These fields are used to fragment and reassemble packets.
- **Time to Live (TTL)**—It is possible for an IP packet to roam aimlessly around the Internet. If there is a routing problem or a routing loop, then you don't want packets to be forwarded forever. A routing loop is when a packet is continually routed through the same routers over and over. The TTL field is initially set to a number and decremented by every router that is passed through. When TTL reaches 0 the packet is discarded.
- **Protocol**—In the layered protocol model, the layer that determines which application the data is from or which application the data is for is indicated using the Protocol field. This field does not identify the application, but identifies a protocol that sits above the IP layer that is used for application identification.
- **Header Checksum**—A value calculated based on the contents of the IP header. Used to determine if any errors have been introduced during transmission.
- **Source IP Address**—32-bit IP address of the sender.
- **Destination IP Address**—32-bit IP address of the intended recipient.
- **Options and Padding**—A field that varies in length from 0 to a multiple of 32-bits. If the option values are not a multiple of 32-bits, 0s are added or padded to ensure this field contains a multiple of 32 bits.
-

3. TCP Header Format:



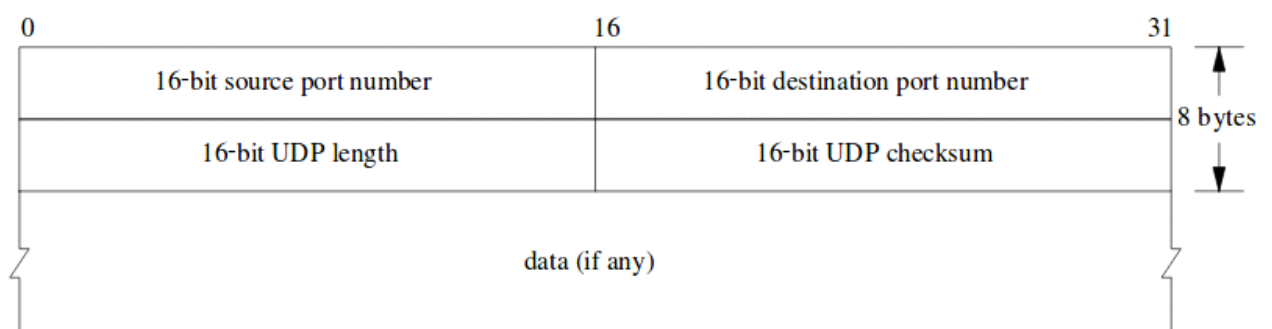
Each TCP header has ten required fields totaling 20 bytes (160 bits) in size. They can also optionally include an additional data section up to 40 bytes in size.

- Source and destination TCP port numbers are the communication endpoints for sending and receiving devices.
- Message senders use sequence numbers to mark the ordering of a group of messages. Both senders and receivers use the acknowledgment numbers field to communicate

the sequence numbers of messages that are either recently received or expected to be sent.

- The data offset field stores the total size of a TCP header in multiples of four bytes. A header not using the optional TCP field has a data offset of 5 (representing 20 bytes), while a header using the maximum-sized optional field has a data offset of 15 (representing 60 bytes).
- Reserved data in TCP headers always has a value of zero. This field serves the purpose of aligning the total header size as a multiple of four bytes.
- TCP uses a set of six standard and three extended control flags (each an individual bit representing on or off) to manage data flow in specific situations. One bit flag, for example, initiates TCP connection reset logic. The detailed operation of these fields goes beyond the scope of this article.
- TCP senders use a number called window size to regulate how much data they send to a receiver before requiring an acknowledgment in return. If the window size becomes too small, network data transfer will be unnecessarily slow, while if the window size becomes too large, the network link can become saturated (unusable for any other applications) or the receiver may not be able to process incoming data quickly enough (also resulting in slow performance).
- The checksum value inside a TCP header is generated by the protocol sender as a mathematical technique to help the receiver detect messages that are corrupted or tampered with.
- The urgent pointer field is often set to zero and ignored, but in conjunction with one of the control flags, it can be used as a data offset to mark a subset of a message as requiring priority processing.
- Usages of optional TCP data go beyond the scope of this article but include support for special acknowledgment and window scaling algorithms.

4. UDP Header Format:



Because UDP is significantly more limited in capability than TCP, its headers are much

smaller.

A UDP header contains 8 bytes, divided into the following four required fields:

1. Source port number (2 bytes)
2. Destination port number (2 bytes)
3. Length of data (2 bytes)
4. UDP checksum (2 bytes)

UDP inserts header fields into its message stream in the order listed above.

- Source and destination UDP port numbers are the communication endpoints for sending and receiving devices.
- The length field in UDP represents the total size of each datagram including both header and data. This field ranges in value from a minimum of 8 bytes (the required header size) to sizes above 65,000 bytes.
- Similar to TCP, a UDP checksum allows receivers to cross-check incoming data for any corrupted bits of the message.

Testing

1. Run Wireshark tool
2. Run client and server program
3. Send and receive file
4. Capture UDP packets in Wireshark

Conclusion:

Hence we have studied UDP Socket Programming in C for file transmission of text, audio and video and captured UDP packets in Wireshark tool.

Assignment Group-A_5

Problem Definition:

Write a program for error detection and correction for 7/8 bits ASCII codes using Hamming Codes or CRC. Demonstrate the packets captured traces using Wireshark Packet Analyzer Tool for peer to peer mode.(50% students will perform Hamming Code and others will perform CRC) (Use C/C++)

1. Prerequisite:

1. Data Link Layer: Roles, Protocols(Ethernet)
2. C/C++ Programming Syntax
3. Wireshark Tool

2. Learning Objectives:

- Students will able to understand Data Link Layer of OSI Model
- Students will able to understand hamming code and CRC.

3. Theory

Computer Networks Error Detection and Correction

Error

A condition when the receiver's information does not matches with the sender's information. During transmission, digital signals suffer from noise that can introduce errors in the binary bits traveling from sender to receiver. That means a 0 bit may change to 1 or a 1 bit may change to 0.

Some popular techniques for error detection are:

1. Simple Parity check

2. Two-dimensional Parity check
3. Checksum

4. Cyclic redundancy check

1. Hamming Codes

Hamming code is a set of error-correction codes that can be used to detect and correct [bit](#) errors that can occur when computer data is moved or stored. Like other error-correction code, Hamming code makes use of the concept of parity and parity bits, which are bits that are added to data so that the validity of the data can be checked when it is read or after it has been received in a data transmission. Using more than one parity bit, an error-correction code can not only identify a single bit error in the data unit, but also its location in the data unit.

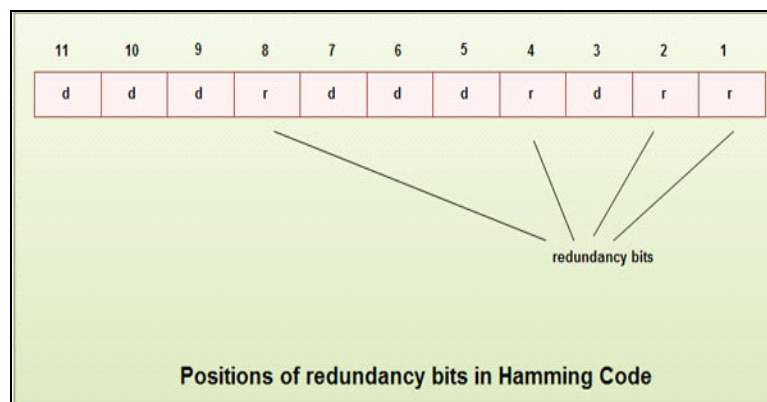
Calculating the Hamming Code

Determining the positions of redundancy bits

We know that to detect errors in a 7 bit code, 4 redundant bits are required.

Now, the next task is to determine the positions at which these redundancy bits will be placed within the data unit.

- These redundancy bits are placed at the positions which correspond to the power of 2.
- For example in case of 7 bit data, 4 redundancy bits are required, so making total number of bits as 11. The redundancy bits are placed in position 1, 2, 4 and 8 as shown in fig.



- In Hamming code, each r bit is the VRC for one combination of data bits. r₁ is the VRC bit for one combination of data bits, r₂ is the VRC for another combination of data bits and so on.
- Each data bit may be included in more than one VRC calculation.
- r₁ bit is calculated using all bits positions whose binary representation includes a 1 in the rightmost position.
- bit calculated using all the bit positions with a 1 in the second position and so on.
- Therefore the various r bits are parity bits for different combination of bits.

The various combinations are:

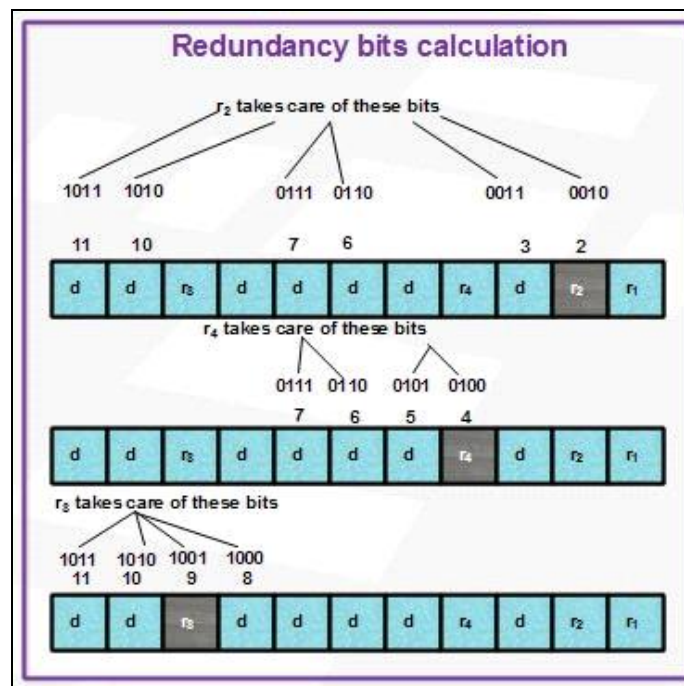
r_1 : bits 1,3,5, 7, 9, 11

r_2 : bits 2, 3, 6, 7, 10, 11

r_4 : bits 4, 5, 6, 7

r_8 : bits 8, 9, 10, 11

Example of Hamming Code Generation



Suppose a binary data 1001101 is to be transmitted. To implement hamming code for this, following steps are used:

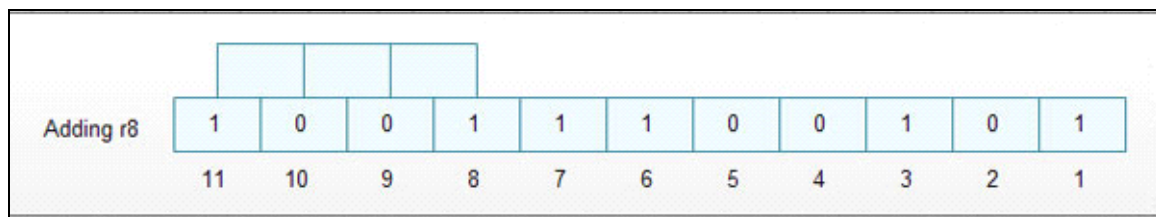
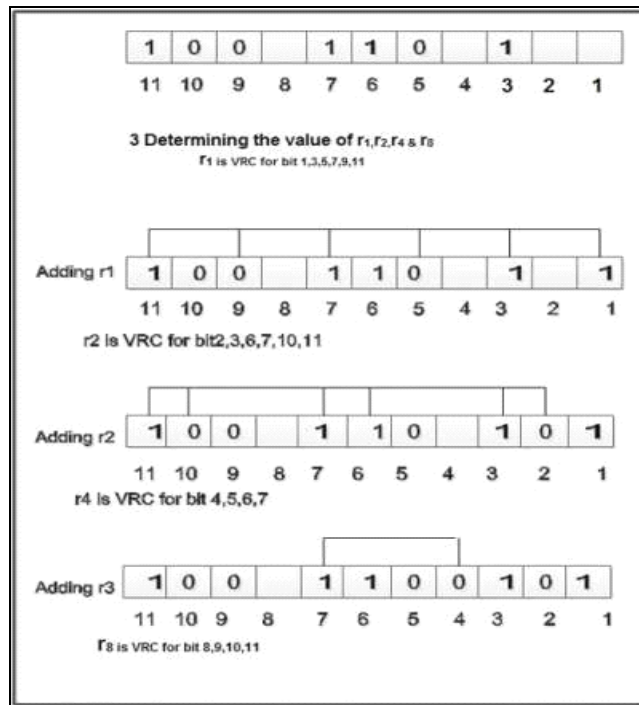
1. Calculating the number of redundancy bits required. Since number of data bits is 7, the value of r is calculated as

$$2^r \geq m + r + 1$$

$$2^4 \geq 7 + 4 + 1$$

Therefore no. of redundancy bits = 4

2. Determining the positions of various data bits and redundancy bits. The various r bits are placed at the position that corresponds to the power of 2 *i.e.* 1, 2, 4, 8.



