**INDEX**

**Subject: Computer Networks (3150710)**

**Enrollment No: 180170107030**

**Name of the Student: Gabu Siddharth Merambhai**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sr. No** | **Aim** | **CO** | **Date** | **Page No** | **Marks (out of**  **10)** | **Faculty Signature** |
| 1 | Networking Commands | CO2 | 10/7 | 1 |  |  |
| 2 | Configuration of Wireshark packet analyzer and capturing packets using TCPDUMP/Wiresharp. | CO1 | 17/7 | 16 |  |  |
| 3 | Dissection of Packets using Wireshark | CO1 | 24/7 | 19 |  |  |
| 4 | Simulating LAN configuration using the Cisco Packet tracer emulator. | CO4 | 31/7 | 22 |  |  |
| 5 | Design a heterogeneous network that consists of a wired and wireless LAN connection using a router. | CO4 | 7/8 | 29 |  |  |
| 6 | Network Subnetting | CO4 | 14/8 | 33 |  |  |
| 7 | Wireless Router Configuration in Cisco Packet Tracer | CO4 | 4/9 | 37 |  |  |
| 8 | Implementing echo sever using Socket programming. | CO3 | 11/9 | 40 |  |  |
| 9 | Implementing file server using socket  programming. | CO3 | 18/9 | 43 |  |  |
| 10 | Implementing Error Detection mechanism using  Socket Programming. | CO5 | 9/10 | 46 |  |  |

|  |  |  |
| --- | --- | --- |
| **Sr. No** | **Practical Description** | **CO** |
| 1 | **Basics and Networking Commands** | **CO2** |
| **Execution of various Networking Commands.**  Understand the below commands properly. Execute the commands with different switch options.  Note: Refer manual for the each command using man command in linux platform Students are supposed to prepare a write up for the below commands along with the screenshot of their execution with different parameters applied.  1. ifconfig 2. traceroute 3. netstat 4. arp 5. ssh  6. ping 7. nslookup 8. iwconfig 9. ftp 10. wget | |
| 2 | **Packet Capturing and Dissection** | **CO1** |
| **Capturing and filtering packets using TCPDUMP/Wireshark Tool.**  You need to first install Linux version of Wireshark software in your computer which is available for download a[t http://www.wireshark.org/.](http://www.wireshark.org/)  In this assignment, you will be evaluated for your familiarity in using Tcpdump and Wire shark utility for packet capture and analysis. Start packet capture  in Wireshark application and then open your web browser and  type in an URL of website of your choice ([www.](http://www/) Vgecg.ac.in, [www.google.com,](http://www.google.com/) etc). Apply various filters on it as discussed during the lab session. | |
| 3 | **Dissection of packets using wire shark.**  Configure the Webserver in your local machine (e.g., Apache web server) and deploy the server side scripting page which validates your userid and password. Design a client side html page which prompts for user id and password.  Trace the step by step transactional flow between client and webserver (i.e., dns, http and tcp stream) when you press submit on client page. Find the user id and password  from the TCP stream. Summarize the flow in your own words. | |

|  |  |  |
| --- | --- | --- |
| 4 | **Network topology Configuration** | **CO4** |
| **LAN Configuration**  Create and Simulate LAN configuration using Cisco Packet tracer emulator. Save the Pkt file of your network configuration. | |
| 5 | **Heterogeneous Network Configuration**  Design a heterogeneous network which consists of wired and wireless LAN connected using router. Create a file server which allows user to download and upload the files.  Prepare the document which should comprise the screen shot. Save the Pkt file of your network configuration. | |
| 6 | **Subnetting Network**  You have sub-netted your class C network 192.168.1.0 with a subnet mask of 255.255.255.252. Please list the following: number of networks, number of hosts per network, the full range of the first three networks, and the usable address range from those first three networks. Additionally, identify the broadcast addresses for each network.  Design this network in packet tracer and simulate the working of the network. Save the Pkt file of  your network design | |
| 7 | **Wireless Router Configuration**  Configuring various network parameter in Wireless Router   1. Configure security key 2. Enable DHCP server 3. Apply Application/Network/Mac layer Filter 4. Configure Port Filtering | |
| 8 | **Socket Programming**  **Client-Sever Programming in C** | **CO3/**  **CO5** |
| Implementing connection oriented echo sever using Socket programming. | |
| 9 | Implementing connection oriented file server using socket programming. | |
| 10 | Implementing connection less Error Detection mechanism using Socket Programming. | |

Practical – 1

**Aim:** Networking Command

**Arp(Address Resolution Protocol):**

Displays and modifies entries in the Address Resolution Protocol (ARP) cache. The ARP cache contains one or more tables that are used to store IP addresses and their resolved Ethernet or Token Ring physical addresses. There is a separate table for each Ethernet or Token Ring network adapter installed on your computer. Used without parameters, Arp displays help information.

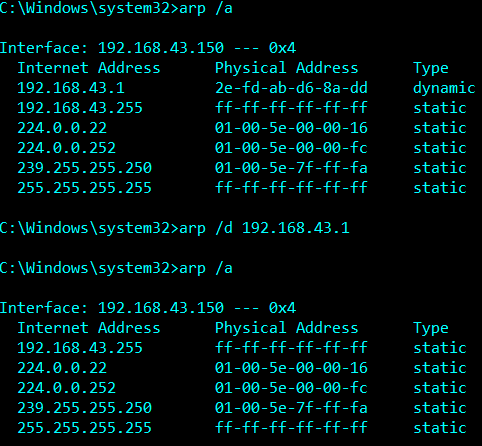
Syntax: Arp [parameter [<addr>]

/a - Displays current arp cache tables for all interfaces.

/d [inetaddr] - Deletes an entry with a specific IP address, where inetaddr is the IP address.

/n -To display the arp cache entry for a specific IP address.

/s [<inetaddr> <etheraddr>] - Adds a static entry to the arp cache that resolves the IP address inetaddr to the physical address etheraddr.



**Ipconfig:**

Displays all current TCP/IP network configuration values and refreshes Dynamic Host Configuration Protocol (DHCP) and Domain Name System (DNS) settings. Used without parameters, ipconfig displays Internet Protocol version 4 (IPv4) and IPv6 addresses, subnet mask, and default gateway for all adapters.

Syntax: ipconfig [/allcompartments]

/all - Display full configuration information.

/release [adapter] - Release the IP address for the specified adapter.

/renew [adapter] - Renew the IP address for the specified adapter.

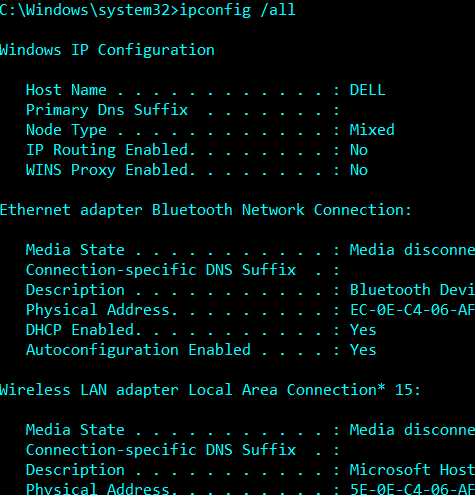
/flushdns - Purge the DNS Resolver cache.

/registerdns - Refresh all DHCP leases and re-register DNS names.

/displaydns - Display the contents of the DNS Resolver Cache.

/showclassid [adapter] - Display all the DHCP class IDs allowed for adapter.

/setclassid adapter [classid] - Modify the dhcp class id.



**Ping:**

The ping command verifies IP-level connectivity to another TCP/IP computer by sending Internet Control Message Protocol (ICMP) echo Request messages. The receipt of corresponding echo Reply messages are displayed, along with round-trip times. ping is the primary TCP/IP command used to troubleshoot connectivity, reachability, and name resolution. Used without parameters, ping displays help.

Syntax: ping [/parameters]

/t - Specifies that ping continue sending echo Request messages to the destination until interrupted.

/n <count> - Specifies the number of echo Request messages sent. The default is 4.

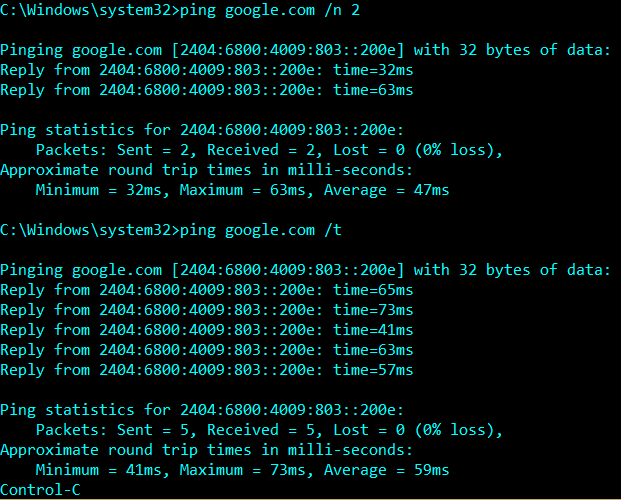
/l <size> - Specifies the length, in bytes, of the Data field in the echo Request messages sent. The default is 32. The maximum size is 65,527.

/w <timeout> - Specifies the amount of time, in milliseconds, to wait for the echo Reply message that corresponds to a given echo Request message to be received.

/4 - Specifies that IPv4 is used to ping.

/6 - Specifies that IPv6 is used to ping.

/s - Specifies that the Internet timestamp option in the IP header is used to record the time of arrival for the echo Request message and corresponding echo Reply message for each hop.



**Tracert:**

Determines the path taken to a destination by sending Internet Control Message Protocol (ICMP) echo Request or ICMPv6 messages to the destination with incrementally increasing time to Live (TTL) field values. The path displayed is the list of near/side router interfaces of the routers in the path between a source host and a destination. The near/side interface is the interface of the router that is closest to the sending host in the path.

Syntax: tracert [-d] [-h maximum\_hops] [-j host-list] [-w timeout]

[-R] [-S srcaddr] [-4] [-6] target\_name

-d Do not resolve addresses to hostnames.

-h maximum\_hops Maximum number of hops to search for target.

-j host-list Loose source route along host-list (IPv4-only).

-w timeout Wait timeout milliseconds for each reply.

-R Trace round-trip path (IPv6-only).

-S srcaddr Source address to use (IPv6-only).

-4 Force using IPv4.

-6 Force using IPv6.



**Getmac:**

This tool enables an administrator to display the MAC address

for network adapters on a system.

Parameter List:

/S system Specifies the remote system to connect to.

/U [domain\]user Specifies the user context under

which the command should execute.

/P [password] Specifies the password for the given

user context. Prompts for input if omitted.

/FO format Specifies the format in which the output

is to be displayed.

Valid values: "TABLE", "LIST", "CSV".

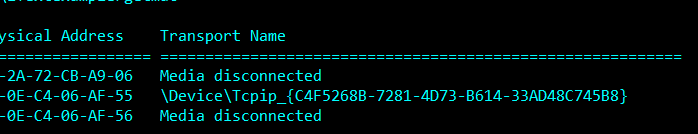
/NH Specifies that the "Column Header" should

not be displayed in the output.

Valid only for TABLE and CSV formats.

/V Specifies that verbose output is displayed.

/? Displays this help message.



**Pathping:**

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

Usage: pathping [-g host-list] [-h maximum\_hops] [-i address] [-n]

[-p period] [-q num\_queries] [-w timeout]

[-4] [-6] target\_name

Options:

-g host-list Loose source route along host-list.

