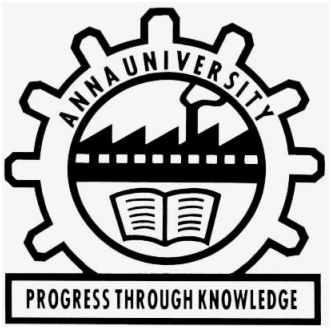
DEPARTMENT OF COMPUTER TECHNOLOGY

ANNA UNIVERSITY, MIT CAMPUS

CHROMEPET, CHENNAI – 600 044



BONAFIDE CERTIFICATE

Certified that the bonafide record of the practical work done by

Mr/Ms. _____SURIYAA V_____ of Register Number _____2020503550_____

for the _____ laboratory during the period

August 2022 to December 2022.

Date: **Lab In-charge**

INDEX

EX NO	DATE	EXPERIMENT NAME	PAGE NO	T. SIGN
1	22.08.2022	BASIC NETWORK COMMANDS		
2	29.08.2022	IMPLEMENTATION OF CHAT APPLICATION USING TCP		
3	12.09.2022	UDP ECHO SERVER AND FABRICATION OF TABLES		
4	24.09.2022	WIRESHARK LAB		
5	10.10.2022	IMPLEMENTATION OF HTTP SERVER USING SOCKETS		
6	07.11.2022	DOMAIN NAME SERVER		
7	14.11.2022	IMPLEMENTATION OF SMTP SERVER		
8	14.11.2022	WEB CACHING		
9	21.11.2022	TCP FLOW CONTROL		
10	28.11.2022	ROUTING ALGORITHMS		
11	03.12.2022	IMPLEMENTATION OF ERROR DETECTION AND CORRECTION (CRC AND CHECKSUM)		
12	05.12.2022	NS3 SIMULATION AND NETWORK CONSTRUCTION USING IT		

EXP NO: 1 DATE:	BASIC NETWORK COMMANDS

AIM:

To study about Basic Networking Commands and implementing it.

NETWORK COMMANDS:

Command: ifconfig

Used in Unix operating System

Used to configure network interface

a) ifconfig - Displays the status of the currently active interfaces.
b) Ifconfig -a - Displays the status of all interfaces, even those that are down.
c) ifconfig -s - Display a short list, instead of details.

```
dcl@dcl-Veriton-M4660G:~$ ifconfig -a
enp2s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.116.74  netmask 255.255.255.0  broadcast 192.168.116.255
        inet6 fe80::ff9b:fc57:589f:ca01  prefixlen 64  scopeid 0x20<link>
        ether 94:c6:91:cc:7b:67  txqueuelen 1000  (Ethernet)
        RX packets 197652  bytes 259286631 (259.2 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 114830  bytes 14421988 (14.4 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 4321  bytes 516155 (516.1 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 4321  bytes 516155 (516.1 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

dcl@dcl-Veriton-M4660G:~$ ifconfig
enp2s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.116.74  netmask 255.255.255.0  broadcast 192.168.116.255
        inet6 fe80::ff9b:fc57:589f:ca01  prefixlen 64  scopeid 0x20<link>
        ether 94:c6:91:cc:7b:67  txqueuelen 1000  (Ethernet)
        RX packets 197685  bytes 259289699 (259.2 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 114847  bytes 14424987 (14.4 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 4325  bytes 516753 (516.7 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 4325  bytes 516753 (516.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

```
dcl@dcl-Veriton-M4660G:~$ ifconfig -s
Iface      MTU    RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
enp2s0    1500   197694      0      0 0        114853      0      0      0 BMRU
lo       65536     4325      0      0 0          4325      0      0      0 LRU
```

Command: nslookup

Displays information about Domain Name Server.

Used to configure network interface

a) nslookup gmail.com - Displays the ip address of the domain
b) nslookup -type =soa gmail.com - Displays the authoritative information about the domain, the e-mail address of the domain admin, the domain serial number, etc…
c) nslookup -type =port gmail.com - Specify the port number of domain
d) nslookup -type =mx gmail.com - Display the mail exchange server for the domain

```
dcl@dcl-Veriton-M4660G:~$ nslookup gmail.com
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
Name:   gmail.com
Address: 142.250.182.37
Name:   gmail.com
Address: 2404:6800:4007:805::2005
```