4. Thus data 1 0 0 1 1 1 0 0 1 0 1 with be transmitted.

Error Detection & Correction

Data sent: 1 0 0 1 1 1 0 0 1 0 1

Data received: 1 0 0 1 0 1 0 0 1 0 1 (seventh bit changed)

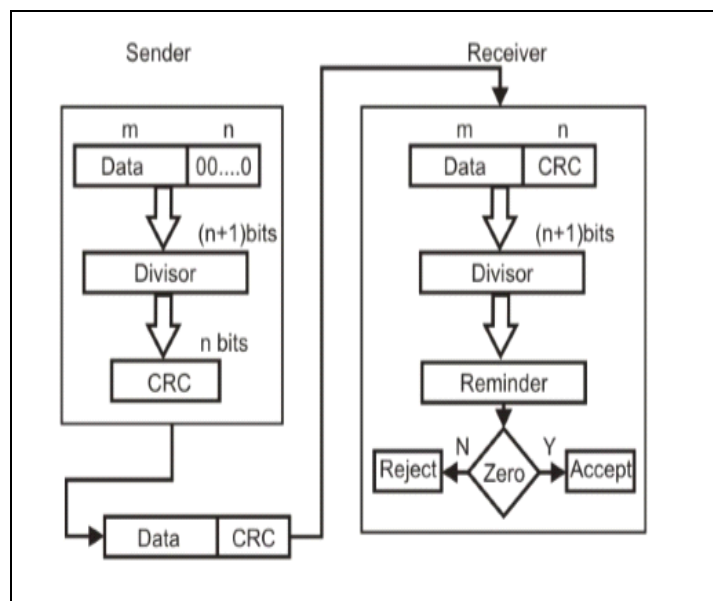
The receive takes the transmission and recalculates four new VRCs using the same set of bits used by sender plus the relevant parity (r) bit for each set as shown in fig.

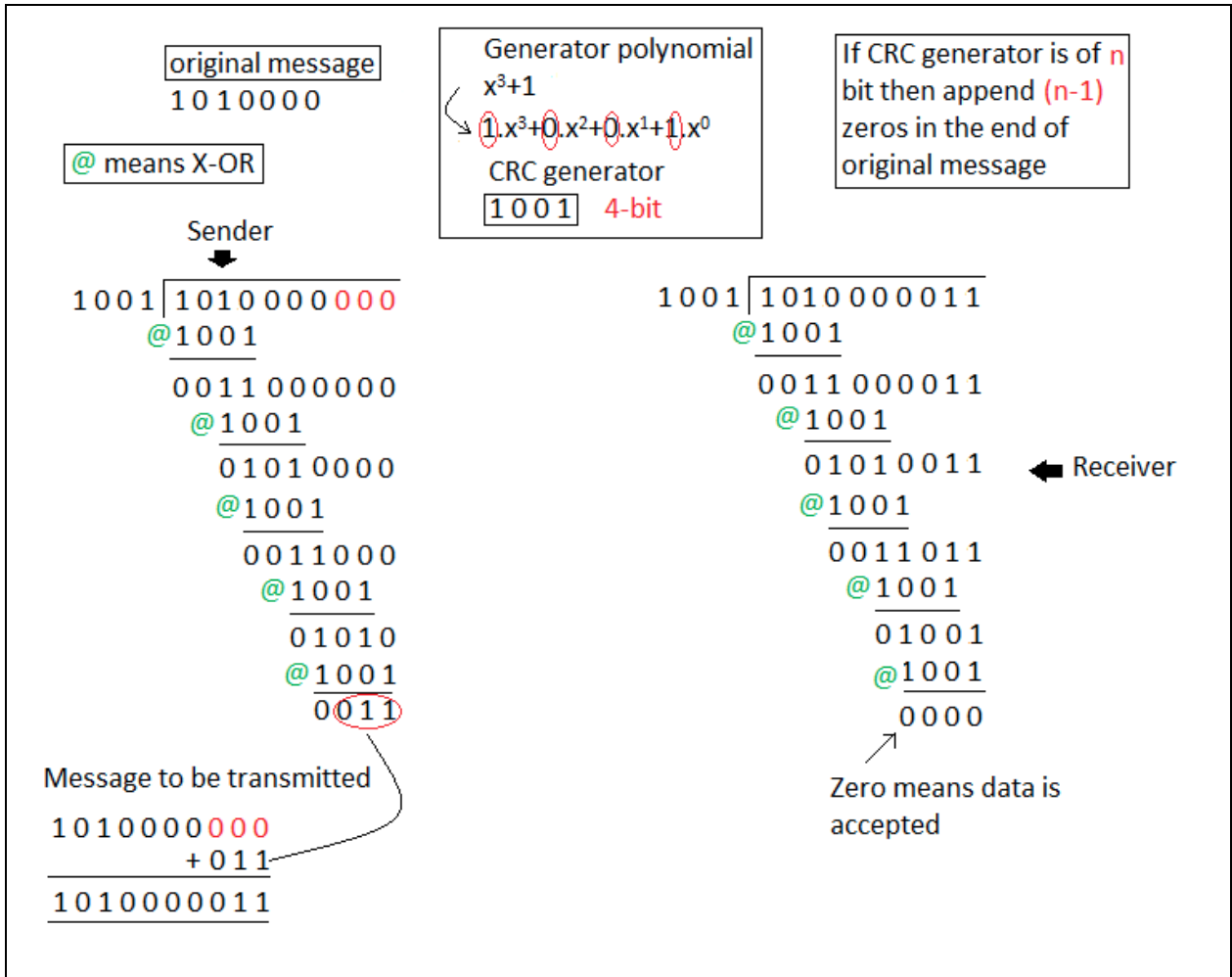
Then it assembles the new parity values into a binary number in order of r position (r₈, r₄, r₂, r₁).

In this example, this step gives us the binary number 0111. This corresponds to decimal 7. Therefore bit number 7 contains an error. To correct this error, bit 7 is reversed from 0 to 1.

2. Cyclic redundancy check (CRC)

- CRC is based on binary division.
- In CRC, a sequence of redundant bits, called cyclic redundancy check bits, are appended to the end of data unit so that the resulting data unit becomes exactly divisible by a second, predetermined binary number.
- At the destination, the incoming data unit is divided by the same number. If at this step there is no remainder, the data unit is assumed to be correct and is therefore accepted.
- A remainder indicates that the data unit has been damaged in transit and therefore must be rejected.





Conclusion:

Hence we have studied error detection and correction using Hamming Codes and CRC.

Assignment Group-A_6

Problem Definition:

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. (Use JAVA/PYTHON)

1. Prerequisite:

1. Data Link Layer: Roles, Protocols
2. Java Programming Syntax
3. Wireshark Tool

2. Learning Objectives:

- Students will be able to understand Go back N and Selective Repeat Modes of Sliding Window Protocol

3. Theory

Data-link layer is responsible for implementation of point-to-point flow and error control mechanism.

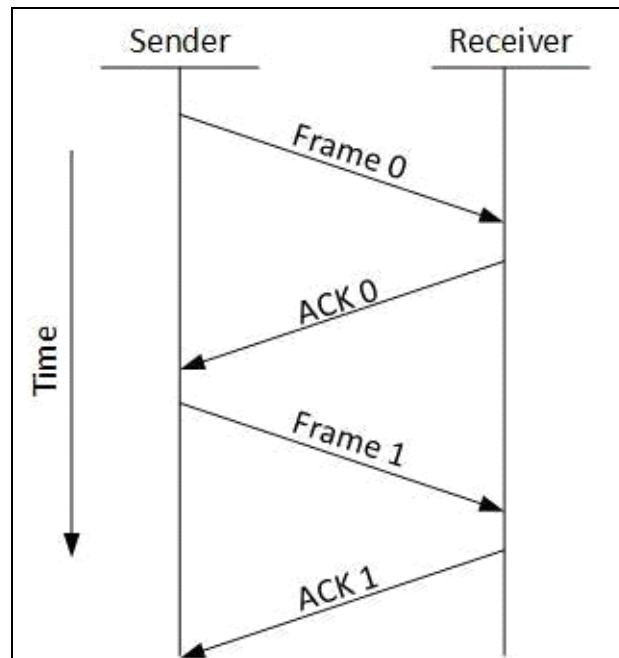
Flow Control

When a data frame (Layer-2 data) is sent from one host to another over a single medium, it is required that the sender and receiver should work at the same speed. That is, sender sends at a speed on which the receiver can process and accept the data. What if the speed (hardware/software) of the sender or receiver differs? If sender is sending too fast the receiver may be overloaded, (swamped) and data may be lost.

Two types of mechanisms can be deployed to control the flow:

1. Stop and Wait

This flow control mechanism forces the sender after transmitting a data frame to stop and wait until the acknowledgement of the data-frame sent is received.



2. Sliding Window

In this flow control mechanism, both sender and receiver agree on the number of data-frames after which the acknowledgement should be sent. As we learnt, stop and wait flow control mechanism wastes resources, this protocol tries to make use of underlying resources as much as possible.

Error Control

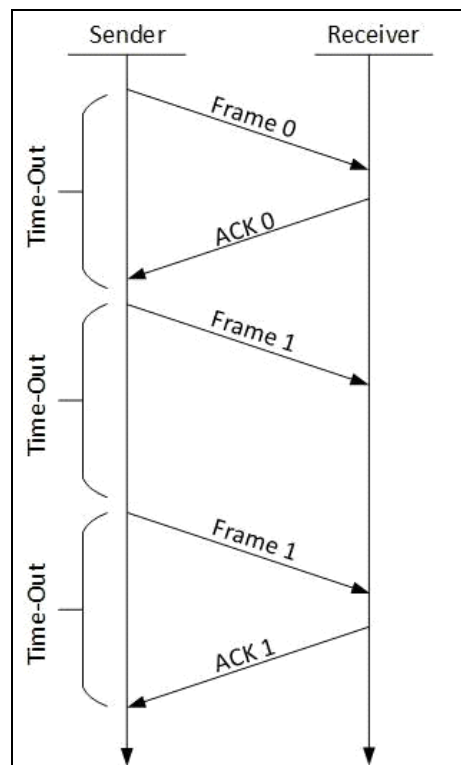
When data-frame is transmitted, there is a probability that data-frame may be lost in the transit or it is received corrupted. In both cases, the receiver does not receive the correct data-frame and sender does not know anything about any loss. In such case, both sender and receiver are equipped with some protocols which helps them to detect transit errors such as loss of data-frame. Hence, either the sender retransmits the data-frame or the receiver may request to resend the previous data-frame.

Requirements for error control mechanism:

- **Error detection** - The sender and receiver, either both or any, must ascertain that there is some error in the transit.
- **Positive ACK** - When the receiver receives a correct frame, it should acknowledge it.
- **Negative ACK** - When the receiver receives a damaged frame or a duplicate frame, it sends a NACK back to the sender and the sender must retransmit the correct frame.
- **Retransmission:** The sender maintains a clock and sets a timeout period. If an acknowledgement of a data-frame previously transmitted does not arrive before the timeout the sender retransmits the frame, thinking that the frame or its acknowledgement is lost in transit.