-h maximum\_hops Maximum number of hops to search for target.

-i address Use the specified source address.

-n Do not resolve addresses to hostnames.

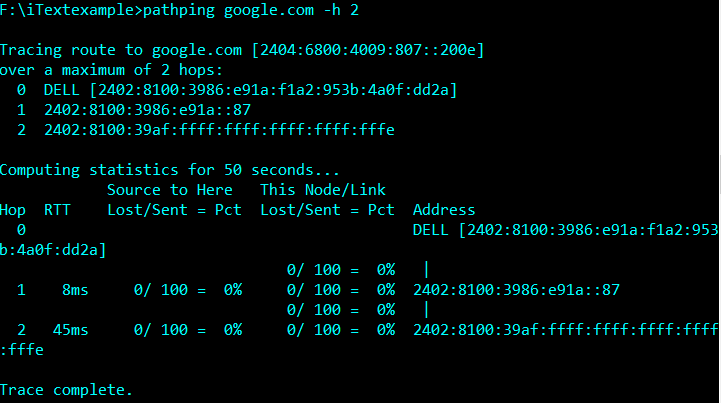
-p period Wait period milliseconds between pings.

-q num\_queries Number of queries per hop.

-w timeout Wait timeout milliseconds for each reply.

-4 Force using IPv4.

-6 Force using IPv6.



**Netstat:**

Displays protocol statistics and current TCP/IP network connections.

NETSTAT [-a] [-b] [-e] [-f] [-n] [-o] [-p proto] [-r] [-s] [-x] [-t] [interval]

-a Displays all connections and listening ports.

-e Displays Ethernet statistics. This may be combined with the –s option.

-f Displays Fully Qualified Domain Names (FQDN) for foreign addresses.

-n Displays addresses and port numbers in numerical form.

-o Displays the owning process ID associated with each connection.

-p proto Shows connections for the protocol specified by proto;

-r Displays the routing table.

-s Displays per-protocol statistics. By default, statistics are

shown for IP, IPv6, ICMP, ICMPv6, TCP, TCPv6, UDP, and UDPv6;

the -p option may be used to specify a subset of the default.

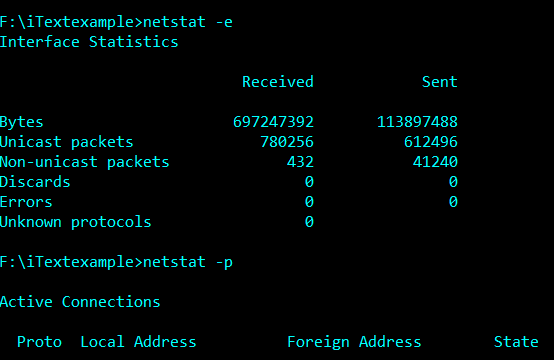
-t Displays the current connection offload state.

-x Displays NetworkDirect connections, listeners, and shared

endpoints.

-y Displays the TCP connection template for all connections.

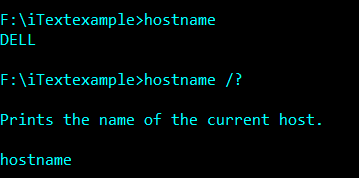
Cannot be combined with the other options.



**Hostname:**

Displays the host name portion of the full computer name of the computer

Hostname /?



**Nslookup:**

Displays information that you can use to diagnose Domain Name System (DNS) infrastructure. Before using this tool, you should be familiar with how DNS works. The nslookup command-line tool is available only if you have installed the TCP/IP protocol.

The nslookup command-line tool has two modes: interactive and noninteractive.

If you need to look up only a single piece of data, we recommend using the non-interactive mode. For the first parameter, type the name or IP address of the computer that you want to look up. For the second parameter, type the name or IP address of a DNS name server. If you omit the second argument, **nslookup** uses the default DNS name server.

If you need to look up more than one piece of data, you can use interactive mode. Type a hyphen (-) for the first parameter and the name or IP address of a DNS name server for the second parameter. If you omit both parameters, the tool uses the default DNS name server. While using the interactive mode, you can:

* Interrupt interactive commands at any time, by pressing CTRL+B
* Exit, by typing **exit**.
* Treat a built-in command as a computer name, by preceding it with the escape character (). An unrecognized command is interpreted as a computer name.

Commands: (identifiers are shown in uppercase, [] means optional)

NAME - print info about the host/domain NAME using default server

NAME1 NAME2 - as above, but use NAME2 as server

help or ? - print info on common commands

set OPTION - set an option

all - print options, current server and host

[no]debug - print debugging information

[no]d2 - print exhaustive debugging information

[no]defname - append domain name to each query

[no]recurse - ask for recursive answer to query

[no]search - use domain search list

[no]vc - always use a virtual circuit

domain=NAME - set default domain name to NAME

srchlist=N1[/N2/.../N6] - set domain to N1 and search list to N1,N2, etc.

root=NAME - set root server to NAME

retry=X - set number of retries to X

timeout=X - set initial time-out interval to X seconds

type=X - set query type (ex. A,AAAA,A+AAAA,ANY,CNAME,MX,NS,PTR,

SOA,SRV)

querytype=X - same as type

class=X - set query class (ex. IN (Internet), ANY)

[no]msxfr - use MS fast zone transfer

ixfrver=X - current version to use in IXFR transfer request

server NAME - set default server to NAME, using current default server

lserver NAME - set default server to NAME, using initial server

root - set current default server to the root

ls [opt] DOMAIN [> FILE] - list addresses in DOMAIN (optional: output to FILE)

-a - list canonical names and aliases

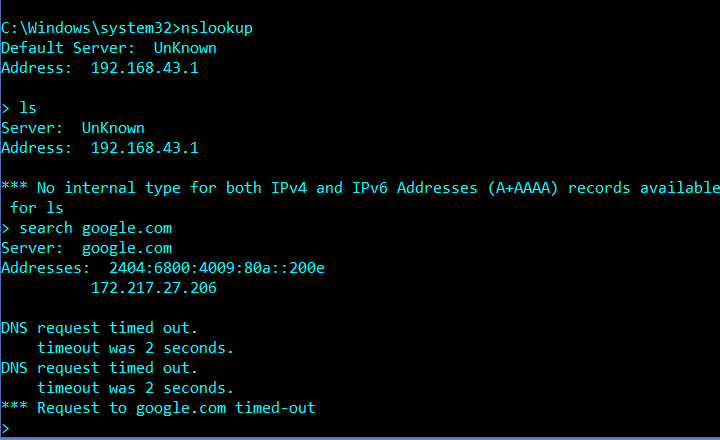
-d - list all records

-t TYPE - list records of the given RFC record type (ex. A,CNAME,MX,NS,

PTR etc.)

view FILE - sort an 'ls' output file and view it with pg

exit - exit the program



**Telnet:**

Communicates with a computer running the telnet Server service.

telnet [-a][-e escape char][-f log file][-l user][-t term][host [port]]

-a Attempt automatic logon. Same as -l option except uses

the currently logged on user's name.

-e Escape character to enter telnet client prompt.

-f File name for client side logging

-l Specifies the user name to log in with on the remote system.

Requires that the remote system support the TELNET ENVIRON option.

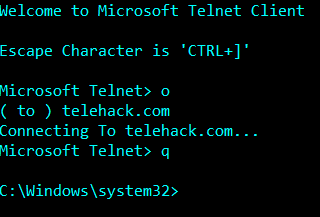
-t Specifies terminal type.

Supported term types are vt100, vt52, ansi and vtnt only.

host Specifies the hostname or IP address of the remote computer

to connect to.

port Specifies a port number or service name.



**Where:**

Displays the location of files that match the search pattern. By default, the search is done along the current directory and in the paths specified by the PATH environment variable.

Parameter List:

/R Recursively searches and displays the files that match the

given pattern starting from the specified directory.

/Q Returns only the exit code, without displaying the list

of matched files. (Quiet mode)

/F Displays the matched filename in double quotes.

/T Displays the file size, last modified date and time for all

matched files.

pattern Specifies the search pattern for the files to match.

Wildcards \* and ? can be used in the pattern. The

"$env:pattern" and "path:pattern" formats can also be

specified, where "env" is an environment variable and

the search is done in the specified paths of the "env"

environment variable. These formats should not be used

with /R. The search is also done by appending the

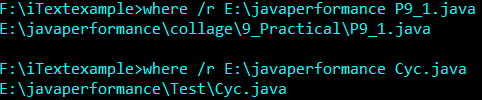
extensions of the PATHEXT variable to the pattern.

/? Displays this help message.

NOTE: The tool returns an error level of 0 if the search is

successful, of 1 if the search is unsuccessful and

of 2 for failures or errors.



**More:**

Displays output one screen at a time.

MORE [/E [/C] [/P] [/S] [/Tn] [+n]] < [drive:][path]filename

command-name | MORE [/E [/C] [/P] [/S] [/Tn] [+n]]

MORE /E [/C] [/P] [/S] [/Tn] [+n] [files]

[drive:][path]filename Specifies a file to display one

screen at a time.

command-name Specifies a command whose output

will be displayed.

/E Enable extended features

/C Clear screen before displaying page

/P Expand FormFeed characters

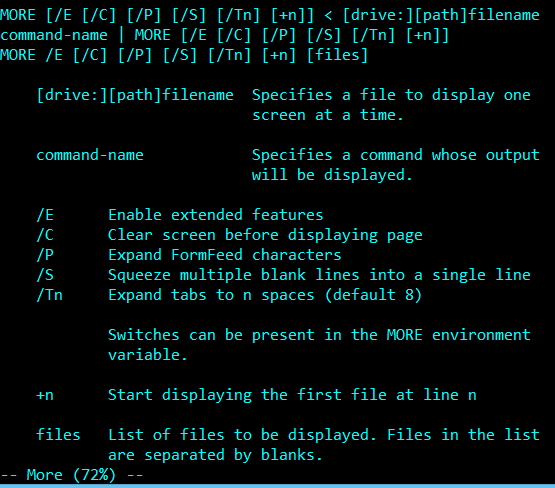
/S Squeeze multiple blank lines into a single line

/Tn Expand tabs to n spaces (default 8)

Switches can be present in the MORE environment

variable.

+n Start displaying the first file at line n



**Color:**

Sets the default console foreground and background colors.

COLOR [attr]

attr Specifies color attribute of console output

Color attributes are specified by TWO hex digits -- the first

corresponds to the background; the second the foreground. Each digit

can be any of the following values:

0 = Black 8 = Gray

1 = Blue 9 = Light Blue

2 = Green A = Light Green

3 = Aqua B = Light Aqua

4 = Red C = Light Red

5 = Purple D = Light Purple

6 = Yellow E = Light Yellow

7 = White F = Bright White

If no argument is given, this command restores the color to what it was

when CMD.EXE started. This value either comes from the current console

window, the /T command line switch or from the DefaultColor registry

value.

The COLOR command sets ERRORLEVEL to 1 if an attempt is made to execute

the COLOR command with a foreground and background color that are the

same.

**Practical - 2**

**AIM:** Configuration of Wireshark packet analyzer and capturing packets using TCPDUMP/Wiresharp.

Wireshark can capture traffic from many different network media types, including Ethernet, Wireless LAN, Bluetooth, USB, and more. The specific media types supported may be limited by several factors, including your hardware and operating system.

The following are some of the many features Wireshark provides:

Available for UNIX and Windows.

Capture live packet data from a network interface.

Open files containing packet data captured with tcpdump/WinDump, Wireshark, and many other packet capture programs.