```
dcl@dcl-Veriton-M4660G:~$ nslookup -type=any google.com
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
Name:   google.com
Address: 142.250.76.174
Name:   google.com
Address: 2404:6800:4009:81a::200e
google.com      text = "facebook-domain-verification=22rm551cu4k0ab0bxsw536tlds4h95"
google.com      text = "docusign=1b0a6754-49b1-4db5-8540-d2c12664b289"
google.com      text = "docusign=05958488-4752-4ef2-95eb-aa7ba8a3bd0e"
google.com      text = "v=spf1 include:_spf.google.com ~all"
google.com      nameserver = ns1.google.com.
google.com      nameserver = ns2.google.com.
google.com      text = "MS=E4A68B9AB2BB9670BCE15412F62916164C0B20BB"
google.com
        origin = ns1.google.com
        mail addr = dns-admin.google.com
        serial = 468663511
        refresh = 900
        retry = 900
        expire = 1800
        minimum = 60
google.com      text = "webexdomainverification.8YX6G=6e6922db-e3e6-4a36-904e-a805c28087fa"
google.com      text = "globalsign-smime-dv=CDYX+XFHUw2wml6/Gb8+59BsH31KzUr6c1l2BPvqKX8="
google.com      text = "google-site-verification=wD8N7i1JTNTkezJ49swvWW48f8_9xveREV4oB-0Hf5o"
google.com      text = "atlassian-domain-verification=5YjTmWmjI92ewqkx2oXmBaD60Td9zWon9r6eakvHX6B77z
google.com      text = "google-site-verification=TV9-DBe4R80X4v0M4U_bd_J9cpOJM0nikft0jAgjmsQ"
google.com      nameserver = ns3.google.com.
google.com      rdata_65 = \# 13 0001000001000060268320268 33
google.com      mail exchanger = 10 smtp.google.com.
google.com      nameserver = ns4.google.com.
google.com      text = "apple-domain-verification=30afIBcvSuDV2PLX"
google.com      rdata_257 = 0 issue "pki.goog"

Authoritative answers can be found from:
```

```
dcl@dcl-Veriton-M4660G:~$ nslookup -type=soa google.com
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
google.com
        origin = ns1.google.com
        mail addr = dns-admin.google.com
        serial = 468663511
        refresh = 900
        retry = 900
        expire = 1800
        minimum = 60

Authoritative answers can be found from:
```

```
dcl@dcl-Veriton-M4660G:~$ nslookup -type=mx gmail.com
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
gmail.com       mail exchanger = 20 alt2.gmail-smtp-in.l.google.com.
gmail.com       mail exchanger = 5 gmail-smtp-in.l.google.com.
gmail.com       mail exchanger = 10 alt1.gmail-smtp-in.l.google.com.
gmail.com       mail exchanger = 30 alt3.gmail-smtp-in.l.google.com.
gmail.com       mail exchanger = 40 alt4.gmail-smtp-in.l.google.com.

Authoritative answers can be found from:
```

Command: traceroute -

Used on Unix-like Operating Systems.

Used to determine the route to a destination by sending ICMP packets to the destination.

a) traceroute -m <maximum_hops> - Maximum number of hops to search for target.
b) traceroute -w timeout google.com - Wait timeout milliseconds for each reply
c) traceroute -4 google.com - Force using IPv4

```
dcl@dcl-Veriton-M4660G:~$ traceroute -m 3 google.com
traceroute to google.com (142.250.182.142), 3 hops max
  1   192.168.116.1  9.669ms  9.972ms  11.164ms
  2   *  *  *
  3   *  *  *
```

```
dcl@dcl-Veriton-M4660G:~$ traceroute -w 3 google.com
traceroute to google.com (142.250.182.142), 64 hops max
  1   192.168.116.1  13.920ms  11.698ms  7.777ms
  2   *  *  *
  3   *  *  *
  4   *  *  *
  5   *  *  *
  6   *  *  *
  7   *  *  *
  8   *  *  *
  9   *  *  *
```

```
student@student-Veriton-M4660G:~/3036$ traceroute -4 google.com
traceroute to google.com (142.250.195.142), 30 hops max, 60 byte packets
 1  _gateway (192.168.117.1)  7.718 ms  7.949 ms  8.117 ms
 2  14.139.190.97 (14.139.190.97)  0.718 ms  0.731 ms  0.667 ms
 3  10.119.233.157 (10.119.233.157)  0.560 ms  0.576 ms  0.588 ms
 4  * * *
 5  10.119.73.122 (10.119.73.122)  2.297 ms  2.856 ms  2.591 ms
 6  72.14.213.20 (72.14.213.20)  2.861 ms  2.277 ms  2.483 ms
 7  * * *
 8  216.239.56.62 (216.239.56.62)  2.094 ms 74.125.242.129 (74.125.242.129)  3.087 ms 142.250.2
 9  142.251.55.61 (142.251.55.61)  1.489 ms 74.125.242.131 (74.125.242.131)  2.525 ms 142.251.5
10  108.170.253.113 (108.170.253.113)  2.480 ms * 108.170.253.97 (108.170.253.97)  3.555 ms
11  142.251.55.63 (142.251.55.63)  2.725 ms  2.508 ms 142.251.55.61 (142.251.55.61)  2.079 ms
12  * * *
13  * * *
14  * * *
15  * * *
16  * * *
17  * * *
18  * * *
19  * * *
20  * * *
21  * * *
22  * * *
23  * * *
24  * * *
25  * * *
26  * * *
27  * * *
28  * * *
29  * * *
30  * * *
```

Command: ping

Used to verify IP-level connectivity to another TCP/IP computer by sending Internet Control Message Protocol (ICMP).

a) ping –c count gmail.com - Number of echo requests to send.
b) ping –n TTL gmail.com - Time to Live
c) ping –w timeout gmail.com -Timeout in milliseconds to wait for each reply.
d) ping –l size gmail.com - Send buffer size