There are three types of techniques available which Data-link layer may deploy to control the errors by Automatic Repeat Requests (ARQ):

1. Stop-and-wait ARQ

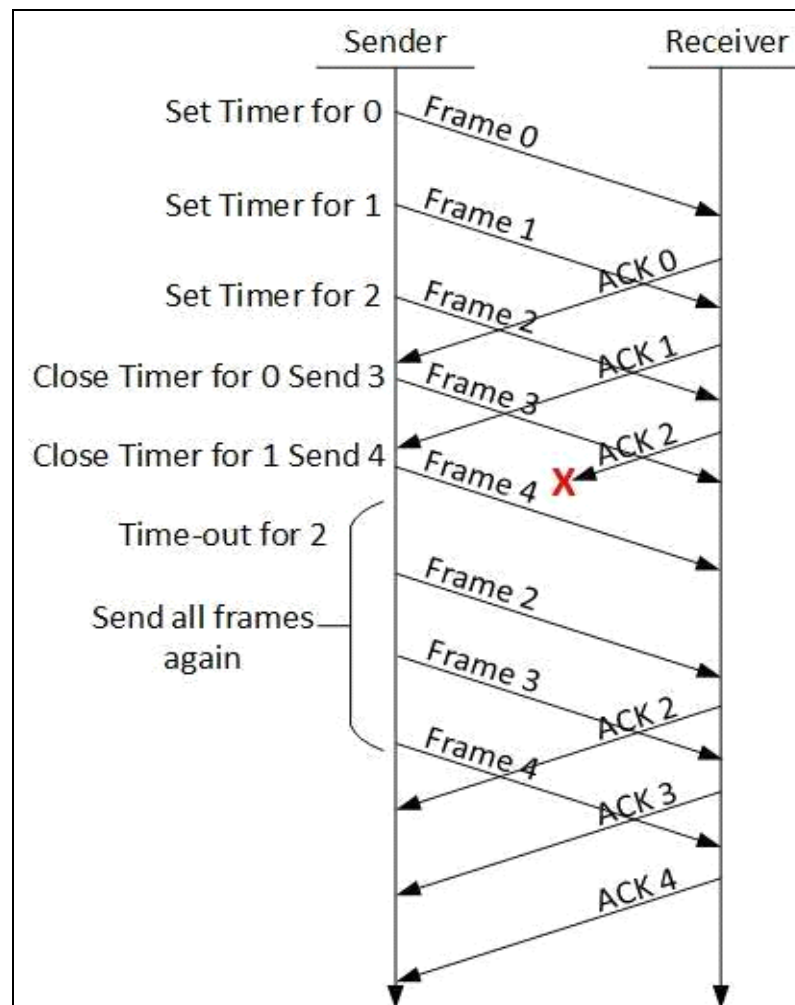


The following transition may occur in Stop-and-Wait ARQ:

- The sender maintains a timeout counter.
- When a frame is sent, the sender starts the timeout counter.
- If acknowledgement of frame comes in time, the sender transmits the next frame in queue.
- If acknowledgement does not come in time, the sender assumes that either the frame or its acknowledgement is lost in transit. Sender retransmits the frame and starts the timeout counter.
- If a negative acknowledgement is received, the sender retransmits the frame.

2. Go-Back-N ARQ

Stop and wait ARQ mechanism does not utilize the resources at their best. When the acknowledgement is received, the sender sits idle and does nothing. In Go-Back-N ARQ method, both sender and receiver maintain a window.

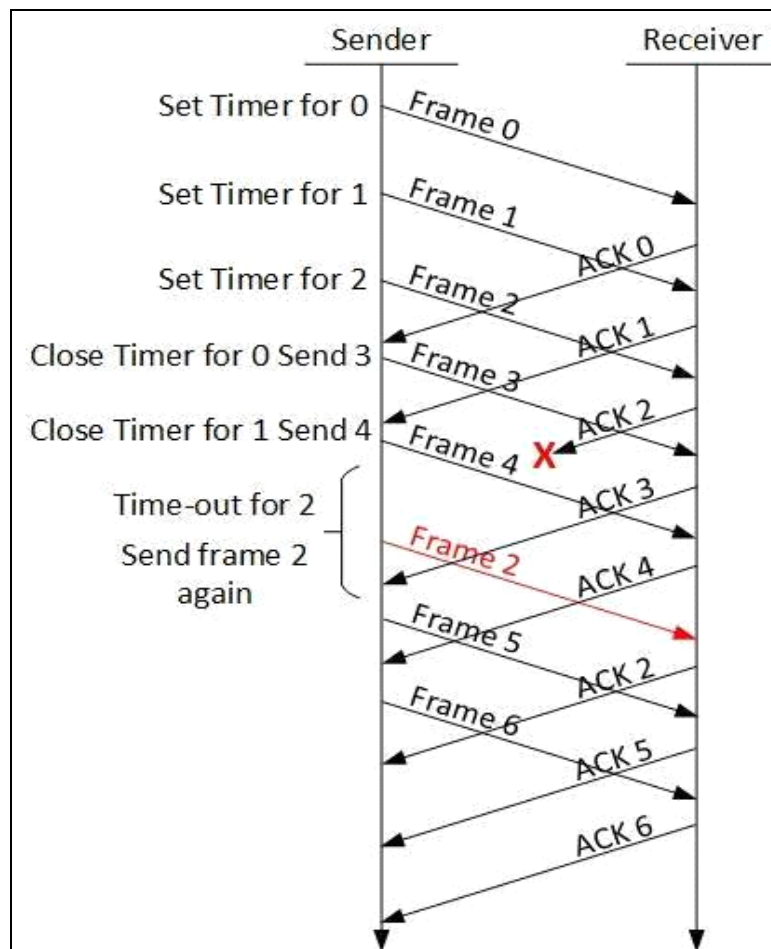


The sending-window size enables the sender to send multiple frames without receiving the acknowledgement of the previous ones. The receiving-window enables the receiver to receive multiple frames and acknowledge them. The receiver keeps track of incoming frame's sequence number.

When the sender sends all the frames in window, it checks up to what sequence number it has received positive acknowledgement. If all frames are positively acknowledged, the sender sends next set of frames. If sender finds that it has received NACK or has not receive any ACK for a particular frame, it retransmits all the frames after which it does not receive any positive ACK.

3. Selective Repeat ARQ

In Go-back-N ARQ, it is assumed that the receiver does not have any buffer space for its window size and has to process each frame as it comes. This enforces the sender to retransmit all the frames which are not acknowledged.



In Selective-Repeat ARQ, the receiver while keeping track of sequence numbers, buffers the frames in memory and sends NACK for only frame which is missing or damaged.

The sender in this case, sends only packet for which NACK is received.

Testing

1. Run Wireshark tool
2. Run program
3. Capture packets in Wireshark

Conclusion:

Hence we have studied Go back N and Selective Repeat Modes of Sliding Window Protocol and captured packets in Wireshark tool.

Assignment Group-A_7

Problem Definition:

Write a program to demonstrate subnetting and find the subnet masks. (Use JAVA/PYTHON)

1. Prerequisite:

1. Network Layer: Roles, Protocols
2. Java Programming Syntax

2. Learning Objectives:

- Students will be able to understand IP Addressing and Subnetting

3. Theory

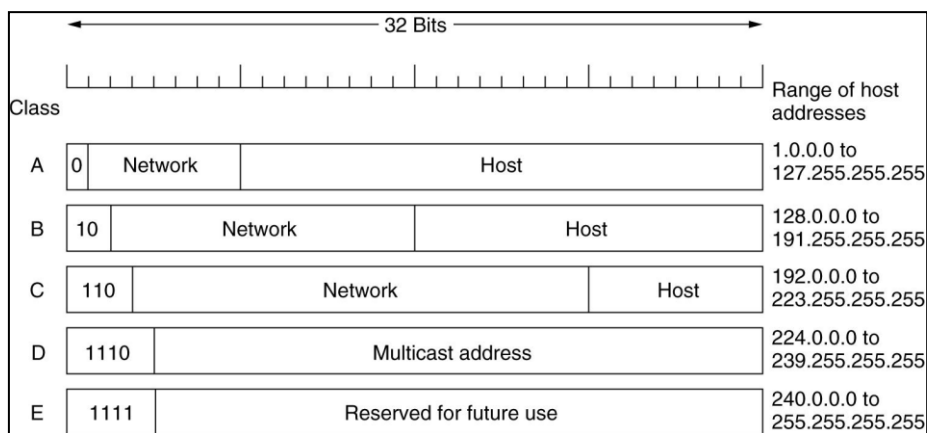
Introduction to IPv4

The identifier used in the IP layer of the TCP/IP protocol suite to identify each device connected to the Internet is called the Internet address or IP address. An IP address is a 32-bit address that uniquely and universally defines the connection of a host or a router to the Internet. IP addresses are unique. They are unique in the sense that each address defines one, and only one, connection to the Internet. Two devices on the Internet can never have the same address. The address space of IPv4 is 2^{32} or 4,294,967,296.

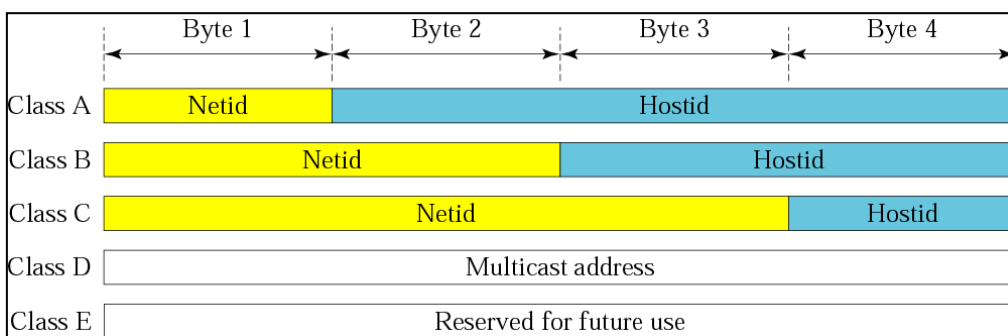
Network classes

Internet addresses are allocated by the Internet Network Information Center (INIC), the organization that administers the Internet. These IP addresses are divided into classes. The most common of these are classes A, B, and C. Classes D and E exist, but are not generally used by end users. Each of the address classes has a different default subnet mask. You can identify the class of an IP address by looking at its first octet. Following are the ranges of Class A, B, and C Internet addresses, each with an example address:

- Class A networks use a default subnet mask of 255.0.0.0 and have 0-127 as their first octet. The address 10.52.36.11 is a class A address. Its first octet is 10, which is between 1 and 126, inclusive.
- Class B networks use a default subnet mask of 255.255.0.0 and have 128-191 as their first octet. The address 172.16.52.63 is a class B address. Its first octet is 172, which is between 128 and 191, inclusive.
- Class C networks use a default subnet mask of 255.255.255.0 and have 192-223 as their first octet. The address 192.168.123.132 is a class C address. Its first octet is 192, which is between 192 and 223, inclusive.



IPv4 Classes and It's Range



Distribution of NetId and HostId

0 0		This host		
0 0	...	0 0	Host	A host on this network
1 1		Broadcast on the local network		
Network	1 1 1 1	...	1 1 1 1	Broadcast on a distant network
127	(Anything)			Loopback

Special IP Address

<i>Class</i>	<i>Netids</i>	<i>Blocks</i>
A	10.0.0	1
B	172.16 to 172.31	16
C	192.168.0 to 192.168.255	256

Addresses for Private Networks

The network address is the beginning address of each block. It can be found by applying the default mask to any of the addresses in the block (including itself). It retains the netid of the block and sets the hostid to zero.

Table 1: Default masks

<i>Class</i>	<i>Mask in binary</i>	<i>Mask in dotted-decimal</i>
A	11111111 00000000 00000000 00000000	255.0.0.0
B	11111111 11111111 00000000 00000000	255.255.0.0
C	11111111 11111111 11111111 00000000	255.255.255.0

Need of Subnetting

Specifically, the network addresses available for assignment to organizations are close to depletion. This is coupled with the ever-increasing demand for addresses from organizations that want connection to the Internet.

There are 4 of the major reasons for subnetting or segmenting network?

1. To divide a large network into smaller segments to reduce traffic and speed up the sections of your network.
2. To connect networks across geographical areas.
3. To connect different topologies such as Ethernet, Token Ring, and FDDI together via routers.
4. To avoid physical limitations such as maximum cable lengths or exceeding the maximum number of computers on a segment.

In this section we briefly discuss solution: Subnetting. A Class A, B, or C TCP/IP network can be further divided, or subnetted, by a system administrator.

Example

A service provider has given you the Class C network range 209.50.1.0. Your company must break the network into 20 separate subnets.

Step 1) Determine the number of subnets and convert to binary

- In this example, the binary representation of 20 = 00010100.