Import packets from text files containing hex dumps of packet data.

Display packets with very detailed protocol information.

Save packet data captured.

Export some or all packets in a number of capture file formats.

Filter packets on many criteria.

Search for packets on many criteria.

Colorize packet display based on filters.

Create various statistics etc.

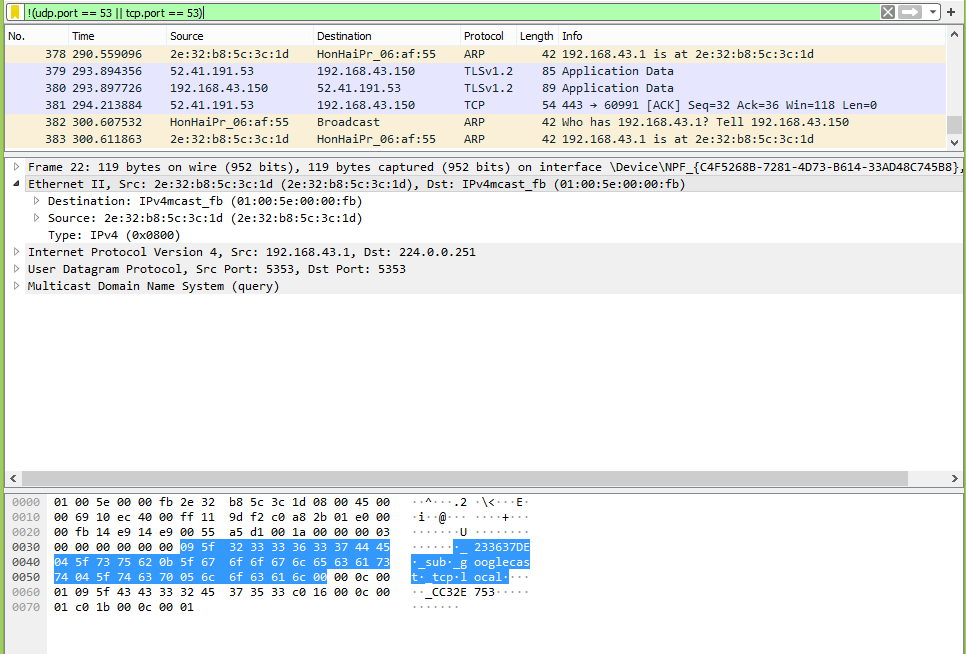


Fig 2.1

In Fig 2.1 as shown above we are applying filter of Non-DNS to capturing.

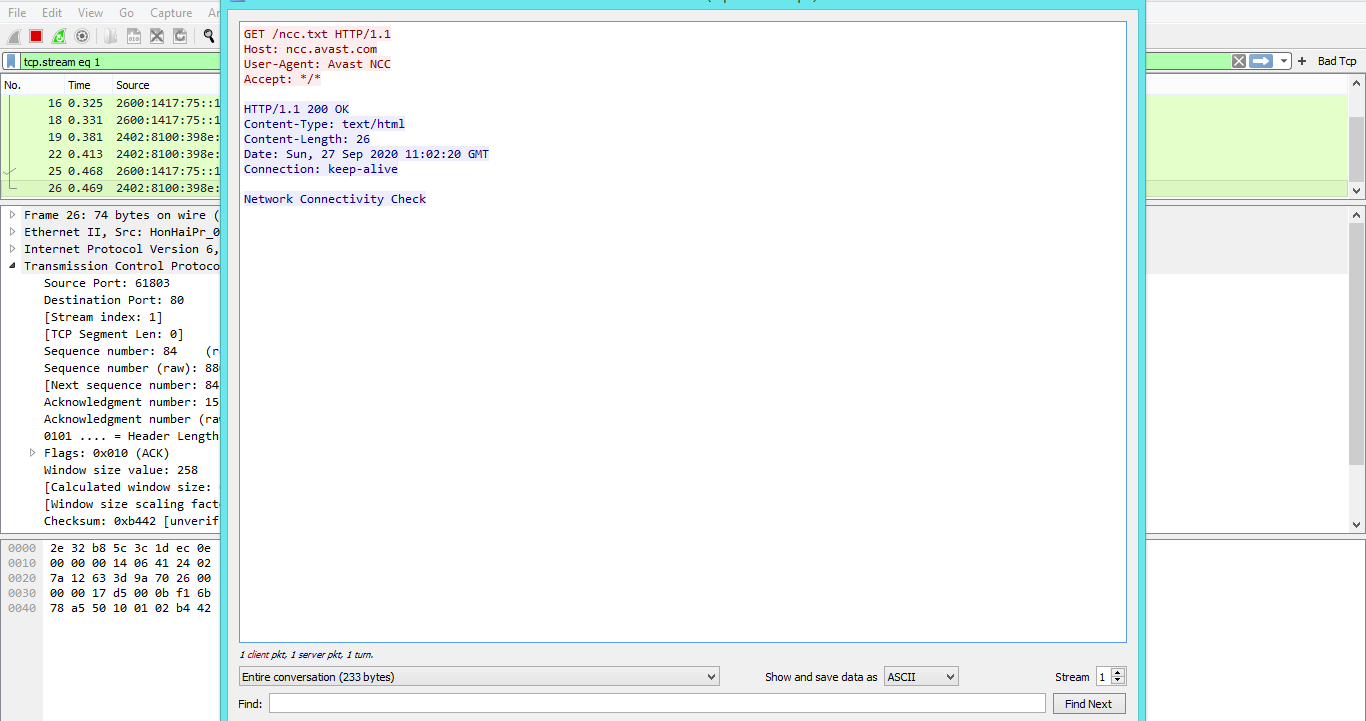


Fig 2.2

Analyzing tcp stream in Fig 2.2 its showing status HTTP 200 connection is ok, content length many other info.

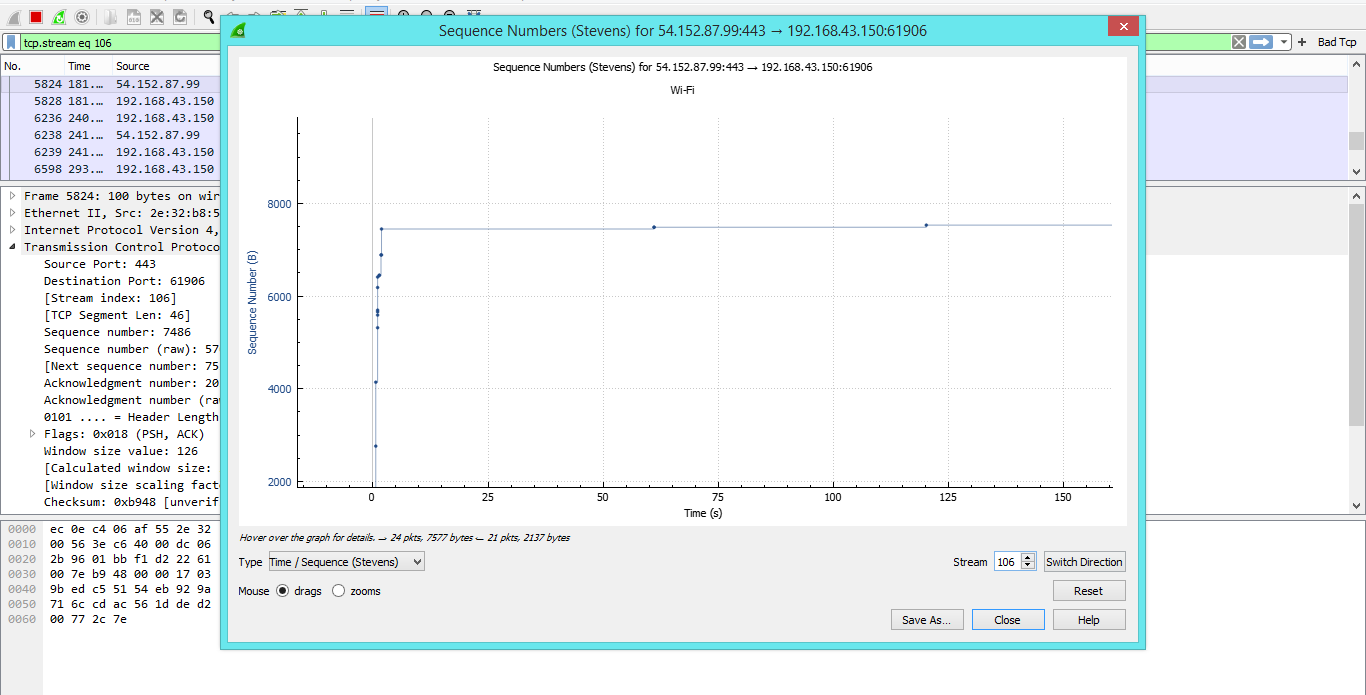


Fig 2.3

Fig 2.3 is tcp stream graph to analyze source to destination tcp stream.

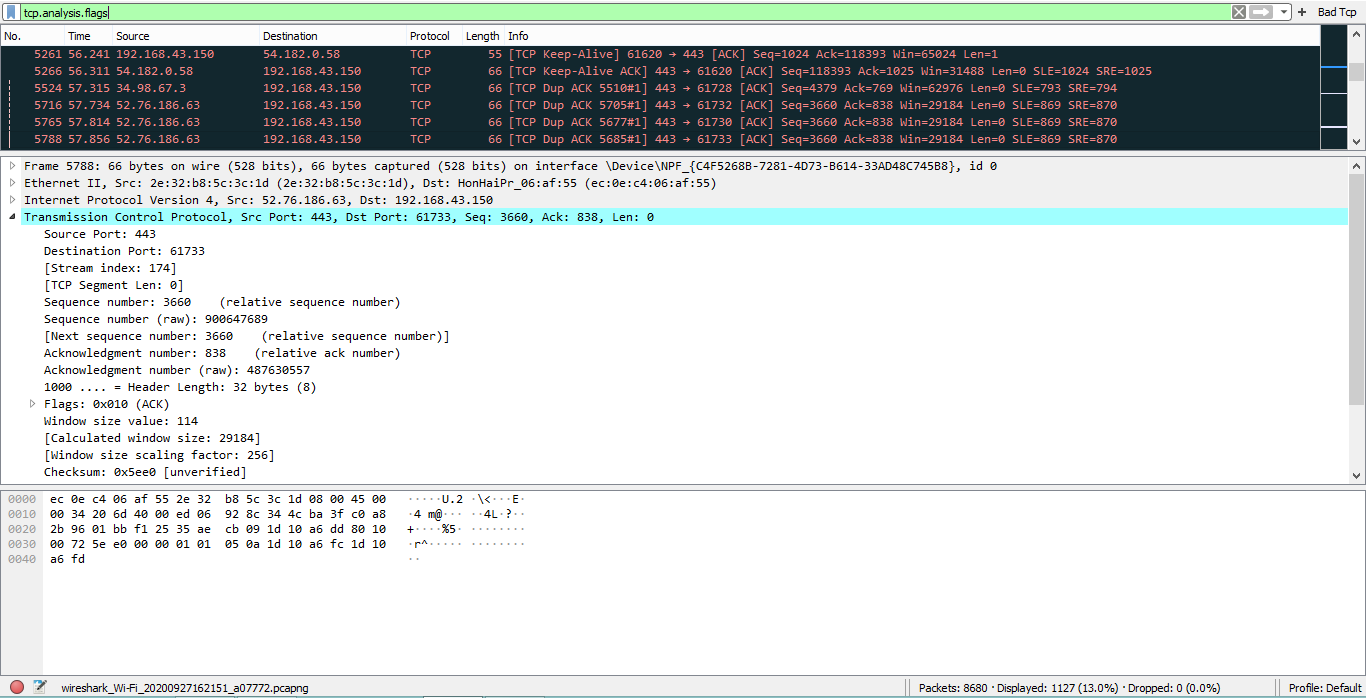


Fig 2.4

In Fig2.4 applying filter to analysis a tcp flags like retransmission, timeout, etc.