```
dcl@dcl-Veriton-M4660G:~$ ping -c 5 gmail.com
PING gmail.com (142.250.193.133) 56(84) bytes of data.

--- gmail.com ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4079ms
```

```
student@student-Veriton-M4660G:~/3036$ ping -n 58 google.com
PING google.com (142.250.182.142) 56(124) bytes of data.
^C
--- google.com ping statistics ---
7 packets transmitted, 0 received, 100% packet loss, time 6133ms
```

```
student@student-Veriton-M4660G:~/3036$ ping -w 2 google.com
PING google.com (142.250.182.14) 56(84) bytes of data.
64 bytes from maa05s18-in-f14.1e100.net (142.250.182.14): icmp_seq=1 ttl=58 time=2.67 ms
64 bytes from maa05s18-in-f14.1e100.net (142.250.182.14): icmp_seq=2 ttl=58 time=2.49 ms

--- google.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 2.494/2.583/2.672/0.089 ms
```

```
student@student-Veriton-M4660G:~/3036$ ping -l 2 google.com
PING google.com (142.250.182.142) 56(84) bytes of data.
64 bytes from maa05s22-in-f14.1e100.net (142.250.182.142): icmp_seq=1 ttl=58 time=2.04 ms
64 bytes from maa05s22-in-f14.1e100.net (142.250.182.142): icmp_seq=2 ttl=58 time=2.06 ms
64 bytes from maa05s22-in-f14.1e100.net (142.250.182.142): icmp_seq=3 ttl=58 time=2.04 ms
64 bytes from maa05s22-in-f14.1e100.net (142.250.182.142): icmp_seq=4 ttl=58 time=1.97 ms
64 bytes from maa05s22-in-f14.1e100.net (142.250.182.142): icmp_seq=5 ttl=58 time=2.09 ms
64 bytes from maa05s22-in-f14.1e100.net (142.250.182.142): icmp_seq=6 ttl=58 time=2.06 ms
64 bytes from maa05s22-in-f14.1e100.net (142.250.182.142): icmp_seq=7 ttl=58 time=2.05 ms
64 bytes from maa05s22-in-f14.1e100.net (142.250.182.142): icmp_seq=8 ttl=58 time=1.97 ms
64 bytes from maa05s22-in-f14.1e100.net (142.250.182.142): icmp_seq=9 ttl=58 time=2.07 ms
64 bytes from maa05s22-in-f14.1e100.net (142.250.182.142): icmp_seq=10 ttl=58 time=2.05 ms
64 bytes from maa05s22-in-f14.1e100.net (142.250.182.142): icmp_seq=11 ttl=58 time=2.15 ms
64 bytes from maa05s22-in-f14.1e100.net (142.250.182.142): icmp_seq=12 ttl=58 time=2.03 ms
64 bytes from maa05s22-in-f14.1e100.net (142.250.182.142): icmp_seq=13 ttl=58 time=2.12 ms
64 bytes from maa05s22-in-f14.1e100.net (142.250.182.142): icmp_seq=14 ttl=58 time=1.89 ms
64 bytes from maa05s22-in-f14.1e100.net (142.250.182.142): icmp_seq=15 ttl=58 time=2.09 ms
^C
--- google.com ping statistics ---
15 packets transmitted, 15 received, 0% packet loss, time 13003ms
rtt min/avg/max/mdev = 1.894/2.045/2.149/0.061 ms, pipe 2
```

```
dcl@dcl-Veriton-M4660G:~$ ping -c 5 -l 2 gmail.com
PING gmail.com (142.250.67.69) 56(84) bytes of data.