Step 2) Reserve required bits in subnet mask and find incremental value

- The binary value of 20 subnets tells us that we need at least 5 network bits to satisfy this requirement (since you cannot get the number 20 with any less than 5 bits – 10100)

- Our original subnet mask is 255.255.255.0 (Class C subnet)

- The full binary representation of the subnet mask is as follows: 255.255.255.0 = 11111111.11111111.11111111.00000000

- We must “convert” 5 of the client bits (0) to network bits (1) in order to satisfy the requirements: New Mask = 11111111.11111111.11111111.11111000

- If we convert the mask back to decimal, we now have the subnet mask that will be used on all the new networks – 255.255.255.248 - Our increment bit is the last possible network bit, converted back to a binary number:

New Mask = 11111111.11111111.11111111.1111(1)000 – bit with the parenthesis is your increment bit. If you convert this bit to a decimal number, it becomes the number 8

Step 3) Use increment to find network ranges

- Start with your given network address and add your increment to the subnetted octet:
209.50.1.0 209.50.1.8 209.50.1.16 ...etc

- You can now fill in your end ranges, which is the last possible IP address before you start the next range 209.50.1.0 – 209.50.1.7 209.50.1.8 – 209.50.1.15 209.50.1.16 – 209.50.1.23
...etc

- You can then assign these ranges to your networks. Remember the first and last address from each range (network / broadcast IP) is unusable.

Conclusion:

Hence we have studied IP Addressing and Subnetting.

Assignment Group-A_8

Problem Definition:

Write a program for DNS lookup. Given an IP address input, it should return URL and vice versa. (Use JAVA/PYTHON)

1. Prerequisite:

1. Application Layer: Roles, Protocols
2. Java Programming Syntax

2. Learning Objectives:

- Students will be able to understand working of DNS Protocol

3. Theory

DNS

Domain Name System (DNS) is the default name resolution service used in a Microsoft Windows Server 2003 network. DNS is part of the Windows Server 2003 TCP/IP protocol suite and all TCP/IP network connections are, by default, configured with the IP address of at least one DNS server in order to perform name resolution on the network.

DNS Architecture

DNS architecture is a hierarchical distributed database and an associated set of protocols that define:

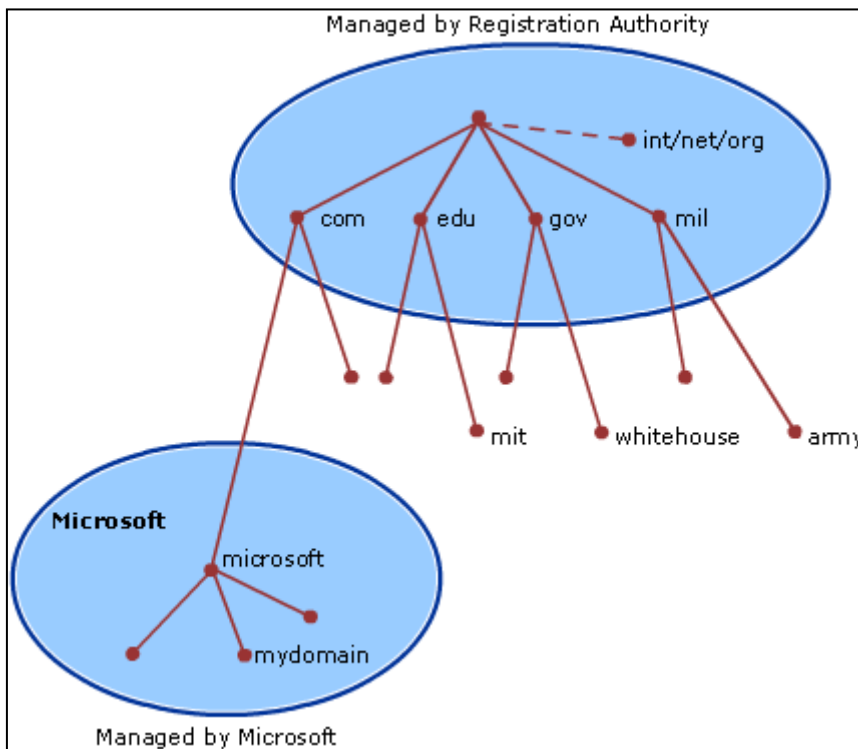
- A mechanism for querying and updating the database.
- A mechanism for replicating the information in the database among servers.
- A schema of the database.

DNS Domain Names

The Domain Name System is implemented as a hierarchical and distributed database containing various types of data, including host names and domain names. The names in a DNS database form a hierarchical tree structure called the domain namespace. Domain names consist of individual labels separated by dots, for example: mydomain.microsoft.com.

A Fully Qualified Domain Name (FQDN) uniquely identifies the hosts position within the DNS hierarchical tree by specifying a list of names separated by dots in the path from the referenced host to the root. The next figure shows an example of a DNS tree with a host called mydomain within the microsoft.com. domain. The FQDN for the host would be mydomain.microsoft.com.

DNS Domain Name Hierarchy



Types of DNS Domain Names

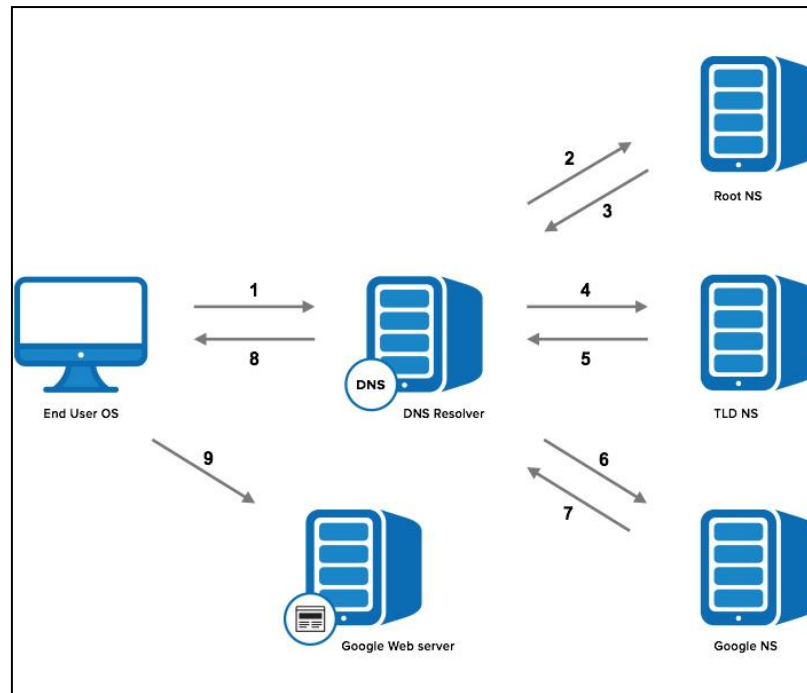
Name Type	Description	Example
Root domain	This is the top of the tree, representing an unnamed level; it is sometimes shown as two empty quotation marks (""), indicating a null value. When used in a DNS domain name, it is stated by a trailing period (.) to designate that the name is located at the root or highest level of the domain hierarchy. In this instance, the DNS domain name is considered to be complete and points to an exact location in the tree of names.	A single period (.) or a period used at the end of a name, such as "example.microsoft.com."

	Names stated this way are called fully qualified domain names (FQDNs).	
Top level domain	A name used to indicate a country/region or the type of organization using a name.	“.com”, which indicates a name registered to a business for commercial use on the Internet.
Second level domain	Variable-length names registered to an individual or organization for use on the Internet. These names are always based upon an appropriate top-level domain, depending on the type of organization or geographic location where a name is used.	“microsoft.com.”, which is the second-level domain name registered to Microsoft by the Internet DNS domain name registrar.
Subdomain	Additional names that an organization can create that are derived from the registered second-level domain name. These include names added to grow the DNS tree of names in an organization and divide it into departments or geographic locations.	“example.microsoft.com.”, which is a fictitious subdomain assigned by Microsoft for use in documentation example names.
Host or resource name	Names that represent a leaf in the DNS tree of names and identify a specific resource. Typically, the leftmost label of a DNS domain name identifies a specific computer on the network. For example, if a name at this level is used in a host (A) RR, it is used to look up the IP address of computer based on its host name.	“host-a.example.microsoft.com.”, where the first label (“host-a”) is the DNS host name for a specific computer on the network.

Working of DNS Lookup

DNS is what translates your familiar domain name (www.google.com) into an IP address your browser can use (173.194.33.174).

Before the page and any resource on the page is loaded, the DNS must be resolved so the browser can establish a TCP connection to make the HTTP request. In addition, for every external resource referenced by a URL, the DNS resolution must complete the same steps (per unique domain) before the request is made over HTTP. The DNS Resolution process starts when the user types a URL address on the browser and hits Enter. At this point, the browser asks the operating system for a specific page, in this case google.com.



Step 1: OS Recursive Query to DNS Resolver

Since the operating system doesn't know where "www.google.com" is, it queries a DNS resolver. The query the OS sends to the DNS Resolver has a special flag that tells it is a "recursive query." This means that the resolver must complete the recursion and the response must be either an IP address or an error.

Step 2: DNS Resolver Iterative Query to the Root Server

The resolver starts by querying one of the root DNS servers for the IP of "www.google.com." This query does not have the recursive flag and therefore is an "iterative query," meaning its response must be an address, the location of an authoritative name server, or an error. The root is represented in the hidden trailing "." at the end of the domain name. Typing this extra "." is not necessary as your browser automatically adds it.

Step 3: Root Server Response

These root servers hold the locations of all of the top level domains (TLDs) such as .com, .de, .io, and newer generic TLDs such as .camera.

The root doesn't have the IP info for "www.google.com," but it knows that .com might know, so it returns the location of the .com servers. The root responds with a list of the 13 locations of the .com gTLD servers, listed as NS or "name server" records.

Step 4: DNS Resolver Iterative Query to the TLD Server

Next the resolver queries one of the .com name servers for the location of google.com. Like the Root Servers, each of the TLDs have 4-13 clustered name servers existing in many locations. There are two types of TLDs: country codes (ccTLDs) run by government

organizations, and generic (gTLDs). Every gTLD has a different commercial entity responsible for running these servers. In this case, we will be using the gTLD servers controlled by Verisign, who run the .com, .net, .edu, and .gov among gTLDs.

Step 5: TLD Server Response

Each TLD server holds a list of all of the authoritative name servers for each domain in the TLD. For example, each of the 13 .com gTLD servers has a list with all of the name servers for every single .com domain. The .com gTLD server does not have the IP addresses for google.com, but it knows the location of google.com's name servers. The .com gTLD server responds with a list of all of google.com's NS records. In this case Google has four name servers, "ns1.google.com" to "ns4.google.com."

Step 6: DNS Resolver Iterative Query to the Google.com NS

Finally, the DNS resolver queries one of Google's name server for the IP of "www.google.com."

Step 7: Google.com NS Response

This time the queried Name Server knows the IPs and responds with an A or AAAA address record (depending on the query type) for IPv4 and IPv6, respectively.

Step 8: DNS Resolver Response to OS

At this point the resolver has finished the recursion process and is able to respond to the end user's operating system with an IP address.

Step 9: Browser Starts TCP Handshake

At this point the operating system, now in possession of www.google.com's IP address, provides the IP to the Application (browser), which initiates the TCP connection to start loading the page.

Conclusion:

Hence we have studied working of DNS protocol.

Assignment Group-B_1

Problem Definition:

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)

1. Prerequisite:

1. Protocols: TCP, UDP, CSMA, Ethernet
2. Network Routing: Shortest path routing, AODV
3. Network Simulator NS2

2. Learning Objectives:

- Students will be able to configure protocols like TCP, UDP, CSMA, Ethernet and AODV in network simulator 2 (NS2).

3. Theory

a. Monitoring traffic for the given topology

Creation of Tcl script

Create first Tcl script and call this as 'example1.tcl'.

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"
```

ns provides you with a very simple way to schedule events with the 'at' command.

The last line finally starts the simulation.