**Practical – 3**

**Aim:** Dissection of Packets using Wireshark.

Fig 3.1 show registration page for user. We are going to use this page for user taking details.

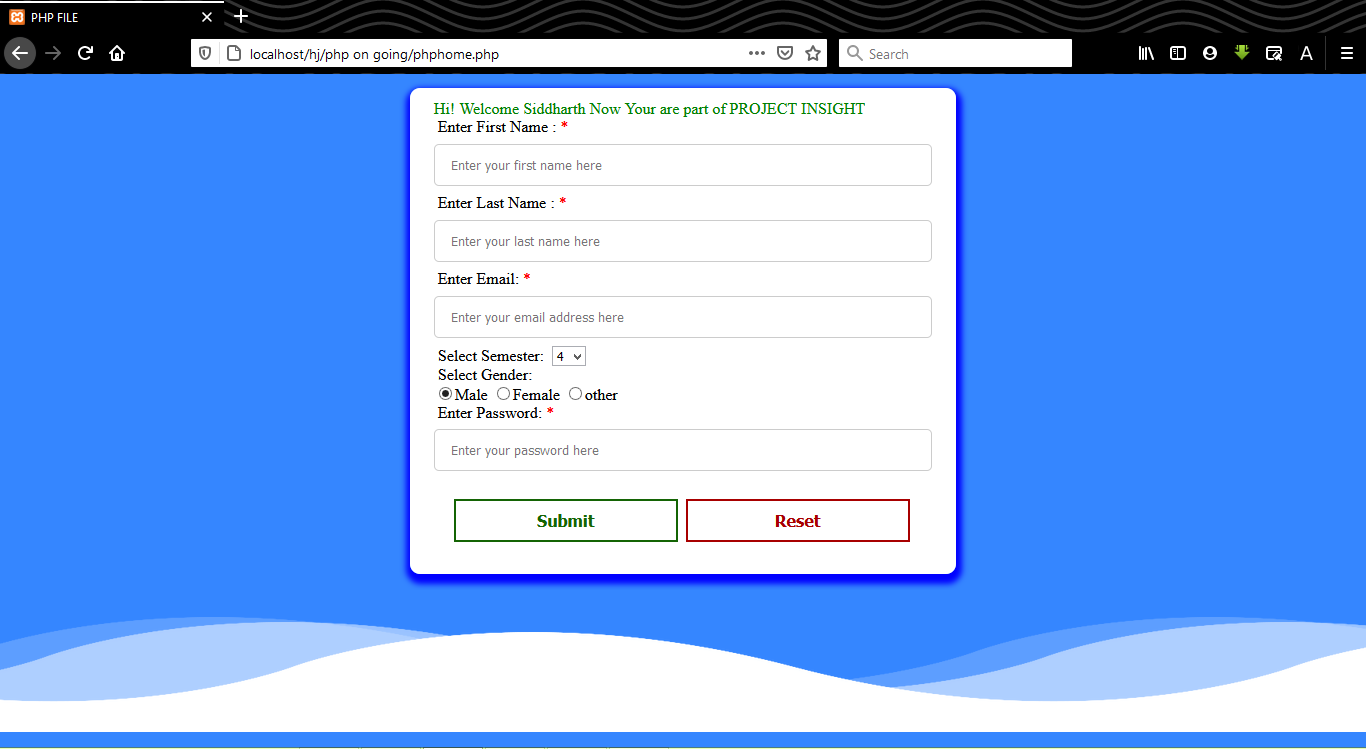


Fig 3.1

Fig 3.2 shows login page that we are going to dissection in wireshark

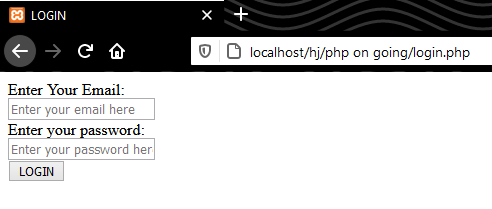


Fig 3.2

In Fig 3.4 & Fig 3.5 we capturing registration page and during dissection using filter “tcp.port == 80” of this we following tcp stream that show username, user email, password, etc.

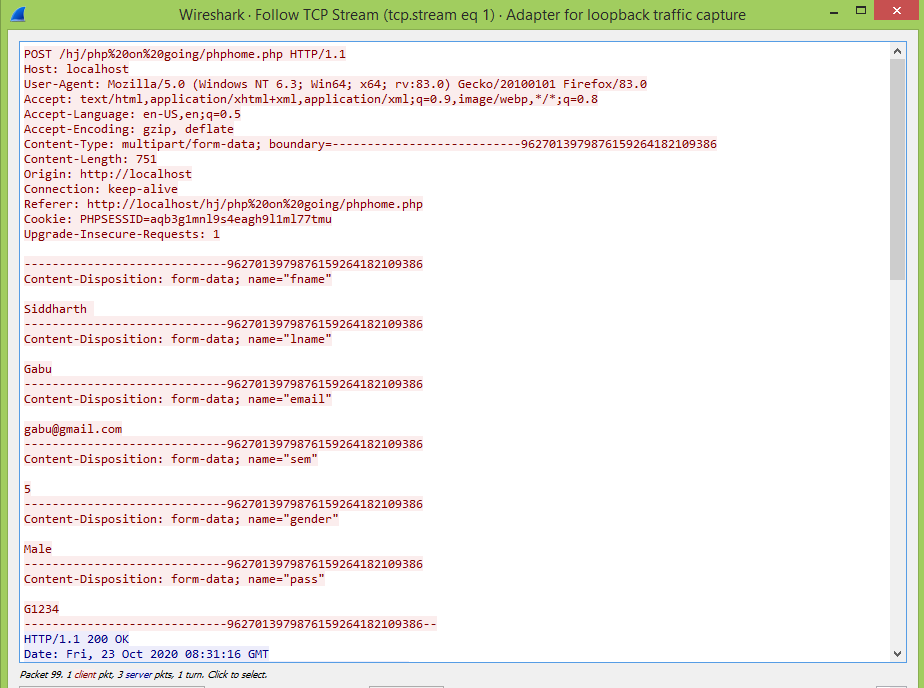


Fig 3.4



Fig 3.5

In Fig 3.6 & 3.7 we are using login page during dissection using only filter “http” of this we following that show username, password.

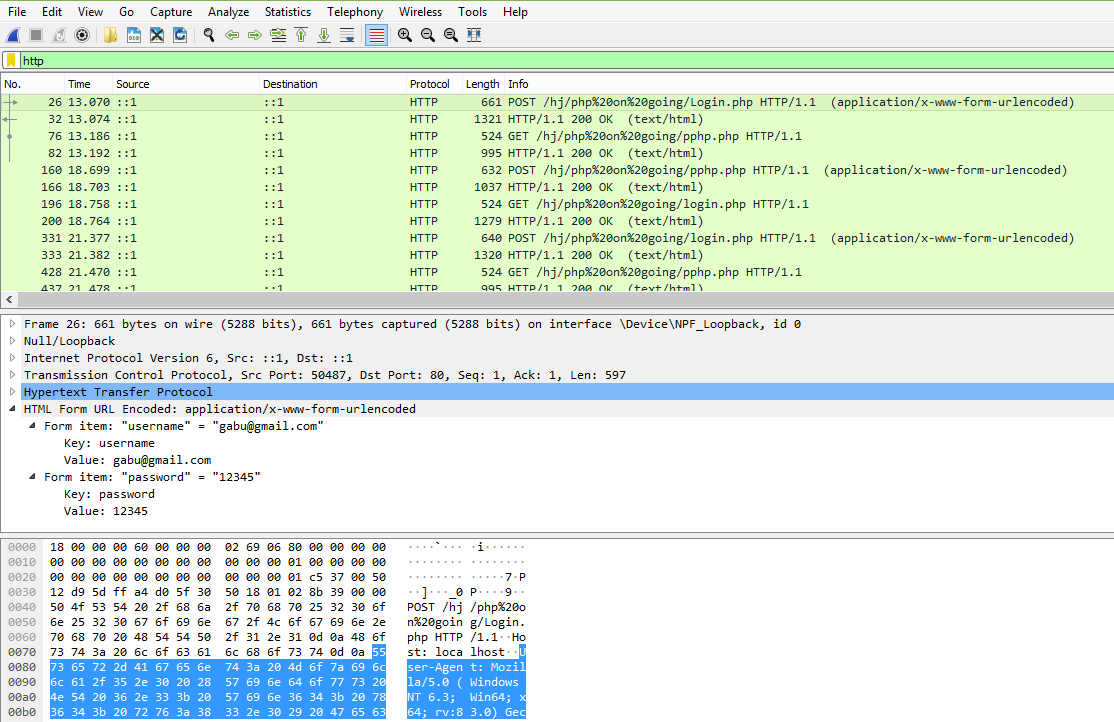


Fig 3.6

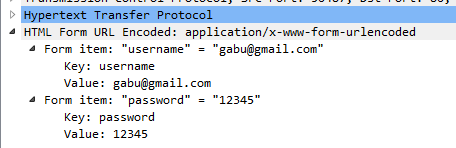


Fig 3.7

**Practical-4**

**Aim:** Simulating LAN configuration using the Cisco Packet tracer emulator.

Packet Tracer is a cross-platform visual simulation tool designed by Cisco Systems that allows users to create network topology and imitate modern computer network. The software allows users to simulate the configuration of Cisco routers and switches using a simulated command line interface. Packet Tracer makes use of a drag and drop user interface, allowing users to add and remove simulated network devices as they see fit. The software is mainly focused towards Certified Cisco Network Associate Academy students as an educational tool for helping them learn fundamental CCNA concepts. Previously students enrolled in a CCNA Academy program could freely download and use the tool free of charge for educational use.

Feature and Advantages of Cisco Packet Tracer :

• Provides a visual demonstration of complex technologies and configurations

• Allows instructors to author customized, guided activities that provide immediate feedback using the Activity Wizard

• Facilitates numerous learning activities such as lectures, individual and group lab activities, homework, assessments, games, network design, troubleshooting, modeling tasks, case studies, and competitions

• Enables visualization, animation, and detailed modeling for exploration, experimentation, and explanation

• Supports self-paced learning outside the classroom

• Supports social learning processes by enabling collaboration and competition

• Supports the majority of protocols and technologies taught in the following Networking Academy curricula: Cisco CCNA® Discovery, CCNA Exploration, and CCNA Security, and can be used to teach concepts from IT Essentials and Cisco CCNP® courses.