--- gmail.com ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 3001ms
```

Command: hostname

Used to obtain the DNS(Domain Name System) name and set the system's hostname or NIS(Network Information System) domain name.

A hostname is a name which is given to a computer and it is attached to the network. Its main purpose is to uniquely identify over a network.

a) hostname -a - Used to get alias name of the host system(if any). It will return an empty line if no alias name is set.
b) hostname -A - Used to get all FQDNs(Fully Qualified Domain Name) of the host system.
c) hostname -b - Used to always set a hostname. Default name is used if none specified

```
student@student-Veriton-M4660G:~/3036$ hostname
student-Veriton-M4660G
student@student-Veriton-M4660G:~/3036$ hostname -b
student-Veriton-M4660G
student@student-Veriton-M4660G:~/3036$ hostname -A
student-Veriton-M4660G
```

Command: route

route command in Linux is used when you want to work with the IP/kernel routing table. It is mainly used to set up static routes to specific hosts or networks via an interface.
It is used for showing or updating the IP/kernel routing table.

a) route - Display the routing table

b) router -n - Display routing table in full numeric form

```
dcl@dcl-Veriton-M4660G:~$ route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         _gateway        0.0.0.0         UG    100    0        0 enp2s0
link-local      0.0.0.0         255.255.0.0     U     1000   0        0 enp2s0
192.168.116.0   0.0.0.0         255.255.255.0   U     100    0        0 enp2s0
dcl@dcl-Veriton-M4660G:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.116.1   0.0.0.0         UG    100    0        0 enp2s0
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 enp2s0
192.168.116.0   0.0.0.0         255.255.255.0   U     100    0        0 enp2s0
```

Command: ifdown

- Used in Linux like OS
- It places the network interface in a state where it cannot transmit or receive data.
- ifdown command to bring network interface down, not allowing the user to transmit and receive data.

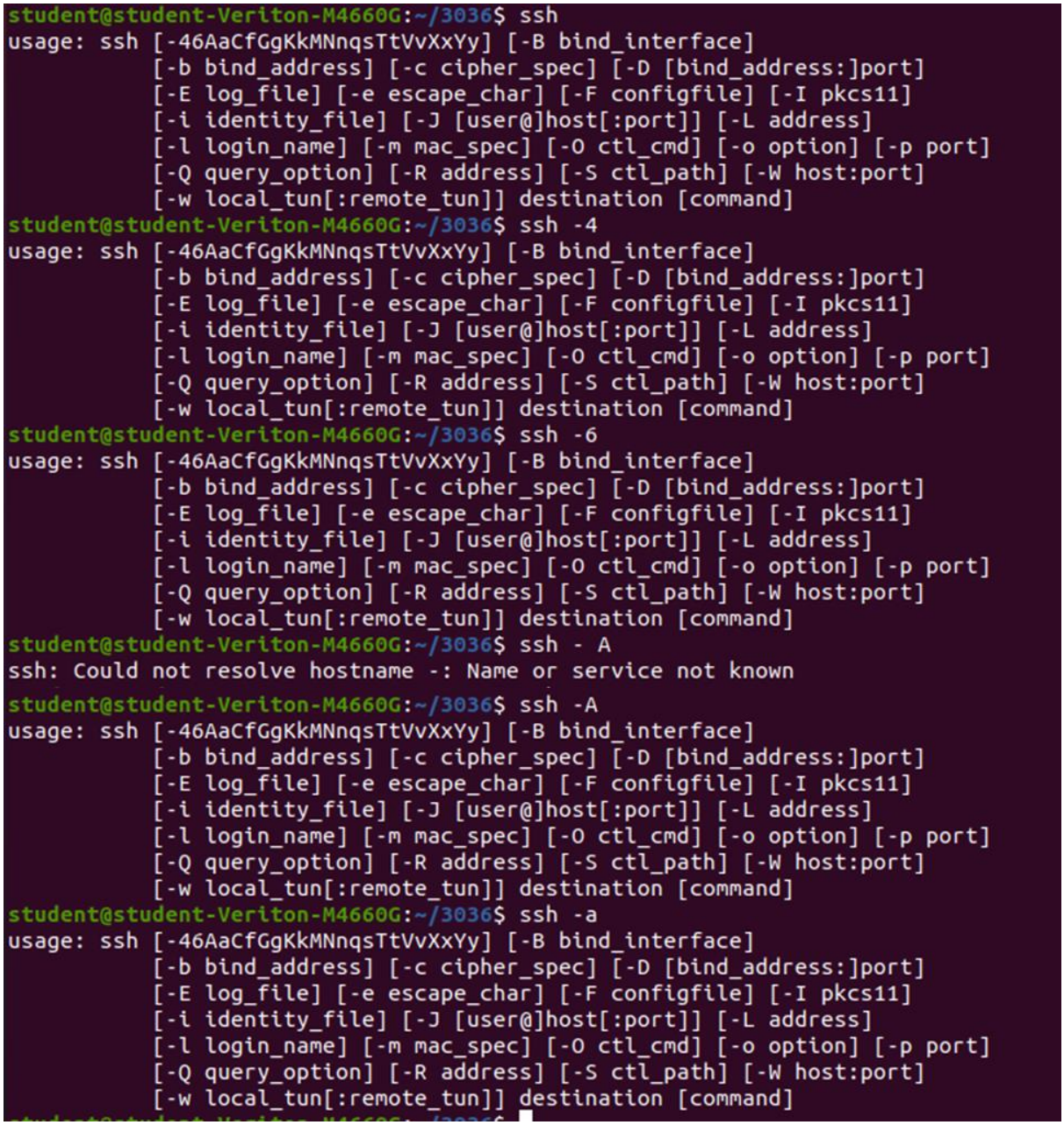
a) -a - used to bring all the interface down which are defined.
b) -allow=CLASS - allow interfaces listed in an allow-CLASS
c) -n - Do not configure any interface