```
$ns run
```

You can actually save the file now and try to run it with 'ns example1.tcl'. You are going to get an error message like 'nam: empty trace file out.nam' though, because until now we haven't defined any objects (nodes, links, etc.) or events.

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.

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

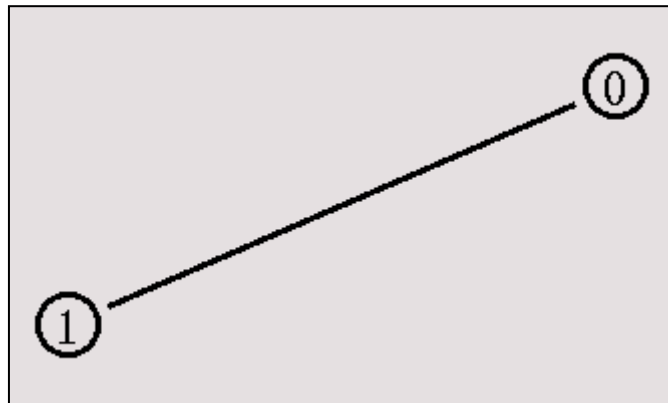
A new node object is created with the command '\$ns node'. The above code creates two nodes and assigns them to the handles 'n0' and 'n1'.

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.

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

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 generator 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.

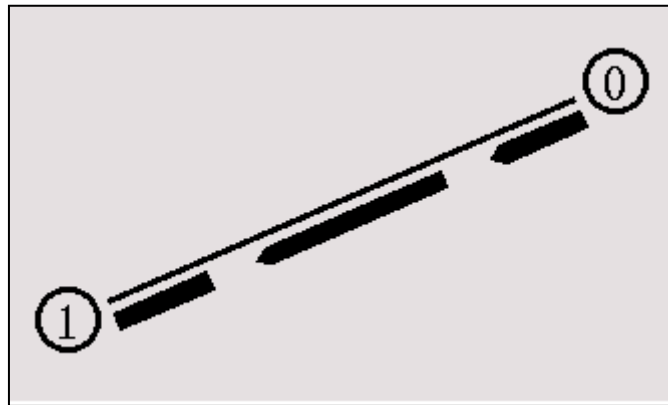

```
$ns connect $udp0 $null0
```

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"
```

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.



b. Analysis of CSMA and Ethernet protocols

Carrier-sense multiple access (CSMA) is a media access control (MAC) protocol in which a node verifies the absence of other traffic before transmitting on a shared transmission medium, such as an electrical bus or a band of the electromagnetic spectrum.

A transmitter attempts to determine whether another transmission is in progress before initiating a transmission using a carrier-sense mechanism. That is, it tries to detect the presence of a carrier signal from another node before attempting to transmit. If a carrier is sensed, the node waits for the transmission in progress to end before initiating its own transmission. Using CSMA, multiple nodes may send and receive on the same medium. Transmissions by one node are generally received by all other nodes connected to the medium.

Variations on basic CSMA include addition of collision-avoidance, collision-detection and collision-resolution techniques.

CSMA with collision detection

CSMA/CD is used to improve CSMA performance by terminating transmission as soon as a collision is detected, thus shortening the time required before a retry can be attempted.

CSMA with collision avoidance

In CSMA/CA collision avoidance is used to improve the performance of CSMA. If the transmission medium is sensed busy before transmission, then the transmission is deferred for a random interval. This random interval reduces the likelihood that two or more nodes waiting to transmit will simultaneously begin transmission upon termination of the detected transmission, thus reducing the incidence of collision.

c. Network Routing: Shortest path routing, AODV

Shortest path routing algorithm

Shortest path can be calculated only for the weighted graphs. The edges connecting two vertices can be assigned a nonnegative real number, called the weight of the edge. A graph with such weighted edges is called a weighted graph.

Let G be a weighted graph. Let u and v be two vertices in G , and let P be a path in G from u to v . The weight of the path P is the sum of the weights of all the edges on the path P , which is also called the weight of v from u via P .

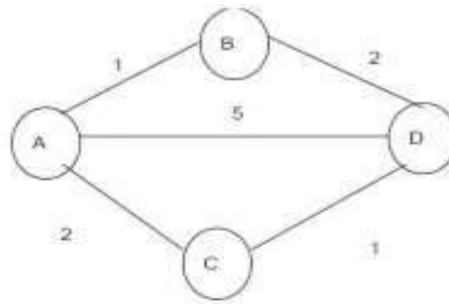
Let G be a weighted graph representing a highway structure. Suppose that the weight of an edge represents the travel time. For example, to plan monthly business trips, a salesperson wants to find the shortest path (that is, the path with the smallest weight) from her or his city to every other city in the graph. Many such problems exist in which we want to find the shortest path from a given vertex, called the source, to every other vertex in the graph. This section describes the shortest path algorithm, also called the greedy algorithm, developed by Dijkstra.

Given a vertex, say vertex (that is, a source), this section describes the shortest path algorithm.

The general algorithm is:

1. Initialize the array `smallestWeight` so that `smallestWeight[u] = weights[vertex, u]`.
 2. Set `smallestWeight[vertex] = 0`.
 3. Find the vertex, v , that is closest to $vertex$ for which the shortest path has not been determined.
 4. Mark v as the (next) vertex for which the smallest weight is found.
 5. For each vertex w in G , such that the shortest path from $vertex$ to w has not been determined and an edge (v, w) exists, if the weight of the path to w via v is smaller than its current weight, update the weight of w to the weight of v + the weight of the edge (v, w) .
- Because there are n vertices, repeat Steps 3 through 5, $n - 1$ times.

Example : Shortest Path



SOURCE : A

Edge	Cost	Path
B	1	A-B
C	2	A-C
D	5	A-D

Direct Cost
Select A-B

Edge	Cost	Path
B	1	A-B
C	2	A-C
D	3	A-B-D

Therefore A-B-D (3) < A-D (5)
Adjusted from B
Select A-C

Edge	Cost	Path
B	1	A-B
C	2	A-C
D	3	A-B-D

Therefore A-B-D (3) < A-D(5)

Ad hoc On-demand Distance Vector (AODV) Routing Protocol

- AODV is a packet routing protocol designed for use in mobile ad hoc networks (MANET)
- Intended for networks that may contain thousands of nodes
- One of a class of demand-driven protocols
 - The route discovery mechanism is invoked only if a route to a destination is not known
- UDP is the transport layer protocol
- Source, destination and next hop are addressed using IP addressing
- Each node maintains a routing table that contains information about reaching destination nodes.
 - Each entry is keyed to a destination node.

Routing Table Fields

- Destination IP address
- Destination Sequence Number
- Valid Destination Sequence Number Flag
- Other state and routing flags

d. Analysis of congestion control (TCP and UDP)

Congestion

Congestion occurs when the source sends more packets than the destination can handle. When this congestion occurs performance will degrade. Congestion occurs when these buffers gets filled on the destination side. The packets are normally temporarily stored in the buffers of the source and the destination before forwarding it to their upper layers.

When congestion occurs, the destination has only two options with the arriving packets, to drop it or keep it. If the destination drops the new arriving packets and keeps the old packets then this mechanism is called 'Y' model. If the destination drops the old packets and fills them with new packet, then this mechanism is called Milk model. In both the cases packets are dropped. Two common ways to detect congestion are timeout and duplicate acknowledgement.

Congestion control

Congestion control can be used to calculate the amount of data the sender can send to the destination on the network. Determining the amount of data is not easy, as the bandwidth changes from time to time, the connections get connected and disconnected. Based on these factors the sender should be able to adjust the traffic. TCP congestion control algorithms are used to detect and control congestion. The following are the congestion algorithms we will be discussing.

1. Additive Increase/ Multiplicative Decrease
2. Slow Start
3. Congestion Avoidance
4. Fast Retransmit
5. Fast recovery

1. Additive Increase / Multiplicative Decrease

This algorithm is used on the sender side of the network. The congestion window S_{SIZE} is the amount of data the sender can send into the network before receiving the ACK. Advertised window R_{SIZE} is the amount of data the receiver side can receive on the network. The TCP source set the congestion window based on the level of congestion on the network. This is done by decreasing the congestion window when the congestion increases and increases the congestion window if the congestion decreases. This mechanism is commonly called as Additive Increase/ Multiplicative Decrease.

2. Slow start

The main disadvantage in the Additive Increase/ Multiplicative Decrease method is the sender decreases the congestion by half when it detects congestion and increase only by one for each successful ACK received. If the window size is large and/or the congestion window size is increased from 1, then we waste many congestion windows. The slow start algorithm is used to solve this problem of increment by one. The S_{SIZE} is the amount of data the sender can send into the network before receiving the ACK. R_{SIZE} is the amount of data the receiver side can receive on the

network. The Ssthresh is the slow start threshold used to control the amount of data flow on the network. The slow start algorithm is used when the ssthresh is less than the threshold Ssthresh.

3. Congestion avoidance

The ssthresh is the amount of data the sender can send into the network before receiving the ACK. rwnd is the amount of data the receiver side can receive on the network. The Ssthresh is the slow start threshold used to control the amount of data flow on the network. The congestion avoidance algorithm is used when the ssthresh is greater than the threshold Ssthresh. As the packets are sent the ssthresh is increased by one full size segment per roundtrip time. This continues until congestion is detected.

4. Fast retransmission

Both the above three algorithms use timeout for detecting the congestion. The disadvantage here is the sender need to wait for the timeout to happen. To improve the congestion detection the sender uses duplicate ACK. Every time a packet arrives at the receiving side, the receiver sends an ACK to the sender. When a packet arrives out of order at the receiving side, TCP cannot yet acknowledge the data the packet contains because the earlier packet has not yet arrived. The receiver sends the same ACK which it sent last time resulting in duplicate ACK. This is illustrated below.

5. Fast recovery

Fast recovery algorithm governs the transmission of new data until a non-duplicate ACK arrives. The reason for not performing slow start is that the receipt of the duplicate ACKs not only indicates that a segment has been lost, but also that segments are most likely leaving the network. The fast retransmit and fast recovery algorithms are usually implemented together as follows.

1. When the third duplicate ACK is received, set ssthresh no more than ssthresh = $\max(\text{FlightSize} / 2, 2 * \text{SMSS})$, where FlightSize is the amount of outstanding data in the network
2. Retransmit the lost segment and set ssthresh to ssthresh plus $3 * \text{SMSS}$. This artificially "inflates" the congestion window by the number of segments (three) that have left the network and which the receiver has buffered.
3. For each additional duplicate ACK received, increment ssthresh by SMSS. This artificially inflates the congestion window in order to reflect the additional segment that has left the network.
4. Transmit a segment, if allowed by the new value of ssthresh and the receiver's advertised window.
5. When the next ACK arrives that acknowledges new data, set ssthresh to ssthresh (the value set in step 1). This is termed "deflating" the window. This ACK should be the acknowledgment elicited by the retransmission from step 1, one RTT after the retransmission (though it may arrive sooner in the presence of significant out-of-order delivery of data segments at the receiver). Additionally, this ACK should acknowledge

all the intermediate segments sent between the lost segment and the receipt of the third duplicate ACK, if none of these were lost.

Conclusion:

Hence we have implemented TCP, UDP, CSMA, Ethernet protocols, Shortest path routing, AODV and congestion control (TCP and UDP) using NS2.

Assignment Group-B_2

Problem Definition:

Configure RIP/OSPF/BGP using packet Tracer.

1. Prerequisite:

1. Protocols: RIP, OSPF, BGP
2. Packet Tracer

2. Learning Objectives:

- Students will be able to configure protocols like RIP, OSPF, BGP using Packet Tracer.

3. Theory

Routing Protocols

Routing protocols maintain routing tables where a routing table contains a route to every destination network.

Dynamic Routing Protocols

There are three types of it as follows:

1. Routing Information Protocol (RIP)
2. Open Shortest Path First (OSPF)
3. Border Gateway Protocol (BGP)

RIP and OSPF are Interior Gateway Protocols (IGPs); they are designed to operate in a single autonomous system (AS). (An AS is a group of networks administered by the same authority). BGP is an Exterior Gateway Protocol (EGP), which allows routers in different autonomous systems to exchange routes. Because BGP routers must regulate traffic between networks controlled by organizations with different policies.

How Routing Protocols Work

A router constructs its routing table using the information it receives from other routers. The router changes its routing table in response to routing updates that provide additional information or notification that conditions in the network have changed (for example, a link has failed). This responsiveness explains why using a routing protocol is often called dynamic routing.