Cisco packet tracer supports following Protocols

|  |  |
| --- | --- |
| **Layer** | **Cisco Packet Tracer Supported Protocols** |
| **Application** | FTP , SMTP, POP3, HTTP, TFTP, Telnet, SSH, DNS, DHCP, NTP, SNMP, AAA, ISR VOIP, SCCP config and calls ISR command support, Call Manager Express |
| **Transport** | TCP and UDP, TCP Nagle Algorithm & IP Fragmentation, RTP |
| **Network** | BGP, IPv4, ICMP, ARP, IPv6, ICMPv6, IPSec, RIPv1/ v2/ng, Multi-Area OSPF, EIGRP, Static Routing, Route Redistribution, Multilayer Switching, L3 QoS, NAT, CBAL , Zone-based policy firewall and Intrusion Protection System on the ISR, GRE VPN, IPSec VPN |
| **Network Access / Interface** | Ethernet (802.3), 802.11, HDLC, Frame Relay, PPP, PPPoE, STP, RSTP, VTP, DTP, CDP, 802.1q, PAgP, L2 QoS, SLARP, Simple WEP, WPA, EAP |

WorkSpace :

Cisco Packet Tracer has two workspaces—logical and physical.

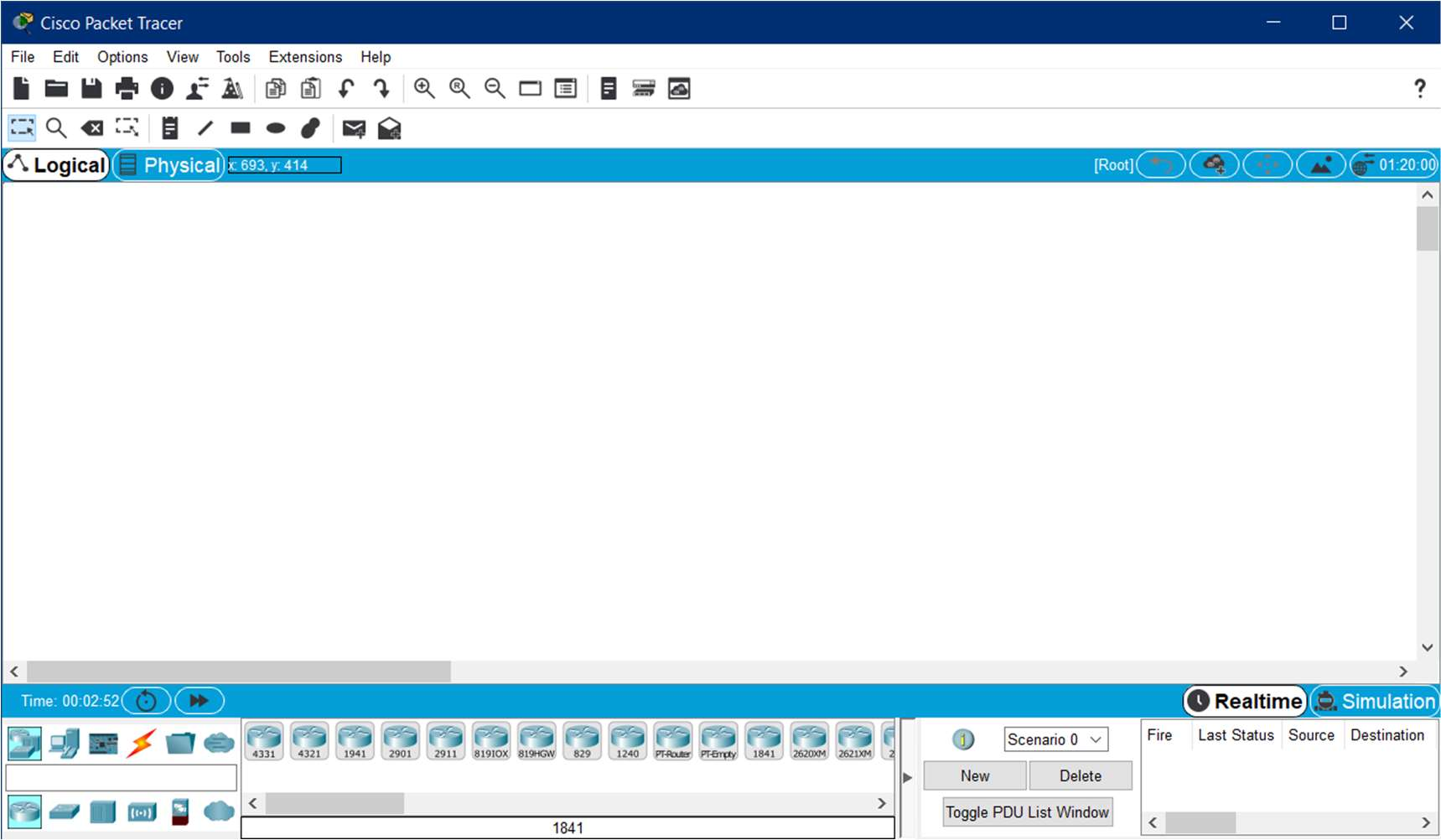
1 ) Logical Workspace :

The logical workspace allows users to build logical network topologies by placing, connecting, and clustering virtual network devices.

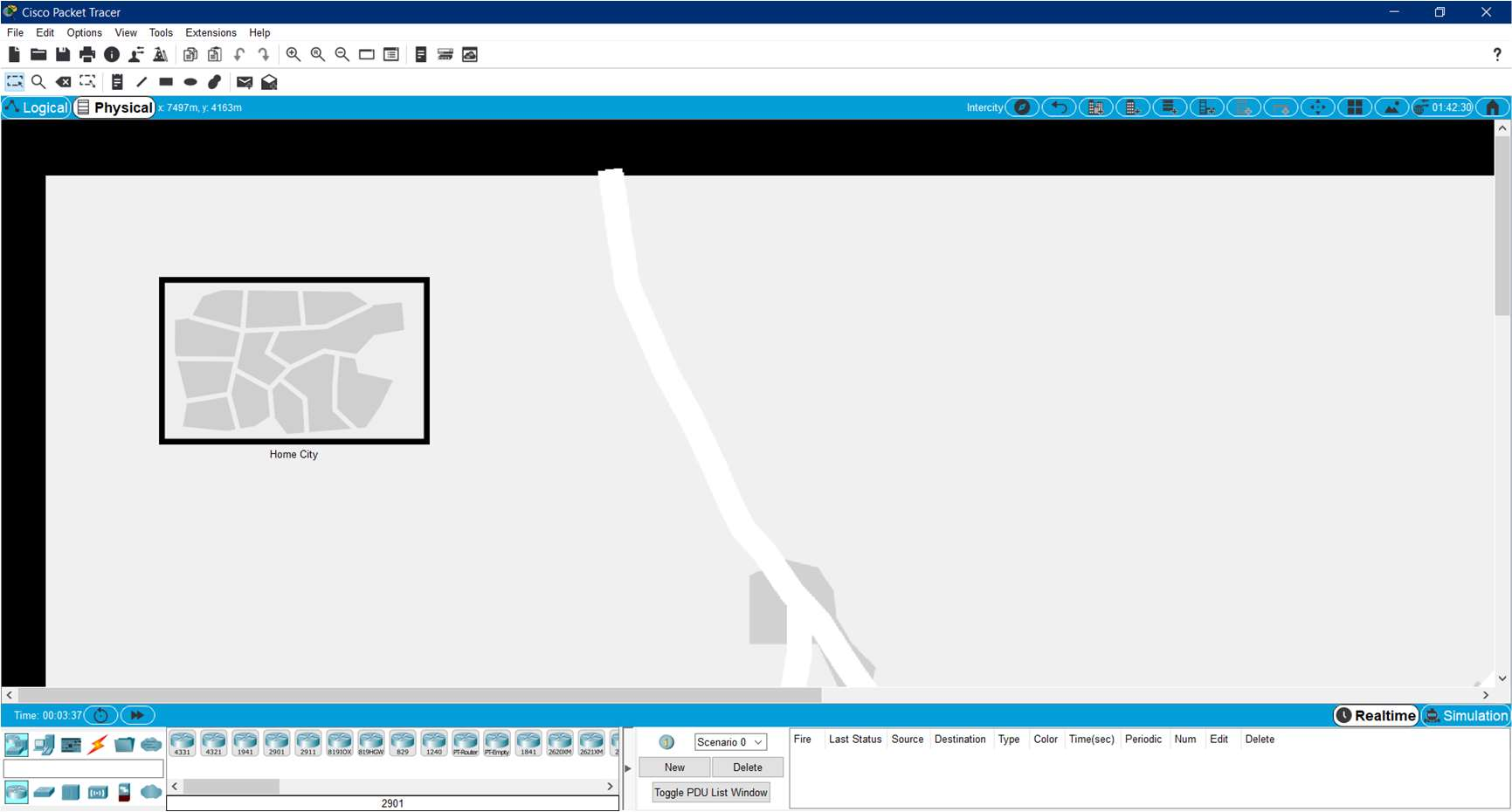
2 ) Physical Workspace :

The physical workspace provides a graphical physical dimension of the logical network, giving a sense of scale and placement in how network devices such as routers, switches, and hosts would look in a real environment. The physical view also provides geographic representations of networks, including multiple cities, buildings, and wiring closets.

Logical Workspace :



Physical Workspace :



Additional Features of Cisco Packet Tracer :

* Lab grading function
* International language support
* Compatible with the following platforms: Windows, Windows XP; Vista (Vista Basic, Vista Premium); Windows 7; and Linux (Ubuntu, Fedora)
* Available to registered Networking Academy instructors, students, and alumni

**Star Topology :**

A star topology is a topology for a Local Area Network (LAN) in which all nodes are individually connected to a central connection point, like a hub or a switch. A star takes more cable than e.g. a bus, but the benefit is that if a cable fails, only one node will be brought down.

All traffic emanates from the hub of the star. The central site is in control of all the nodes attached to it. The central hub is usually a fast, self contained computer and is responsible for routing all traffic to other nodes. The main advantages of a star network is that one malfunctioning node does not affect the rest of the network. However this type of network can be prone to bottleneck and failure problems at the central site.

A star network is often combined with a bus topology. The central hub is then connected to the backbone of the bus. This combination is called a tree.