```
student@student-Veriton-M4660G:~/3036$ sudo ifdown -av
ifdown: reading directory /etc/network/interfaces.d
/bin/run-parts --verbose /etc/network/if-down.d
run-parts: executing /etc/network/if-down.d/avahi-autoipd
run-parts: executing /etc/network/if-down.d/openvpn
run-parts: executing /etc/network/if-down.d/wpasupplicant
ifdown: configuring interface lo=lo (inet)
/bin/run-parts --verbose /etc/network/if-down.d
run-parts: executing /etc/network/if-down.d/avahi-autoipd
run-parts: executing /etc/network/if-down.d/openvpn
run-parts: executing /etc/network/if-down.d/wpasupplicant
/bin/run-parts --verbose /etc/network/if-post-down.d
run-parts: failed to stat component /etc/network/if-post-down.d/avahi-daemon: No such file or directory
run-parts: executing /etc/network/if-post-down.d/wireless-tools
run-parts: executing /etc/network/if-post-down.d/wpasupplicant
/sbin/ip link set down dev lo 2>/dev/null

/bin/run-parts --verbose /etc/network/if-post-down.d
run-parts: failed to stat component /etc/network/if-post-down.d/avahi-daemon: No such file or directory
run-parts: executing /etc/network/if-post-down.d/wireless-tools
run-parts: executing /etc/network/if-post-down.d/wpasupplicant
```

Command: ssh

- Used in Linux like OS
- Stands for Secure Shell.
- Used to securely connect to a remote server/system.
- secure in the sense that it transfers the data in encrypted form between the host and the client.

a) -4 - Allows IPv4 addresses only.
b) -6 - Allows IPv6 addresses only.
c) -A -Authentication agent connection forwarding is enabled.
d) -a - Authentication agent connection forwarding is disabled.



Command: scp (Secure Copy Protocol)

- Used in Linux like OS
- Used to do secure copy allows secure transferring of files in between the local host and the remote host or between two remote hosts.
- It uses the same authentication and security as it is used in the Secure Shell (SSH) protocol.

a) scp -P port - Specifies the port to connect on the remote host.
b) scp -q - Disables the progress meter.
c) scp -r - Recursively copy entire directories

```
student@student-Veriton-M4660G:~/3036$ scp -q
usage: scp [-346BCpqrTv] [-c cipher] [-F ssh_config] [-i identity_file]
           [-J destination] [-l limit] [-o ssh_option] [-P port]
           [-S program] source ... target
student@student-Veriton-M4660G:~/3036$ scp -r
usage: scp [-346BCpqrTv] [-c cipher] [-F ssh_config] [-i identity_file]
           [-J destination] [-l limit] [-o ssh_option] [-P port]
           [-S program] source ... target
student@student-Veriton-M4660G:~/3036$ scp -P 2249
usage: scp [-346BCpqrTv] [-c cipher] [-F ssh_config] [-i identity_file]
           [-J destination] [-l limit] [-o ssh_option] [-P port]
           [-S program] source ... target
```

Command: tracepath

- Used in Linux like OS
- Used to trace path to destination discovering MTU (Maximum Transmission Unit) along this path. It uses a UDP port or some random port.
- It is similar to traceroute, but it does not require superuser privileges.

a) tracepath -n www.google.com - prints primarily IP addresses numerically.
b) tracepath -b www.google.com - print both of host names and IP addresses.
c) tracepath -l 29 www.google.com - This option sets the initial packet length to pktlen instead of 65535 for tracepath