The protocol must dictate parameters such as the following:

■ **How routers compute a route's metric and select the best route for their routing table:**

Routing protocols can have a relatively complicated system for calculating a route's metric. So that you can select the best routing protocol (or protocols) for your network environment. If necessary, you can change which routes are chosen by altering the default metrics that a protocol assigns certain routes.

■ **What information routers include in routing updates:** With some routing protocols, routers exchange their entire routing tables. With other routing protocols, routers exchange only portions of the routing table.

■ **Which routers and router interfaces send and receive updates:** Most protocols specify that when routers receive an update on an interface, they do not send the same update from that interface. This common sense rule minimizes overhead.

■ **When routers send and receive updates and hellos:** To lower overhead and conserve bandwidth, you can alter how often routers send certain messages.

1. Routing Information Protocol (RIP)

RIP is one of the oldest dynamic routing protocols on the Internet that is still in use. RIP is an intradomain routing protocol that uses a distance vector approach to determine the paths between routers. RIP minimizes the number of hops on each path, where each point-to-point link or LAN constitutes a hop. Each RIP-enabled router periodically sends the content of its routing table to all its neighboring routers in an update message. For each routing table entry, the router sends the destination (host IP address or network IP address and associated prefix) and the distance to the destination measured in hops. When a router receives an update message from a neighboring router, it updates its own routing table

Configuring RIP on CISCO ROUTER

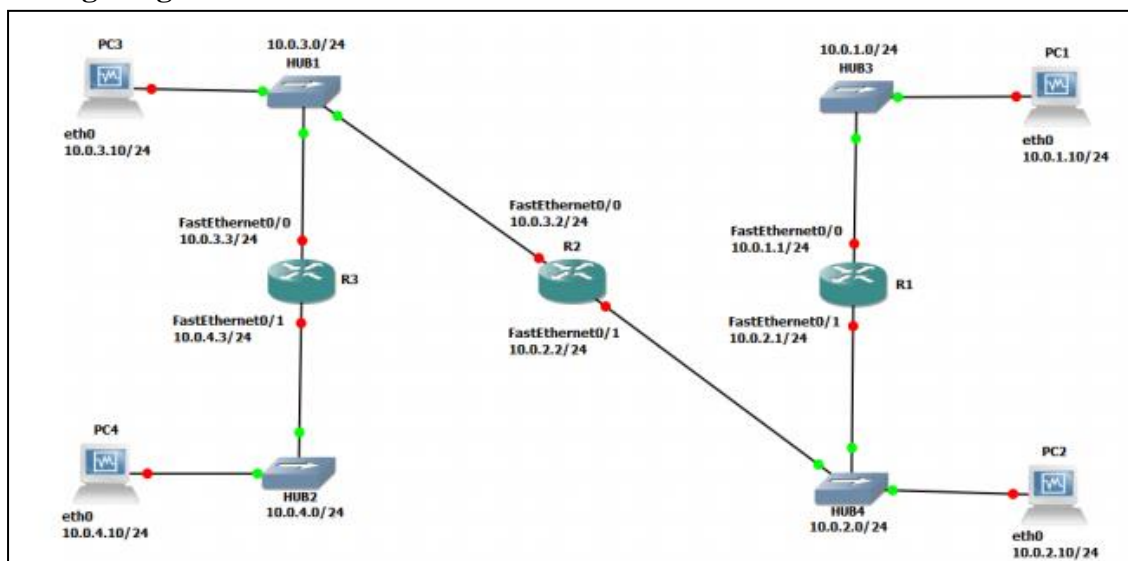


Fig. 1 Network topology

Cisco Routers	Ethernet Interface FastEthernet 0/0	Ethernet Interface FastEthernet 0/1
Router1	10.0.1.1 / 24	10.0.2.1 / 24
Router2	10.0.3.2 / 24	10.0.2.2 / 24
Router3	10.0.3.3 / 24	10.0.4.3 / 24

Linux PC	Ethernet Interface eth0	Ethernet Interface eth1
PC1	10.0.1.10 / 24	Disabled
PC2	10.0.2.10 / 24	Disabled
PC3	10.0.3.10 / 24	Disabled
PC4	10.0.4.10 / 24	Disabled

Table 1: IP addresses of the Cisco routers and Linux PCs

Above Figures describe the network configuration

Exercise 1. Configuring RIP on Cisco routers

In this exercise, you will configure all the routers to run RIP. After the configuration, all the routers should be able to ping all the other routers. Following is a brief overview of the basic commands used to configure RIP on a Cisco router. Make sure you type in the command in the correct command mode (note the prompt).

IOS MODE: GLOBAL CONFIGURATION

`router rip` `no router rip`

Enables or disables RIP on the local router.

IOS MODE: PRIVILEGED EXEC

`debug ip rip` `no debug ip rip`

Enables or disables a debugging mode where the router displays a message for each received RIP packet.

IOS MODE: ROUTER CONFIGURATION

`network Netaddr` `no network Netaddr`

Associates or disables the network IP addresses `Netaddr` with RIP. RIP sends updates only on interfaces on which the network address has been associated with RIP.

`passive-interface Iface` `no passive-interface Iface`

Sets or disables the interface `Iface` in RIP passive mode. On an interface in passive mode, the router processes incoming RIP packets but does not transmit RIP packages.

`offset-list 0 in valueIface` `offset-list 0 out valueIface`

Increases the metric (hop count) of incoming RIP packages that arrive or outgoing RIP packets that are sent on interface `Iface` by `value`.

`timers basic update invalid hold-down flush`

update: The time interval between transmissions of RIP update messages (default: 30 seconds).

invalid: The time interval after which a route, which has not been updated, is declared

invalid (default: 180 seconds).

hold-down: Determines how long after a route has been updated as unavailable. A router will wait before accepting a new route with a lower metric. This introduces a delay for processing incoming RIP packets with routing updates after a link failure (default: 180 seconds).

flush: The amount of time that must pass before a route that has not been updated is removed from the routing tables (default: 240 seconds).

`flash-update-threshold time`

Sets the router not to perform triggered updates, when the next transmission of routing updates is due in `time`. If `time` is set to the same value as the update timer, then triggered update are disabled. In RIP, a triggered update means that a router sends a RIP packet with a routing update, whenever one of its routing table entries changes.

1. Connect the PCs and the Cisco Routers as shown in Figure1. The PCs and routers are connected with Ethernet hubs.

2. Start Routers by clicking the right button and select Start; then, open a terminal by clicking the right button and select Console.
3. On Router1, Router2, and Router3, configure the IP addresses as shown in Table 1, and enable the routing protocol RIP. The commands to set up Router 1 are as follows:

```
Router1>enable
Router1#configure terminal
Router1(config)#no ip routing
Router1(config)#ip routing
Router1(config)#router rip
Router1(config-router)#version 2
Router1(config-router)#network 10.0.0.0
Router1(config-router)#interface FastEthernet0/0
Router1(config-if)#no shutdown
Router1(config-if)#ip address 10.0.1.1 255.255.255.0
Router1(config-if)#interface FastEthernet0/1
Router1(config-if)#no shutdown
Router1(config-if)#ip address 10.0.2.1 255.255.255.0
Router1(config-if)#end
Router1#clear ip route *
```

4. After you have configured the routers, check the routing table at each router with the show ip route command. Each router should have four entries in the routing table: two entries for directly connected networks and two other entries for remote networks that were added by RIP.
5. From each router, issue a ping command to the IP address of interfaces FastEthernet0/0 and FastEthernet0/1 on all remote routers.

2. Open Shortest Path First (OSPF)

OSPF is a link state routing protocol, in which each router sends information on the cost metric of its network interfaces to all other routers in the network. The information about the interfaces is sent in messages that are called link state advertisements (LSAs). LSAs are disseminated using flooding; that is, a router sends its LSAs to all its neighbors, which, in turn, forward the LSAs to their neighbors and so on. However, each LSA is forwarded only once. Each router maintains a link state database of all received LSAs, which provides the router with complete information about the topology of the network. Routers use their link state databases to run a shortest-path algorithm that computes the shortest paths in the network.

The network configuration is shown in Figure 2 and Table 2. Note that PC1-4 are set up as routers.

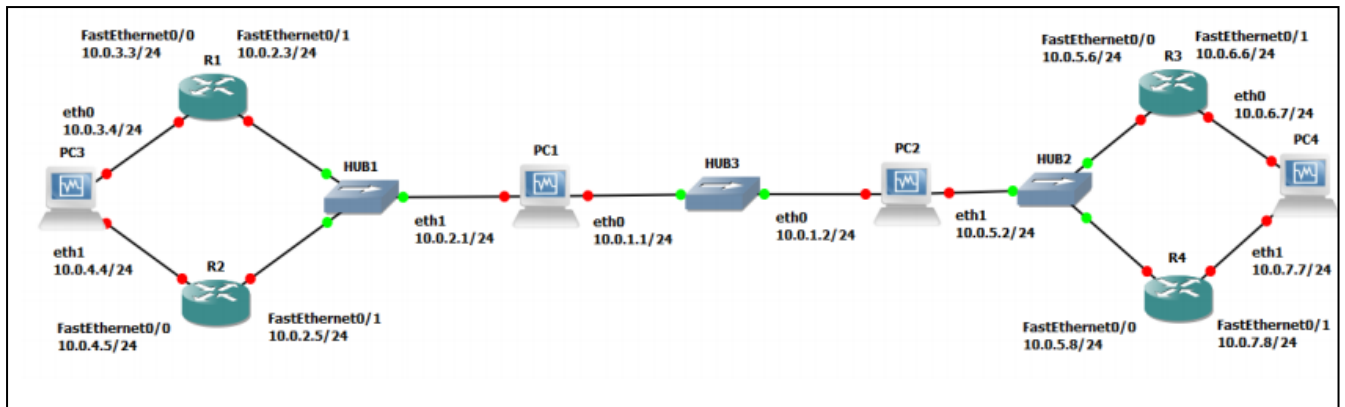


Figure 4.4 Network topology for Part 5

Cisco Routers	Ethernet Interface FastEthernet 0/0	Ethernet Interface FastEthernet 0/1
Router1	10.0.3.3 / 24	10.0.2.3 / 24
Router2	10.0.4.5 / 24	10.0.2.5 / 24
Router3	10.0.5.6 / 24	10.0.6.6 / 24
Router4	10.0.5.8 / 24	10.0.7.8 / 24

PCs	Interface eth0	Interface eth1
PC1	10.0.1.1 / 24	10.0.2.1 / 24
PC2	10.0.1.2 / 24	10.0.5.2 / 24
PC3	10.0.3.4 / 24	10.0.4.4 / 24
PC4	10.0.6.7 / 24	10.0.7.7 / 24

Table 2: . IP addresses of the routers and PCs

Configuring OSPF on Cisco routers

In this exercise, you configure OSPF on the Cisco routers. A brief description of the basic IOS commands used to configure OSPF on a Cisco router follows. As usual, each command must be issued in a particular IOS command mode.

IOS MODE: GLOBAL CONFIGURATION

```
router ospf process-id
```

Enables an OSPF routing process. Each router can execute multiple OSPF processes.

process-id is a number that identifies the process. In this lab, only one OSPF process is started per router, and the process-id value is always set to 1. (The process-id of a router does not need to be the same on all routers). The command enters the router configuration mode, which has the following command prompt:

```
Router1(config-router)#
```

```
no router ospf process-id
```

Disables the specified OSPF process.

IOS MODE: PRIVILEGED EXEC

```
show ipospf
```

Displays general information about the OSPF configuration

```
show ipospf database
```

Displays the link state database.

```
show ipospf border-routers
```

Displays the Area Border Router (ABR) and Autonomous System Boundary Router (ASBR).

```
clear ipospf process-id process
```

Resets the specified OSPF process.

IOS MODE: ROUTER CONFIGURATION

```
network NetaddrInvNetmask area AreaID
```

Associates a network prefix with OSPF and associates an OSPF area to the network address. The prefix is specified with an IP address (Netaddr) and an inverse net mask (InvNetmask). For example, Netaddr=10.0.0.0 and InvNetmask=0.255.255.255 specify the network prefix 10.0.0.0/8 and the broadcast area AreaID is a number that associates an area with the address range. Area 0 is reserved to specify the backbone area.