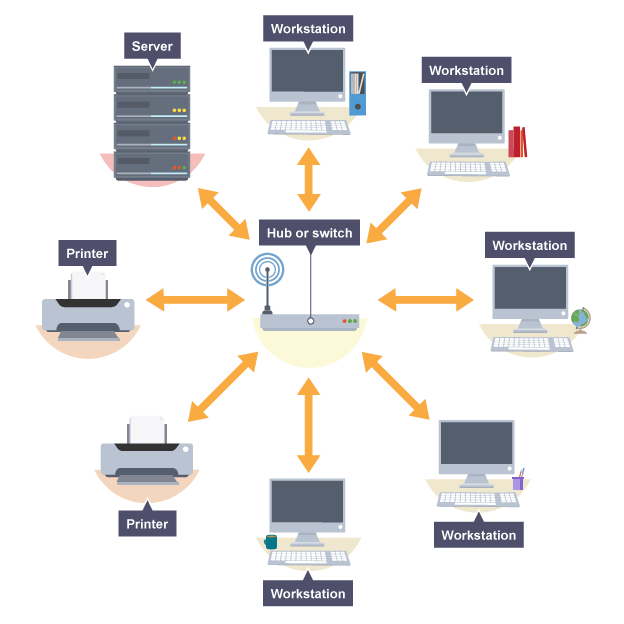
Advantages and disadvantages of a star network :

-->The advantages of a star network are:

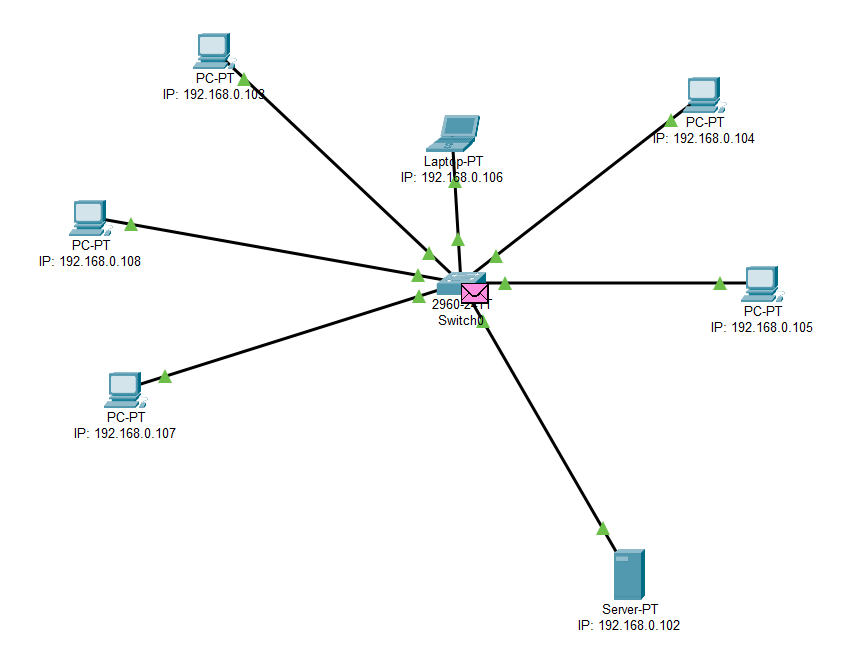
* it is very reliable – if one cable or device fails then all the others will continue to work
* it is high-performing as no data collisions can occur

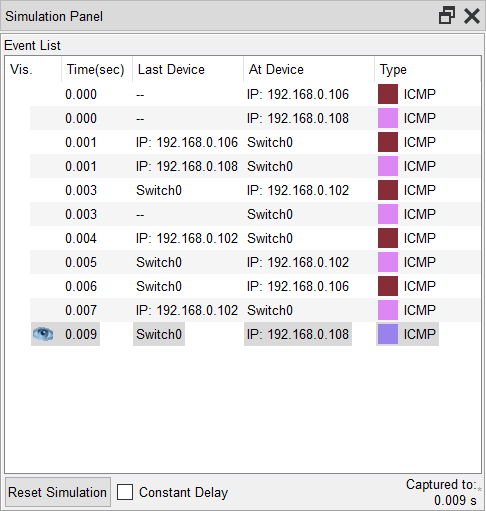
-->The disadvantages of a star network are:

* it is expensive to install as this type of network uses the most cable (network cable is expensive)
* extra hardware is required (hubs or switches) which adds to cost
* if a hub or switch fails, all the devices connected to it will have no network connection

**Here is the picture of Star Topology in General Type .**

**Simulation of Star Topology in Cisco Packet Tracer**



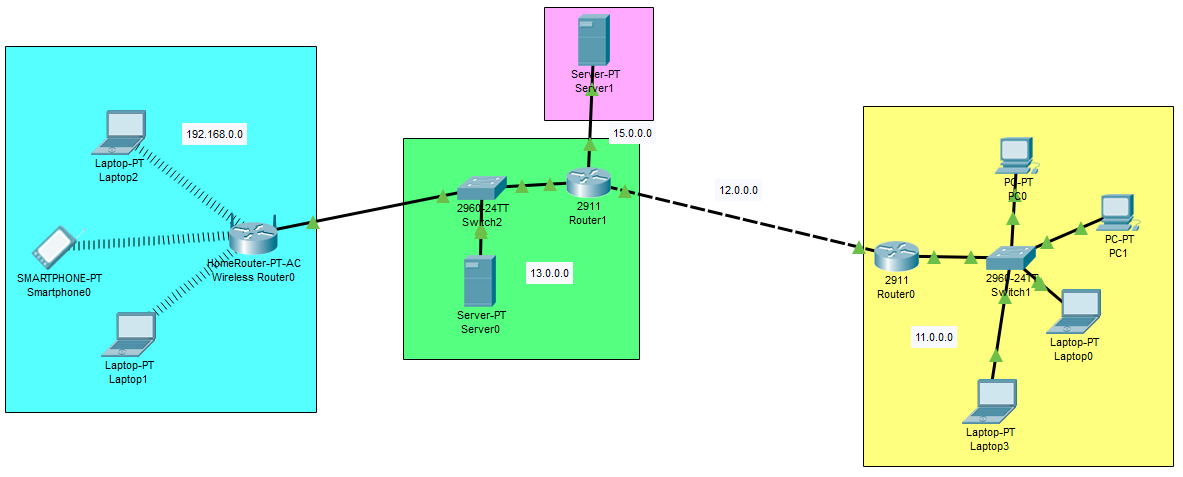


* Here in the Simulation Laptop with IP address of 192.168.0.106 wants to communicate with Server with IP address of 192.168.0.102.
* Because laptop and server both are connected with a common device i.e switch, the message will be passed via switch.
* Hence , message will leave laptop and then will be received by switch,which then will be passed to the server.Server will follow same method for giving acknowledgement back to laptop.

Hence , Cisco Packet Tracer allows us to do rea time networking in simulation mode so we can get the idea if we put this type of method or network in real world what will happen.

**Practical - 5**

**AIM:** Design a heterogeneous network which consist of wired and wireless LAN connected using router. Create a file server which allows user to download and upload the files.

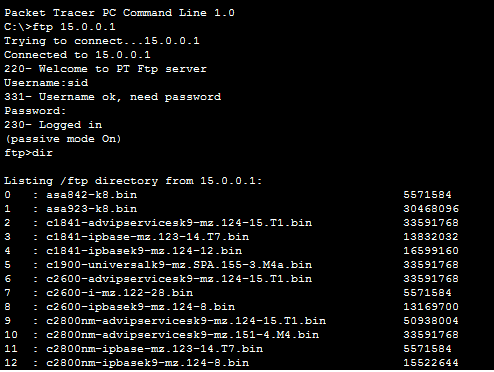


Wireless LAN Fig5.1 Wired LAN

**File SERVER:**

Fig5.2

**Establish connection from wired LAN’s PC to server, list out directories on server and upload the file ‘testfile.txt’:**

****

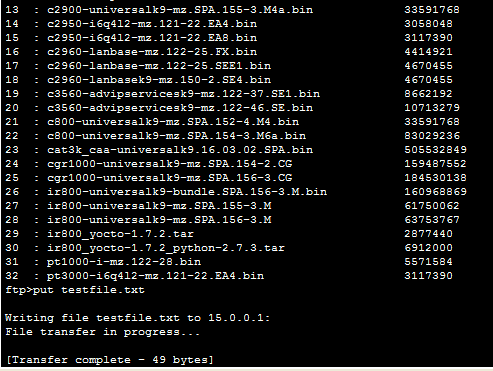
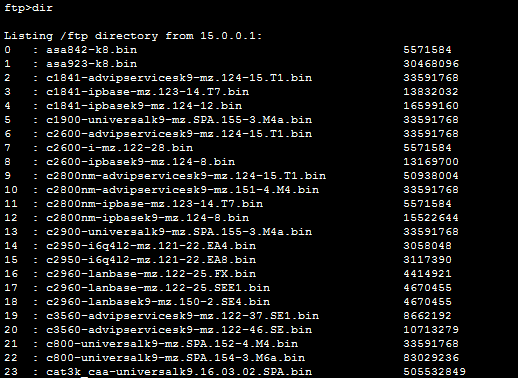
****

Fig5.3

**Now check uploaded file is there:**

****

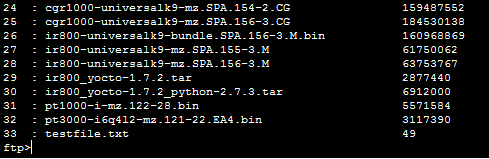
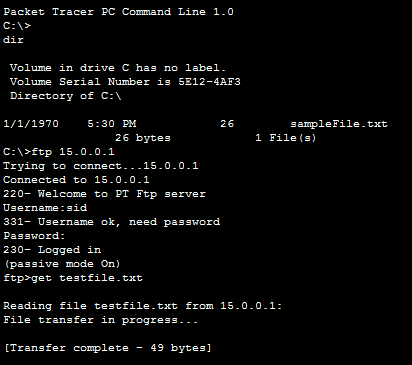
****

Fig5.4

**Establish connection from wireless LAN’s PC to server and download file ‘testfile.txt’:**

****

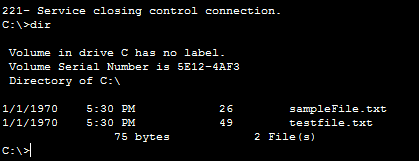
****

Fig5.5

**Practical - 6**

**Aim: You have sub-netted your class C network 192.168.1.0 with a subnetmask of 255.255.255.252. Please list the following:**

**1)number of networks**

**2)number of hosts per network**

**3)the full range of the**

**4)first three networks, and the usable address range from those first three networks.**

**Additionally, identify the broadcast addresses for each network. Design this network in a packet tracer and simulate the working of the network. Save the Pkt file of your network design**

Ip : 192.168.1.0 ->192.168.1.00000000

Now subnet mask is 255.255.255.252 -> 255.255.255.11111100

**192.168.1.000000**​ **00**​

**->subnet network id(6 bits)**​

**->host id(2 bits)**

**Number of Sub- Networks: 2^6 = 64**

**Number Host per subnet : 2^2-2 = 2**

**first three networks:**

|  |  |  |
| --- | --- | --- |
| **Subnet id** | **Usable Addresses** | **Broadcast id** |
| 192.168.1.0 | 192.168.1.1  192.168.1.2 | 192.168.1.3 |
| 192.168.1.4 | 192.168.1.5  192.168.1.6 | 192.168.1.7 |
| 192.168.1.8 | 192.168.1.9  192.168.1.10 | 192.168.1.11 |

Design Network in Cisco Packet Tracer for Email Server and Configure Devices for same. Show in Fig6.1