```
student@student-Veriton-M4660G:~/3036$ tracepath -b google.com
 1?: [LOCALHOST]                      pmtu 1500
 1:  _gateway (192.168.117.1)                              19.373ms
 1:  _gateway (192.168.117.1)                              16.788ms
 2:  14.139.190.97 (14.139.190.97)                          6.810ms
 3:  10.119.233.157 (10.119.233.157)                        0.774ms
 4:  no reply
 5:  10.119.73.122 (10.119.73.122)                          4.394ms
 6:  72.14.213.20 (72.14.213.20)                            2.980ms asymm  7
 7:  no reply
 8:  no reply
 9:  no reply
^C
student@student-Veriton-M4660G:~/3036$ tracepath -l 50 google.com
 1:  _gateway                                              19.525ms
 2:  14.139.190.97                                          0.684ms
 3:  10.119.233.157                                         0.691ms
 4:  no reply
 5:  10.119.73.122                                          2.186ms
 6:  72.14.195.128                                          3.064ms asymm  7
 7:  no reply
^C
```

SCP COMMAND FOR FILE TRANSFER:

scp [options] [*username*@][*source_host*:]*file1* [*username*@][*destination_host*:]*file2*

Used to transfer files from one computer to another.

```
dcl@dcl-Veriton-M4660G:~/Desktop$ scp file.txt rusa@192.168.116.71:/home/rusa/
rusa@192.168.116.71's password:
file.txt
```

ADDITIONAL COMMANDS:

a) dig www.google.com - stands for Domain Information Groper. It is used for retrieving information about DNS name servers.

```
student@student-Veriton-M4660G:~/3036$ dig google.com

; <<>> DiG 9.16.1-Ubuntu <<>> google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 61902
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;google.com.                    IN      A

;; ANSWER SECTION:
google.com.             216     IN      A       142.250.182.14

;; Query time: 4 msec
;; SERVER: 127.0.0.53#53(127.0.0.53)
;; WHEN: Mon Aug 22 10:58:08 IST 2022
;; MSG SIZE  rcvd: 55
```

b) netstat - used for troubleshooting and configuration, that can also serve as a monitoring tool for connections over the network.

```
student@student-Veriton-M4660G:~/3036$ netstat
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 student-Veriton-M:59410 49.44.207.239:https     ESTABLISHED
tcp        0      0 student-Veriton-M:46838 maa03s26-in-f1.1e:https ESTABLISHED
tcp        0      0 student-Veriton-M:51030 104.18.18.126:https     ESTABLISHED
tcp        0      0 student-Veriton-M:35688 172.67.193.59:https     ESTABLISHED
tcp        0      0 student-Veriton-M:55826 maa05s17-in-f14.1:https TIME_WAIT
tcp        0      0 student-Veriton-M:38428 maa03s31-in-f14.1:https ESTABLISHED
tcp        0      0 student-Veriton-M:51824 180.149.61.152:http     ESTABLISHED
tcp        0      0 student-Veriton-M:40504 146.60.190.35.bc.:https ESTABLISHED
tcp        0      0 student-Veriton-M:60018 172.67.42.201:https     ESTABLISHED
tcp        0      0 student-Veriton-M:37578 maa03s45-in-f2.1e:https ESTABLISHED
tcp        0      0 student-Veriton-M:45894 117.18.237.29:http      ESTABLISHED
tcp        0      0 student-Veriton-M:35394 69.173.158.64:https     ESTABLISHED
tcp        0      0 student-Veriton-M:35664 87.70.96.34.bc.go:https ESTABLISHED
tcp        0      0 student-Veriton-M:58594 maa05s25-in-f2.1e:https ESTABLISHED
tcp        0      0 student-Veriton-M:60326 server-18-67-161-:https TIME_WAIT
tcp        0      0 student-Veriton-M:45242 104.18.13.76:https      ESTABLISHED
tcp        0      0 student-Veriton-M:33456 104.18.32.68:http       ESTABLISHED
tcp        0      0 student-Veriton-M:54318 ec2-18-140-209-49:https ESTABLISHED
tcp        0      0 student-Veriton-M:36540 maa05s26-in-f1.1e:https ESTABLISHED
tcp        0      0 student-Veriton-M:41936 maa03s36-in-f2.1e:https ESTABLISHED
tcp        0      0 student-Veriton-M:34268 139.148.107.34.bc:https TIME_WAIT
tcp        0      0 student-Veriton-M:60980 xx-fbcdn-shv-01-b:https ESTABLISHED
tcp        0      0 student-Veriton-M:57400 104.21.91.69:https      ESTABLISHED
```

c) netstat -r - prints network connections, routing tables, interface statistics, masquerade connections, and multicast memberships.

```
student@student-Veriton-M4660G:~/3036$ netstat -r
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
default         _gateway        0.0.0.0         UG        0 0          0 enp1s0
link-local      0.0.0.0         255.255.0.0     U         0 0          0 enp1s0
192.168.117.0   0.0.0.0         255.255.255.0   U         0 0          0 enp1s0
```

d) Sudo ifup -av - brings the network interface up, allowing it to transmit and receive data.