Example: To run OSPF on Router 1 for the address range 10.0.0.0/8 and assign it to Area 1, type

```
Router1(config-router)# network 10.0.0.0 0.255.255.255 area 1
```

```
no network Netaddr InvNetmask area AreaID
```

Disables OSPF for the specified network area.

```
passive-interface Iface
```

Sets interface Iface into passive mode. In passive mode, the router only receives and

OSPF messages.

```
router-id IPaddress
```

Assigns the IP address IPaddress as the router identifier (router-id) of the local OSPF router. In OSPF, the router-id is used in LSA messages to identify a router. In IOS, by default, a router selects the highest IP address as the router-id. This commands can be used to set the value explicitly.

1. Connect the routers as shown in Figure 2.
2. Configure the Cisco routers to run OSPF.

The following commands are used to configure Router1:

```
Router1> enable
Router1#configure terminal
Router1(config)#interface FastEthernet0/0
Router1(config-if)# no shutdown
Router1(config-if)#ip address 10.0.3.3 255.255.255.0
Router1(config-if)#interface FastEthernet0/1
Router1(config-if)# no shutdown
Router1(config-if)#ip address 10.0.2.3 255.255.255.0
Router1(config-if)#exit
Router1(config)# no ip routing
Router1(config)#ip routing
Router1(config)#no router rip
Router1(config)#router ospf 1
Router1(config-router)#network 10.0.0.0 0.255.255.255 area 1
Router1(config-if)#end
Router1#clear ip route *
```

These commands configure the IP addresses of the routers, disable RIP, and enable OSPF for Area 1 and network 10.0.0.0/8. Since no router-id is specified, the highest IP address of Router1, 10.0.3.3, is used as the router-id. The router-id can be verified by issuing the command show ip OSPF.

3. Set up the PCs as OSPF routers. Refer to Figure 2 for the connections and to Table 2 for the IP addresses. Use the following set of commands.

4. Now configure the PC's similar to the way you configured the routers

```
PC1% telnet localhost 2604
Password:zebra
ospfd>enable
ospfd#configure terminal
ospfd(config)#no router rip
ospfd(config)#router ospf
ospfd(config-router)#network 10.0.0.0/8 area 1
ospfd(config-router)#router-id 10.0.1.1
ospfd(config-router)#no passive-interface eth0
ospfd(config-router)#no passive-interface eth1
ospfd(config-router)#end
ospfd#exit
```

5. Enable ip_forwarding on all the PCs.

Observing convergence of OSPF

In comparison to the distance vector protocol RIP, the link state routing protocol OSPF

quickly adapts to changes in the network topology. In this exercise, you observe the interactions of OSPF after a change to the network topology.

1. On PC1, start to capture traffic with Wireshark on interface FastEthernet0/0. Set a filter to display only OSPF packets.
2. From PC3, run a trace command to PC4. Confirm from the output and Figure 2 whether the path from PC3 to PC4 includes Router3 or Router4.
3. Issue a ping command from PC3 to PC4 (10.0.7.7). Do not terminate the ping command until this exercise is completed.
4. If the path from PC3 to IP address 10.0.7.7 from Step 2 included Router3, then disconnect the Ethernet cable of FastEthernet0/1 interface of Router3. Otherwise, disconnect the Ethernet cable of FastEthernet0/1 interface of Router4. When the Ethernet cable is disconnected, the ping command on PC3 will show that IP address 10.0.7.7 is not reachable.
5. Now OSPF updates the routing tables. Use the Wireshark window on PC1 to observe the transmitted OSPF messages:
 - How quickly are OSPF messages sent after the cable is disconnected?
 - How many OSPF messages are sent?
 - Which type of OSPF packet is used for flooding link state information?
 - Describe the flooding of LSAs to all routers.
 - Which type of encapsulation is used for OSPF packets (TCP, UDP, or other)?
 - What is the destination address of OSPF packets?
6. Wait until the ping command is successful again, that is, ICMP Echo Reply messages arrive at PC3. This happens when the routing tables have been updated.
7. Stop the ping command and save the ping statics output.
 - Count the number of lost packets and calculate the time it took OSPF to update the routing tables. (The ping command issues an ICMP Echo Request message approximately once every second.)
8. Issue another trace command from PC3 to IP address 10.0.7.7. By now, the output should show the new route to PC4.
9. Save the link state database on all Cisco routers to a file, and verify that all routers indeed have the same link state database.
 - Compare the output of the command “show ip ospf database” from the Cisco routers
10. Stop Wireshark on PC1, and save the different types of OSPF packets captured by

Wireshark. Save one copy of each type of OSPF packet that you observed. a) Pick a single link state advertisement packet captured by Wireshark, and describe how to interpret the information contained in the link state advertisement.

3. Border Gateway Protocol (BGP)

This provides some exposure to the inter domain Border Gateway Protocol (BGP), which determines paths between autonomous systems on the Internet.

BGP uses a path vector algorithm, where routers exchange full path information of a route. An important feature of BGP is that it can define routing policies, which can be used by a network to specify which type of traffic it is willing to process. The network configuration for this part is shown in Figure 3, and the IP configuration information is given in table 3. the network has three autonomous systems with AS numbers 100, 200 and 300. PC4, is used to capture the BGP packets transmitted between the ASs.

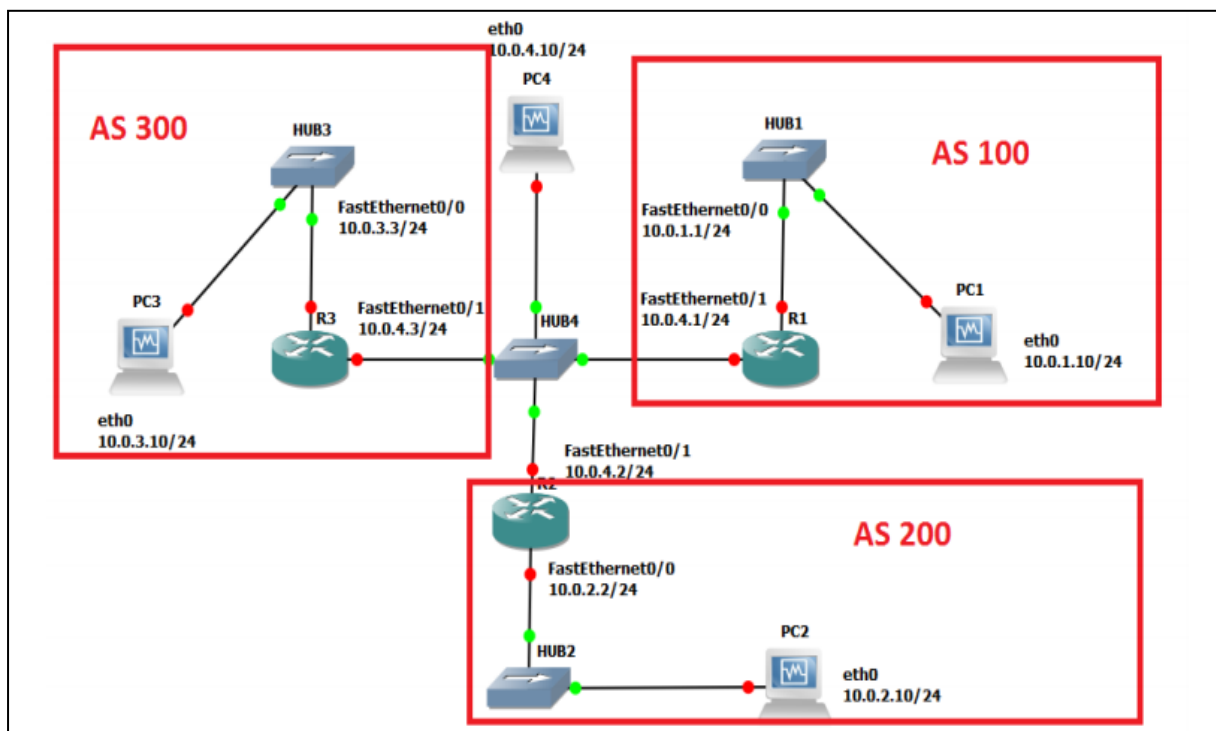


Figure 3 Network topology

VPCS	Ethernet Interface eth0	Ethernet Interface eth1
PC1	10.0.1.10 / 24	Disabled
PC2	10.0.2.10 / 24	Disabled
PC3	10.0.3.10 / 24	Disabled
PC4	10.0.4.10 / 24	Disabled

Cisco Routers	Ethernet Interface FastEthernet 0/0	Ethernet Interface FastEthernet 0/1
Router1	10.0.1.1 / 24	10.0.4.1 / 24
Router2	10.0.2.2 / 24	10.0.4.2 / 24
Router3	10.0.3.3 / 24	10.0.4.3 / 24

Table 3 IP addresses of the routers and PCs

Basic BGP configuration

Here, you configure the Cisco routers as BGP routers and you assign routers to autonomous systems. The configuration is completed when you can issue ping commands between any two PCs. Next we summarize the Cisco IOS commands that are used to enable BGP.

IOS MODE: GLOBAL CONFIGURATION

```
router bgp ASnumber
```

Enables the BGP routing protocol and sets the autonomous system number to ASnumber.

The command enters the router configuration mode with the following prompt:

```
Router1(config-router)#
```

```
no router bgp ASnumber
```

Disables the BGP routing process.

IOS MODE: PRIVILEGED EXEC

```
show ipbgp
```

Displays the BGP routing table.

```
show ipbgp neighbors
```

Displays the neighbors, also called peers, of this BGP router.

```
show ipbgp paths
```

Displays the BGP path information in the local database.

```
clear ipbgp *
```

Deletes BGP routing information

IOS MODE: ROUTER CONFIGURATION

```
network Netaddr
```

```
network Netaddr mask netmask
```

Specifies a network address that will be advertised by the local BGP process. A network mask maybe added to denote the length of the network prefix.

```
neighbor IPaddress remote-as ASnumber
```

Adds a neighbor to the BGP neighbor table. IPaddress is the IP address and ASnumber is the AS number of the neighbor.

```
timers bgp keepalive holdtime
```

Sets the values of the keep alive and hold time timers of the BGP process. BGP routers exchange periodic messages to confirm that the connection between the routers is maintained. The interval between these messages is keep alive seconds (default: 60 seconds). The number of seconds that a BGP router waits for any BGP message before it decides that a connection is down is specified by the hold time (default: 180 seconds).

1. Disable all RIP or OSPF processes that are running on the Cisco routers. Use the following commands:

```
Router1# no router ospf 1
```

```
Router1# no router rip
```

2. Disable all RIP or OSPF processes running on the Linux PCs using the following command. For PC1, on the console at the prompt type:

```
PC1% /etc/init.d/quagga stop
```

3. Assign the IP addresses to Ethernet interface eth0 of each PC as indicated in Table 3

4. Disable eth1 on the Linux PCs using the following command as shown in Table 3. For PC1, on the console at the prompt type:

```
PC1% ifconfig eth1 down
```

5. Add a default gateway to PC1, PC2, and PC3 as follows:

```
PC1% route add default gw 10.0.1.1/24
```

```
PC2% route add default gw 10.0.2.2/24
```

```
PC3% route add default gw 10.0.3.3/24
```

6. Start Wireshark on PC4 and set a display filter to capture only BGP packets.

7. Configure the Cisco routers to run BGP with the autonomous system numbers shown in Figure 3. The routers must know the AS number of their neighbors. Following is the configuration for Router2. Router 2 is in AS 200 and neighbors are AS 100 and AS 300.

```
Router2> enable
Router2# configure terminal
Router2(config)#no ip routing
Router2(config)# ip routing
Router2(config)# interface FastEthernet0/0
Router2(config-if)# no shutdown
Router2(config-if)# ip address 10.2.2 255.255.255.0
Router2(config-if)# interface FastEthernet0/1
Router2(config-if)# no shutdown
Router2(config-if)# ip address 10.0.4.2 255.255.255.240
Router2(config-if)#router bgp 200
Router2(config-router)# neighbor 10.0.4.1 remote-as 100
Router2(config-router)# neighbor 10.0.4.3 remote-as 300
Router2(config-router)# network 10.0.2.0 mask 255.255.255.0
Router2(config-router)# end
Router2# clear ip bgp *
```

8. On PC1, issue a ping command to PC3. The command succeeds when BGP has converged.

9. Once the routing tables have converged, you see all the other AS entries in the BGP routing table. On each Cisco router, save the output of the following commands:

```
Router1# show ip route
```

```
Router1# show ip bgp
```

```
Router1# show ip bgp paths
```

- Describe the different types of BGP messages that you observe in the Wireshark window on PC4.
- Notice that BGP transmits messages over TCP connections. What is a reason that BGP uses TCP to transmit its messages?
- What is the IP address of the next-hop attribute for AS 100 on Router 2?
- What are the BGP peers in this topology?