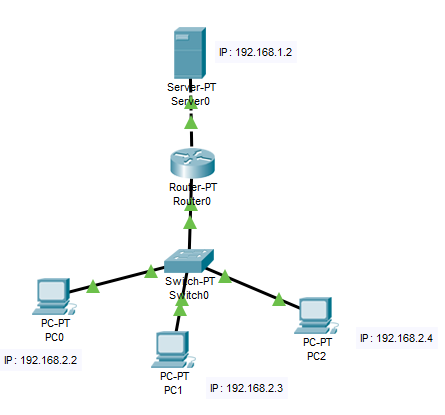


Fig6.1

Setting up Email configuration in on of network devices. Shown in Fig7.2

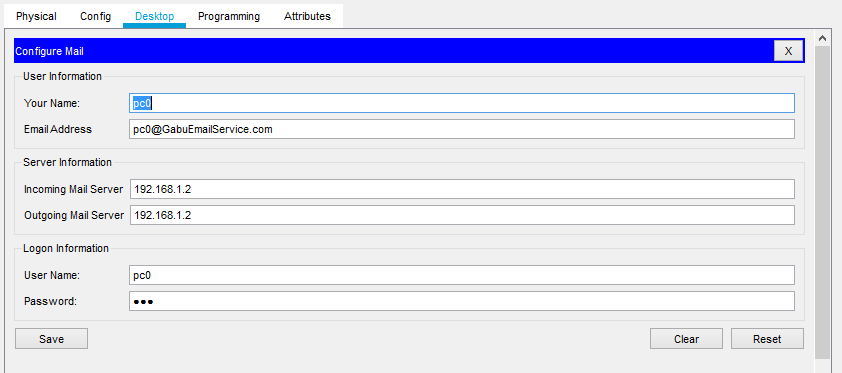


Fig6.2

Composing mail from pc0 to pc1. Show in Fig6.3

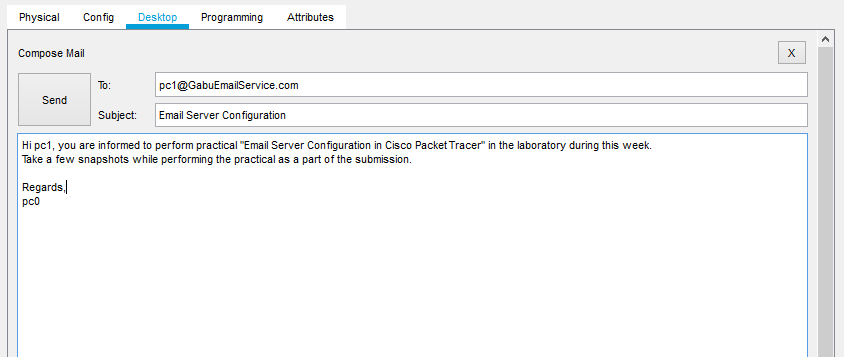


Fig6.3

Checking mail in pc1 that successfully send from pc0. Shown in Fig6.4

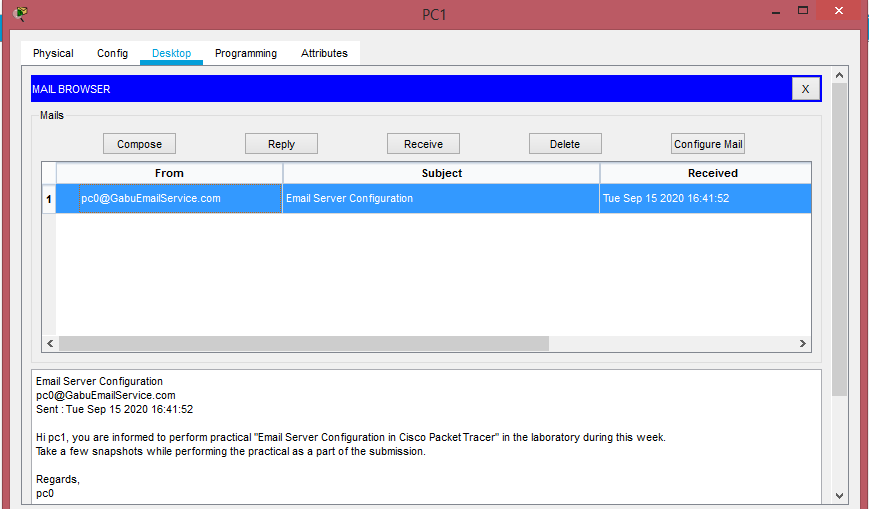


Fig6.4

Checking mail in pc2 that successfully send from pc1. Shown in Fig6.5

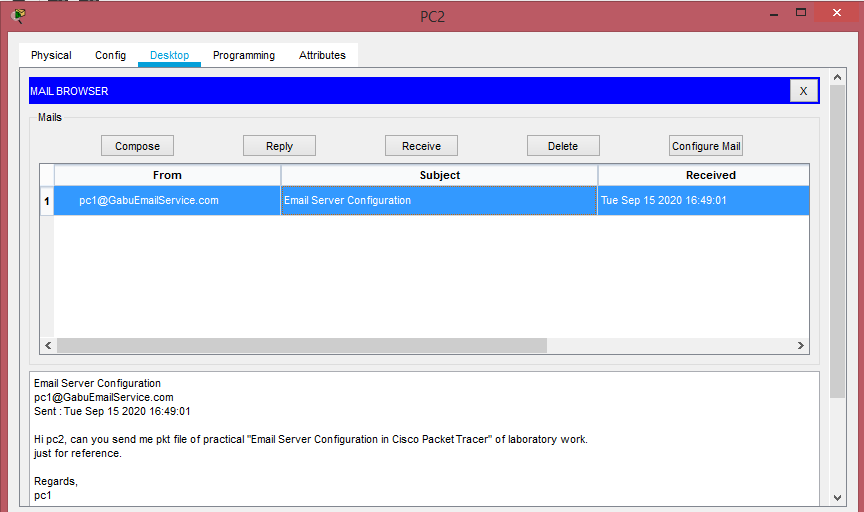


Fig6.5

**Practical - 7**

**AIM**: Router Configuration in Cisco Packet Tracer

Design Network in Cisco Packet Tracer for Router Configuration and Configure Devices for same. Show in Fig7.1

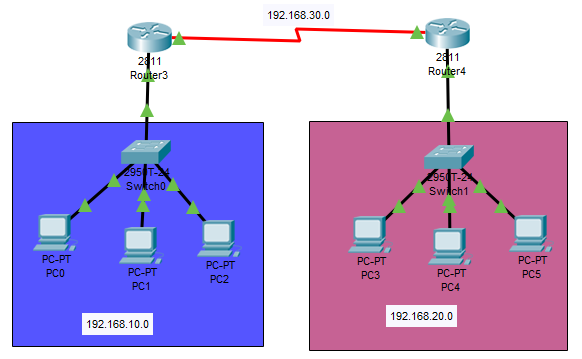
****

Fig7.1

Configuring Router to connect two different networks. Shown in Fig7.2

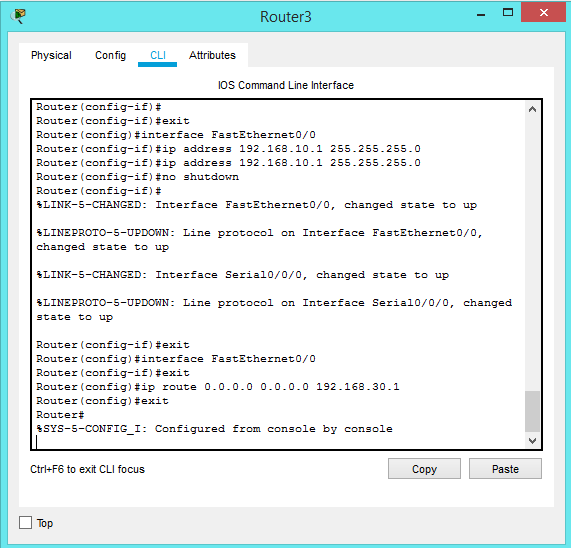


Fig7.2

After configuring Router checking ping command to ping another devices on network. Show in Fig7.3

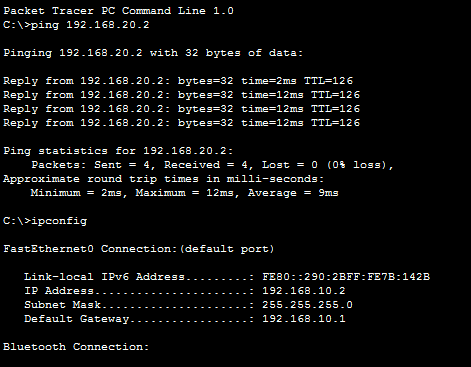


Fig7.3

Sending Packet from one device to another device. Show in Fig7.4

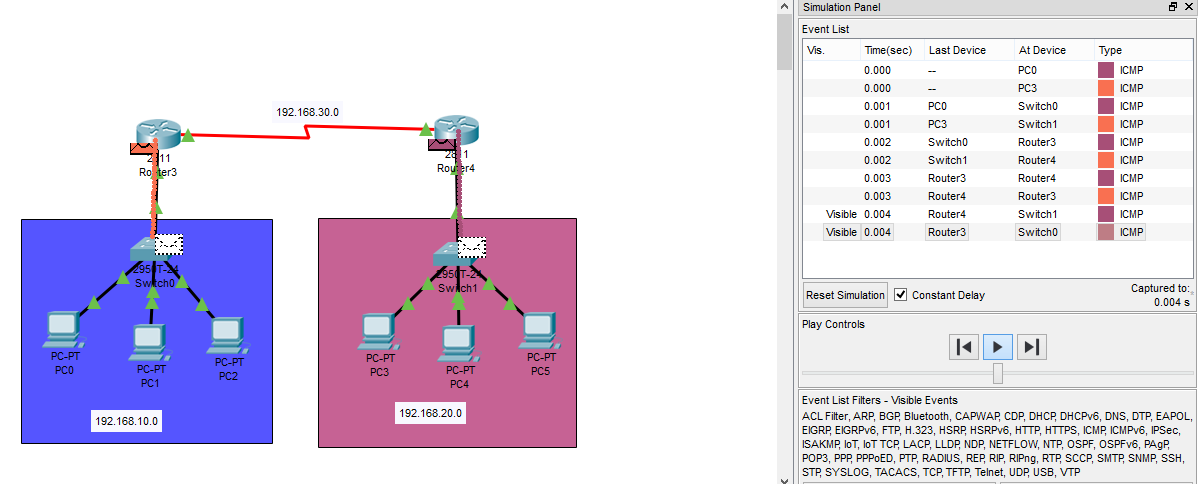


Fig7.4

**Practical - 8**

**AIM:** Implementing echo sever using Socket programming.

**SERVER:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

void error(const char \*msg){

perror(msg);

exit(1);

}

int main(int argc,char \*argv[]){

if(argc<2){

fprintf(stderr,"port not provided. program terminated\n");

exit(1);

}

int sockfd, newsockfd, portno, n, clilen;

char buffer[256];

struct sockaddr\_in serv\_addr,cli\_addr;

sockfd = socket(AF\_INET,SOCK\_STREAM,0);

if(sockfd<0){

error("Error Openinn Socket.");

}

memset(&serv\_addr,0,sizeof(struct sockaddr\_in));

portno = atoi(argv[1]);

serv\_addr.sin\_family = AF\_INET;

serv\_addr.sin\_addr.s\_addr = INADDR\_ANY;

serv\_addr.sin\_port = htons(portno);

if(bind(sockfd,(struct sockaddr \*)&serv\_addr,sizeof(serv\_addr))<0)

error("Binding Faild.");

listen(sockfd, 5);

clilen = sizeof(cli\_addr);

newsockfd = accept(sockfd,(struct sockaddr \*)&cli\_addr,&clilen);

if(newsockfd<0)error("Error on Accept\n");

/\* \*\*\*\*\*\*\*above code is for connection, only do pretty stuff below of this\*\*\*\*\*\*\* \*/

while(1){

memset(buffer,0,256);

n = read(newsockfd,buffer,255);

if(n<0)error("Error on reading");

printf("Client: %s",buffer);

memset(buffer,0,256);

printf("Server: ");

fgets(buffer,255,stdin);

n = write(newsockfd,buffer,strlen(buffer));

if(n<0)error("Erroe on writing");

int i = strncmp("Bye",buffer,3);

if(i==0)break;

}

//Done

close(newsockfd);

close(sockfd);

return 0;

}

**CLIENT:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <netdb.h>

void error(const char \*msg){

perror(msg);

exit(1);

}

int main(int argc,char \*argv[]){

if(argc<3){

fprintf(stderr,"usage %s hostname port\n",argv[0]);

exit(1);

}

int sockfd, portno, n;

char buffer[256];

struct sockaddr\_in serv\_addr;

struct hostent \*server;

sockfd = socket(AF\_INET,SOCK\_STREAM,0);

if(sockfd<0){

error("Error Openinn Socket.");

}

memset(&serv\_addr,0,sizeof(serv\_addr));

portno = atoi(argv[2]);

server = gethostbyname(argv[1]);

if(server == NULL){

fprintf(stderr,"Error , no such host");

exit(1);

}

serv\_addr.sin\_family = AF\_INET;

bcopy((char \*)server->h\_addr,(char \*)&serv\_addr.sin\_addr.s\_addr,server->h\_length);

serv\_addr.sin\_port = htons(portno);

if(connect(sockfd,(struct sockaddr \*)&serv\_addr,sizeof(serv\_addr))<0)error("connection faild");

while(1){

memset(buffer,0,256);

printf("Client: ");

fgets(buffer,255,stdin);

n = write(sockfd,buffer,strlen(buffer));

if(n<0)error("Erroe on writing");

memset(buffer,0,256);

n = read(sockfd,buffer,255);

if(n<0)error("Error on reading");

printf("Server: %s",buffer);

int i = strncmp("Bye",buffer,3);

if(i==0)break;

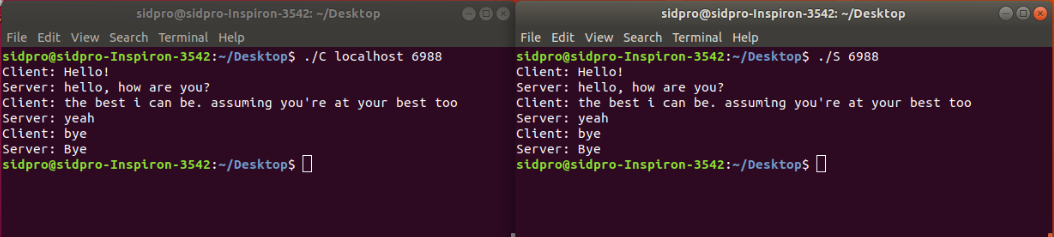
}

//Done

close(sockfd);

return 0;

}

****

**Practical - 9**

**AIM:** Implementing file server using socket programming.

**SERVER:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <ctype.h>

void error(const char \*msg){

perror(msg);

exit(1);

}

int main(int argc,char \*argv[]){

if(argc<2){

fprintf(stderr,"port not provided. program terminated\n");

exit(1);

}

int sockfd, newsockfd, portno, n, clilen;

char buffer[256];

struct sockaddr\_in serv\_addr,cli\_addr;

sockfd = socket(AF\_INET,SOCK\_STREAM,0);

if(sockfd<0){

error("Error Openinn Socket.");

}

memset(&serv\_addr,0,sizeof(struct sockaddr\_in));

portno = atoi(argv[1]);

serv\_addr.sin\_family = AF\_INET;

serv\_addr.sin\_addr.s\_addr = INADDR\_ANY;

serv\_addr.sin\_port = htons(portno);

if(bind(sockfd,(struct sockaddr \*)&serv\_addr,sizeof(serv\_addr))<0)

error("Binding Faild.");

listen(sockfd, 5);

clilen = sizeof(cli\_addr);

newsockfd = accept(sockfd,(struct sockaddr \*)&cli\_addr,&clilen);

if(newsockfd<0)error("Error on Accept\n");

/\* \*\*\*\*\*\*\*above code is for connection only, do pretty stuff below of this\*\*\*\*\*\*\* \*/

FILE \*fp;

int words;

int word = 0;

fp = fopen("testrecivied.txt","a");

char ch;

read(newsockfd,&words,sizeof(int));

while(word!=words){

read(newsockfd,&ch,sizeof(char));

fprintf(fp,"%c",ch);

word++;

}

printf("\nThe file has been successfully reivied. It saved by name \" testrecivied.txt\" \n");

//Done

close(newsockfd);

close(sockfd);

return 0;

}

**CLIENT:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <netdb.h>

#include <ctype.h>

void error(const char \*msg){

perror(msg);

exit(1);

}

int main(int argc,char \*argv[]){

if(argc<3){

fprintf(stderr,"usage %s hostname port\n",argv[0]);

exit(1);

}

int sockfd, portno, n;

char buffer[256];

struct sockaddr\_in serv\_addr;

struct hostent \*server;

sockfd = socket(AF\_INET,SOCK\_STREAM,0);

if(sockfd<0){

error("Error Openinn Socket.");

}

memset(&serv\_addr,0,sizeof(serv\_addr));

portno = atoi(argv[2]);

server = gethostbyname(argv[1]);

if(server == NULL){

fprintf(stderr,"Error , no such host");

exit(1);

}

serv\_addr.sin\_family = AF\_INET;

bcopy((char \*)server->h\_addr,(char \*)&serv\_addr.sin\_addr.s\_addr,server->h\_length);

serv\_addr.sin\_port = htons(portno);

if(connect(sockfd,(struct sockaddr \*)&serv\_addr,sizeof(serv\_addr))<0)error("connection faild");

/\* \*\*\*\*\*\*\*above code is for connection, only do pretty stuff below of this\*\*\*\*\*\*\* \*/

FILE \*fp;

int words = 0;

fp = fopen("test.txt","r");

char ch;

while((ch=fgetc(fp))!=EOF){

words++;

}

printf("Words: %d\n",words);

write(sockfd,&words,sizeof(int));

rewind(fp);

char cp;

while((cp=fgetc(fp))!=EOF){

write(sockfd,&cp,sizeof(char));

}

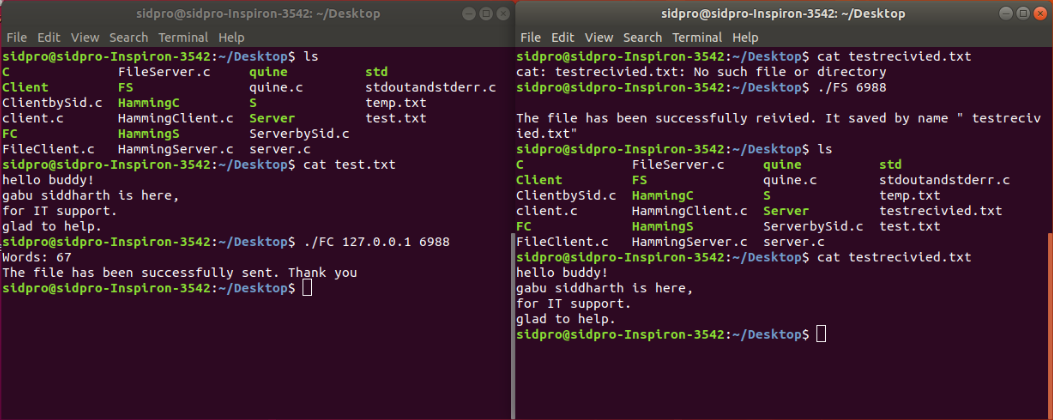
printf("The file has been successfully sent. Thank you\n");

//Done

close(sockfd);

return 0;

}



**Practical - 10**

AIM: Implementing Error Detection mechanism using Socket Programming.

**SERVER:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

void error(const char \*msg){

perror(msg);

exit(1);

}

int main(int argc,char \*argv[]){

if(argc<2){

fprintf(stderr,"port not provided. program terminated\n");

exit(1);

}

int sockfd, portno, n;

socklen\_t clilen;

char buffer[256];

memset(buffer,0,256);

struct sockaddr\_in serv\_addr,cli\_addr;

sockfd = socket(AF\_INET,SOCK\_DGRAM,0);

if(sockfd<0){

error("Error Openinn Socket.");

}

memset(&serv\_addr,0,sizeof(struct sockaddr\_in));

portno = atoi(argv[1]);

serv\_addr.sin\_family = AF\_INET;

serv\_addr.sin\_addr.s\_addr = INADDR\_ANY;

serv\_addr.sin\_port = htons(portno);

if(bind(sockfd,(struct sockaddr \*)&serv\_addr,sizeof(serv\_addr))<0)

error("Binding Faild.");

clilen = sizeof(cli\_addr);

//recvfrom(sockfd,buffer,255,0,(struct sockaddr \*)&cli\_addr,&clilen);

//printf("\nClient : %s",buffer);

/\* \*\*\*\*\*\*\*above code is for connection, only do pretty stuff below of this\*\*\*\*\*\*\* \*/

int data[10];

int test[10];

int p,p1,p2,p4;

printf("The data received is: ");

for(int i=0;i<7;i++){

n = recvfrom(sockfd,&data[i],sizeof(data[i]),0,(struct sockaddr \*)&cli\_addr,&clilen);

if(n<0)printf("Error while receiving");

printf("%d",data[i]);

}

printf("\nEnter data to tested: \n");

for(int i=0;i<7;i++){

scanf("%d",&test[i]);

}

p1= test[6]^test[4]^test[2]^test[0];

p2= test[5]^test[4]^test[1]^test[0];

p4= test[3]^test[2]^test[1]^test[0];

p = (4\*p4)+(2\*p2)+p1;

printf("The data for testing is: ");

for(int i=0;i<7;i++){

printf("%d",test[i]);

}

if(p==0){printf("No Error");}

else{

printf("\nThe Error is at position %d",p);

printf("\nThe correct data is:");

if(test[7-p]==0)

test[7-p]=1;

else

test[7-p]=0;

for(int i=0;i<7;i++){

printf("%d",test[i]);

}

}

printf("\n");

//Done

//close(newsockfd);

close(sockfd);

return 0;

}

**CLIENT:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <netdb.h>

void error(const char \*msg){

perror(msg);

exit(1);

}

int main(int argc,char \*argv[]){

if(argc<3){

fprintf(stderr,"usage %s hostname port\n",argv[0]);

exit(1);

}

int sockfd, portno, n;

char buffer[256];

socklen\_t len;

struct sockaddr\_in serv\_addr;

struct hostent \*server;

sockfd = socket(AF\_INET,SOCK\_DGRAM,0);

if(sockfd<0){

error("Error Openinn Socket.");

}

memset(&serv\_addr,0,sizeof(serv\_addr));

portno = atoi(argv[2]);

server = gethostbyname(argv[1]);

if(server == NULL){

fprintf(stderr,"Error , no such host");

exit(1);

}

serv\_addr.sin\_family = AF\_INET;

bcopy((char \*)server->h\_addr,(char \*)&serv\_addr.sin\_addr.s\_addr,server->h\_length);

serv\_addr.sin\_port = htons(portno);

len = sizeof(serv\_addr);

//sendto(sockfd,buffer,,0,(struct sockaddr \*)&serv\_addr,sizeof(serv\_addr));

/\* \*\*\*\*\*\*\*above code is for connection, only do pretty stuff below of this\*\*\*\*\*\*\* \*/

int data[10];

printf("Plese input 4 bits of data\n");

scanf("%d",&data[0]);

scanf("%d",&data[1]);

scanf("%d",&data[2]);

scanf("%d",&data[4]);

// calculation fro even parity encoding

data[6]= data[4]^data[2]^data[0];

data[5]= data[4]^data[1]^data[0];

data[3]= data[2]^data[1]^data[0];

//sending the encoded data to clients

for(int i=0; i<7; i++){

n = sendto(sockfd,&data[i],sizeof(data[i]),0,(struct sockaddr \*)&serv\_addr,len);

if(n<0)printf("Error while sending");

}

printf("\nThe data sent is: ");

for(int i=0; i<7; i++){

printf("%d",data[i]);

}

printf("\n");

//Done

close(sockfd);

return 0;

}