10. Stop the Wireshark traffic capture on PC4 and save the BGP packets captured by Wireshark.

- a) Use the output to provide answers to the questions in Step 7.
- b) Which BGP message(s) contain(s) the AS-PATH information? Use a BGP message to illustrate your answer.
- c) Use the saved output to provide a brief explanation of how the routers find the proper path between the autonomous systems.

BGP convergence

Disconnect one of the links between two BGP peers and observe how the BGP protocol reconfigures the paths.

1. After previous Exercise, save the output of the command show ip BGP neighbors on Router.2. Pay attention to the neighbor AS information.
2. On PC4, run Wireshark and set a display filter for BGP. Observe the flow of BGP packets between the autonomous systems.
3. On all routers, change the keep alive timer to 10 seconds and the hold time timer to 30 seconds. This speeds up the convergence time by a factor of 6 as compared to the default values. The following are the commands for Router2:

```
Router2# configure terminal
```

```
Router2(config)#router bgp 200
```

```
Router2(config-router)# timers bgp 10 30
```

```
Router2(config-router)#end
```

```
Router2#clear ip bgp *
```

4. Disconnect the cable of interface FastEthernet0/1 on Router1.

- From the output you saved, describe how the BGP routers learn that a link is down. (Hint: Look at the BGP State field)

- Which BGP messages indicate that there is a link problem? Use a BGP message to answer the question.

5. Use the command `show ip BGP neighbors` on Router2 and Router3 to obtain the neighbor information. Save the output.

6. Wait until BGP converges. Save the routing tables on Router2 and Router3. What can you say?

7. Stop the Wireshark traffic captured on PC4 and save the Wireshark BGP packets.

a) From the output you saved, describe how the BGP routers learn that a link is down. (Hint: Look at the BGP State field)

b) Which BGP messages indicate that there is a link problem? Use a BGP message to answer the question.

Conclusion:

Hence we have configured RIP, OSPF and BGP using packet tracer.

Assignment Group-B_3

Problem Definition:

Study of any network simulation tools - To create a network with three nodes and establish a TCP connection between node 0 and node 1 such that node 0 will send TCP packet to node 2 via node 1

1. Prerequisite:

1. Protocols: TCP
2. Network Simulator NS2

2. Learning Objectives:

- Students will be able to create a simple topology and able to configure protocols like TCP in network simulator 2 (NS2).

3. Theory

Introduction

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"
```

ns provides you with a very simple way to schedule events with the 'at' command.

The last line finally starts the simulation.

```
$ns run
```

Three nodes, one link

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

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

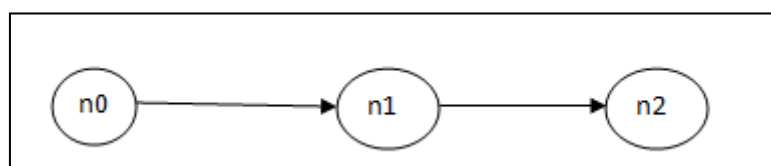
A new node object is created with the command '\$ns node'. The above code creates three nodes and assigns them to the handles 'n0' and 'n1' and 'n2'.

The next line connects the three nodes.

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

This line tells the simulator object to connect the nodes n0 and n1 as well as n1 to n2 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

The next step is to send some data from node n0 to node n2 through 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 n2.

```
#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 generator 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 n2.

```
set null0 [new Agent/Null]
$ns attach-agent $n2 $null0
```

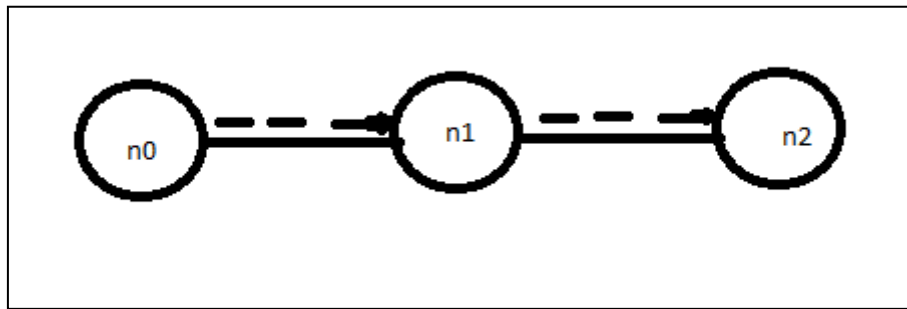
Now the two agents have to be connected with each other.

```
$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 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 2 through node 1.

**Conclusion:**

Thus we have created simple topology and configured TCP in NS2.

Assignment Group-B_4

Problem Definition:

Write a program using TCP sockets for wired network to implement (Use JAVA/PYTHON)

- a. Peer to Peer Chat
- b. Multiuser Chat

1. Prerequisite:

1. Protocols: TCP
2. Java Programming Syntax

2. Learning Objectives:

- Students will able understand TCP socket programming in java
- Students will able to design chat server using TCP socket programming in java

3. Theory

Java Socket Programming

Java Socket programming is used for communication between the applications running on different JRE.

Java Socket programming can be connection-oriented or connection-less.

Socket and ServerSocket classes are used for connection-oriented socket programming and DatagramSocket and DatagramPacket classes are used for connection-less socket programming.

The client in socket programming must know two information:

1. IP Address of Server, and
2. Port number.

ServerSocket class

The **java.net.ServerSocket** class is used by server applications to obtain a port and listen for client requests. The ServerSocket class can be used to create a server socket. This object is used to establish communication with the clients.

Important methods

Method	Description
1) public Socket accept()	returns the socket and establish a connection between server and client.
2) public void bind(SocketAddress host, int backlog)	Binds the socket to the specified server and port in the SocketAddress object.
2) public synchronized void close()	closes the server socket.

Socket class

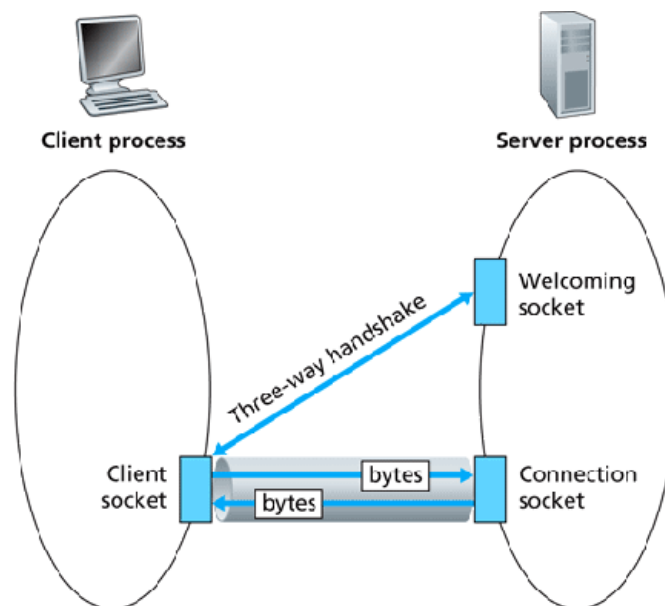
The java.net.Socket class represents the socket that both the client and the server use to communicate with each other. The client obtains a Socket object by instantiating one, whereas the server obtains a Socket object from the return value of the accept() method.

Important methods

Method	Description
1) public Socket(String host, int port)	This method attempts to connect to the specified server at the specified port
2) public void connect(SocketAddress host, int timeout)	This method connects the socket to the specified host. This method is needed only when you instantiate the Socket using the no-argument constructor.
3) public InputStream getInputStream()	returns the InputStream attached with this socket.
4) public OutputStream getOutputStream()	returns the OutputStream attached with this socket.
5) public synchronized void close()	closes this socket

TCP Socket Example

TCP 3-way Handshake

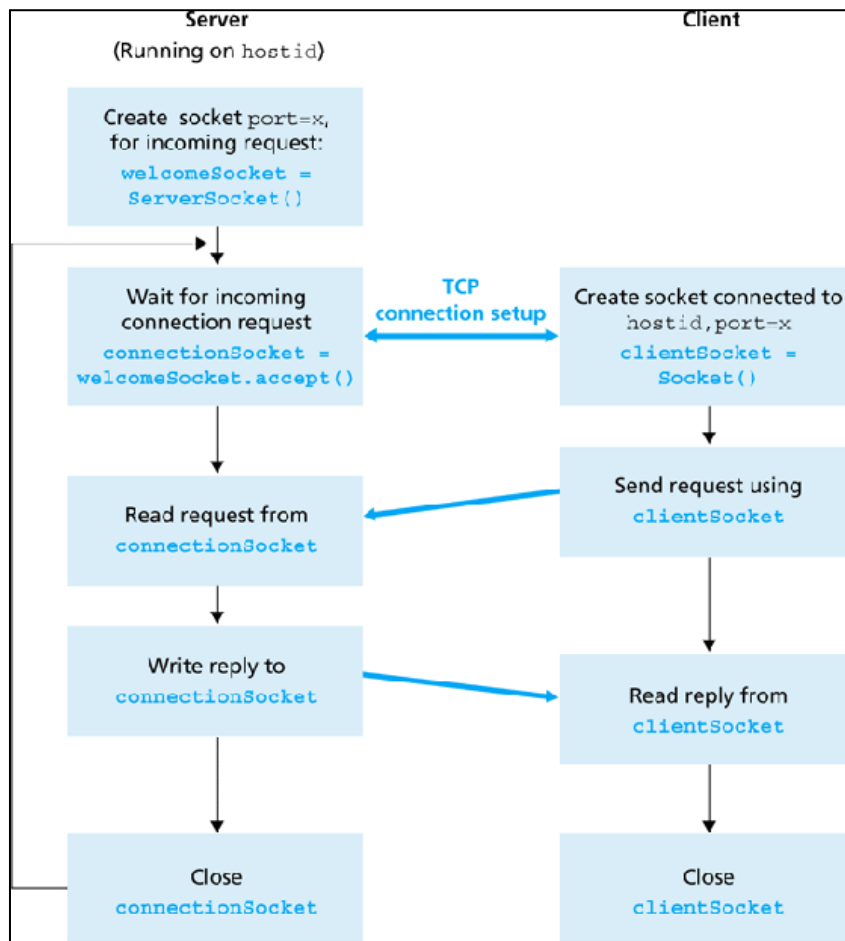


Basics:

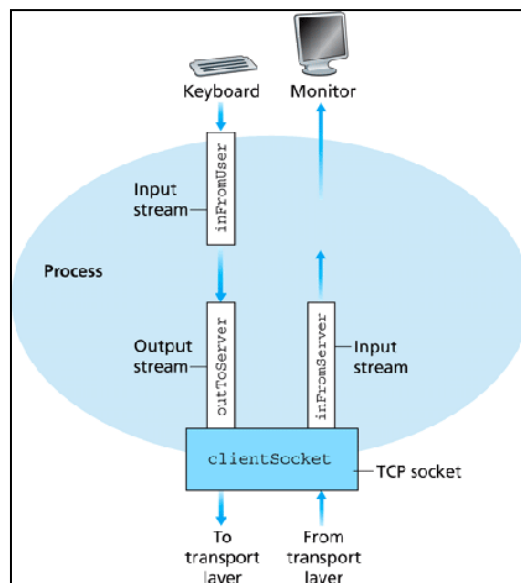
1. Hostname and port are used to specify transport endpoints.
2. Socket -- the communication object.
3. TCP properties: reliable, connection-oriented, byte-stream, connection established before application-level protocols exchange information, two-way communication.
4. UDP properties: unreliable, packet-switched, packet data, no connection overhead, application-level protocols exchange information immediately, two-way communication.

A socket connection is a 4-tuple -- (HostA, PortA, HostB, PortB) -- uniquely defining the connection.

TCP client/server communication flow:



TCP Communication Model



Testing

1. Run Wireshark tool
2. Run client and server program
3. Exchange messages
4. Capture TCP packets in Wireshark

Conclusion:

Hence we have studied chat server using TCP Socket Programming in Java and captured TCP packets in Wireshark tool.