```
student@student-Veriton-M4660G:~/3036$ sudo ifup -av
ifup: reading directory /etc/network/interfaces.d
/bin/run-parts --exit-on-error --verbose /etc/network/if-pre-up.d
run-parts: executing /etc/network/if-pre-up.d/wireless-tools
run-parts: executing /etc/network/if-pre-up.d/wpasupplicant
/bin/run-parts --exit-on-error --verbose /etc/network/if-up.d
run-parts: executing /etc/network/if-up.d/avahi-autoipd
run-parts: executing /etc/network/if-up.d/openvpn
run-parts: executing /etc/network/if-up.d/wpasupplicant
student@student-Veriton-M4660G:~/3036$ sudo ifup -a
student@student-Veriton-M4660G:~/3036$ ifquery -l
lo
student@student-Veriton-M4660G:~/3036$ sudo ifup --force lo
```

e) whois www.google.com - gives information about the domain name

```
student@student-Veriton-M4660G:~/3036$ whois google.com
   Domain Name: GOOGLE.COM
   Registry Domain ID: 2138514_DOMAIN_COM-VRSN
   Registrar WHOIS Server: whois.markmonitor.com
   Registrar URL: http://www.markmonitor.com
   Updated Date: 2019-09-09T15:39:04Z
   Creation Date: 1997-09-15T04:00:00Z
   Registry Expiry Date: 2028-09-14T04:00:00Z
   Registrar: MarkMonitor Inc.
   Registrar IANA ID: 292
   Registrar Abuse Contact Email: abusecomplaints@markmonitor.com
   Registrar Abuse Contact Phone: +1.2086851750
   Domain Status: clientDeleteProhibited https://icann.org/epp#clientDeleteProhibited
   Domain Status: clientTransferProhibited https://icann.org/epp#clientTransferProhibited
   Domain Status: clientUpdateProhibited https://icann.org/epp#clientUpdateProhibited
   Domain Status: serverDeleteProhibited https://icann.org/epp#serverDeleteProhibited
   Domain Status: serverTransferProhibited https://icann.org/epp#serverTransferProhibited
   Domain Status: serverUpdateProhibited https://icann.org/epp#serverUpdateProhibited
   Name Server: NS1.GOOGLE.COM
   Name Server: NS2.GOOGLE.COM
   Name Server: NS3.GOOGLE.COM
   Name Server: NS4.GOOGLE.COM
   DNSSEC: unsigned
   URL of the ICANN Whois Inaccuracy Complaint Form: https://www.icann.org/wicf/
>>> Last update of whois database: 2022-08-22T05:29:57Z <<<
```

RESULT:

Thus, the various networking commands have been executed and implemented successfully.

EXP NO: 2 DATE:	IMPLEMENTATION OF CHAT APPLICATION USING TCP

AIM:

To implement a chat application using TCP.

ALGORITHM:

1) Create a server program and server using server socket.
2) Run Server Program.
3) Set IP Address of Server and Port.
4) Create a Client Program.
5) Set IP Address in client Program which is same as the IP Address of client Program.

CODE:

Server Program:

```
import java.net.*;
import java.io.*;
public class  chatserver {
    public static void main(String args[]) throws Exception {
        ServerSocket ss = new ServerSocket(2000);
        Socket sk = ss.accept();
      BufferedReader cin = new BufferedReader(new InputStreamReader(sk.getInputStream()));
        PrintStream cout = new PrintStream(sk.getOutputStream());
        BufferedReader stdin = new BufferedReader(new InputStreamReader(System.in));
        String s;
        while (true) {
            s = cin.readLine();
            if (s.equalsIgnoreCase("END")) {
                cout.println("BYE");
                break;
            }
            System.out.print("Client : " + s + "\n");
            System.out.print("Server : ");
            s = stdin.readLine();
            cout.println(s);
        }
        ss.close();
        sk.close();
        cin.close();
        cout.close();
        stdin.close();
    }
}
```

Client Program:

```
import java.net.*;
import java.io.*;
public class chatclient
{
    public static void main(String args[]) throws Exception
    {
        Socket sk=new Socket("127.0.0.1",8000);
        BufferedReader sin=new BufferedReader(new
        InputStreamReader(sk.getInputStream()));
        PrintStream sout=new PrintStream(sk.getOutputStream());
        BufferedReader stdin=new BufferedReader(new InputStreamReader(System.in));
        String s;
        while ( true )
        {
            System.out.print("Client : ");
            s=stdin.readLine();
            sout.println(s);
            s=sin.readLine();
            System.out.print("Server : "+s+"\n");
            if (s.equalsIgnoreCase("BYE") )
                break;
        }
        sk.close();
        sin.close();
        sout.close();
        stdin.close();
    }
}
```

OUTPUT:

SAME SYSTEMS: Client and server in same system:

```
dcl@dcl-Veriton-M4660G:~/Downloads$ java chatclient.java
Client : hii
Server : hii
Client : hii hru
Server : good
Client : ok tc bye
Server : bye
```

```
dcl@dcl-Veriton-M4660G:~/Downloads$ java chatserver.java
Client : hii
Server : hii
Client : hii hru
Server : good
Client : ok tc bye
Server : bye
```

DIFFERENT SYSTEMS: Client and server in different system:

```
dcl@dcl-Veriton-M4660G:~/Downloads$ java chatserver.java
Client : hi
Server : hii
Client : fine
Server : cool nice meeting u
Client : oky same here
Server : bye
```

Neighboring system as server (IP : 192.168.117.145)

```
Client : hi
Server : hii
Client : fine
Server : cool nice meeting u
Client : oky same here
Server : bye
```

RESULT:

Thus, the required chat application using TCP protocol was created.

EXP NO: 3 DATE:	UDP ECHO SERVER AND FABRICATION OF UTP CABLES

AIM:

To implement a UDP Echo Server in Java and to study different types of UTP cables and their Fabrication.

ALGORITHM:

UDP Server:

1. Create a UDP socket.
2. Bind the socket to the server address.
3. Wait until the datagram packet arrives from the client.
4. Process the datagram packet and send a reply to the client.
5. Go back to Step 3.

UDP Client:

1. Create a UDP socket.
2. Send a message to the server.
3. Wait until response from the server is received.
4. Process reply and go back to step 2, if necessary.
5. Close socket descriptor and exit.

CODE:

Client:

```
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.util.Scanner;
public class client
{
    public static void main(String args[]) throws IOException
    {
        Scanner sc = new Scanner(System.in);
        DatagramSocket ds = new DatagramSocket();
        // InetAddress ip = InetAddress.getByName("192.168.117.138");
        InetAddress ip = InetAddress.getLocalHost();
        byte buf[] = null;
        while (true)
        {
            String inp = sc.nextLine();
            buf = inp.getBytes();
            DatagramPacket DpSend = new DatagramPacket(buf, buf.length, ip, 1234);
```

```
            ds.send(DpSend);
            if (inp.equals("exit"))
```

Server

```
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.SocketException;
public class server
{
    public static void main(String[] args) throws IOException
    {
        DatagramSocket ds = new DatagramSocket(1234);
        byte[] receive = new byte[65535];
        DatagramPacket DpReceive = null;
        while (true)
        {
            DpReceive = new DatagramPacket(receive, receive.length);
            ds.receive(DpReceive);
            System.out.println("Client:-" + data(receive));
            if (data(receive).toString().equals("exit"))
            {
                System.out.println("Client sent bye.....EXITING");
                break;
            }
            receive = new byte[65535];
        }
    }
    public static StringBuilder data(byte[] a)
    {
        if (a == null)
            return null;
        StringBuilder ret = new StringBuilder();
        int i = 0;
        while (a[i] != 0)
        {
            ret.append((char) a[i]);
            i++;
        }
        return ret;
    }
}
```

OUTPUT:

1. LOCALHOST

Client

```
hello
this is Echo server-client implementation
```

Server

```
Client:-hello
Client:-this is Echo server-client implementation
```

2. BETWEEN TWO SYSTEM

Client:

```
PS C:\Users\HP\Downloads> java Client
hello
THis is client
nice to meet you
Exit
```

Server:

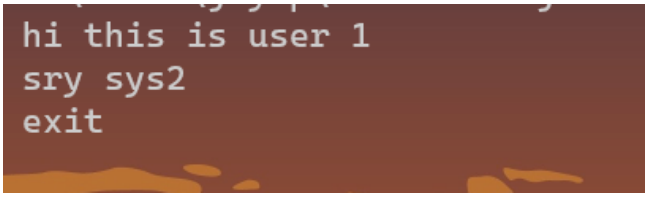
```
/192.168.112.6: hello
/192.168.112.6: THis is client
/192.168.112.6: nice to meet you
/192.168.112.6: Exit
Client is leaving.....EXITING
```

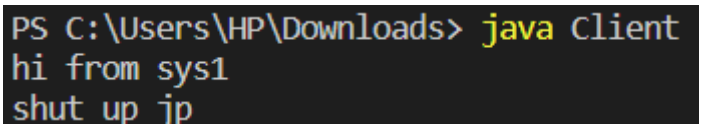
3. MULTIPLE SYSTEM

Server

```
/192.168.112.6: hi from sys1
/192.168.112.167: hi this is user 1
/192.168.112.167: sry sys2
/192.168.112.6: shut up jp
/192.168.112.6: Exit
Client is leaving.....EXITING
```

Client





Fabrication and Simulation of UTP Cables:

THEORY:

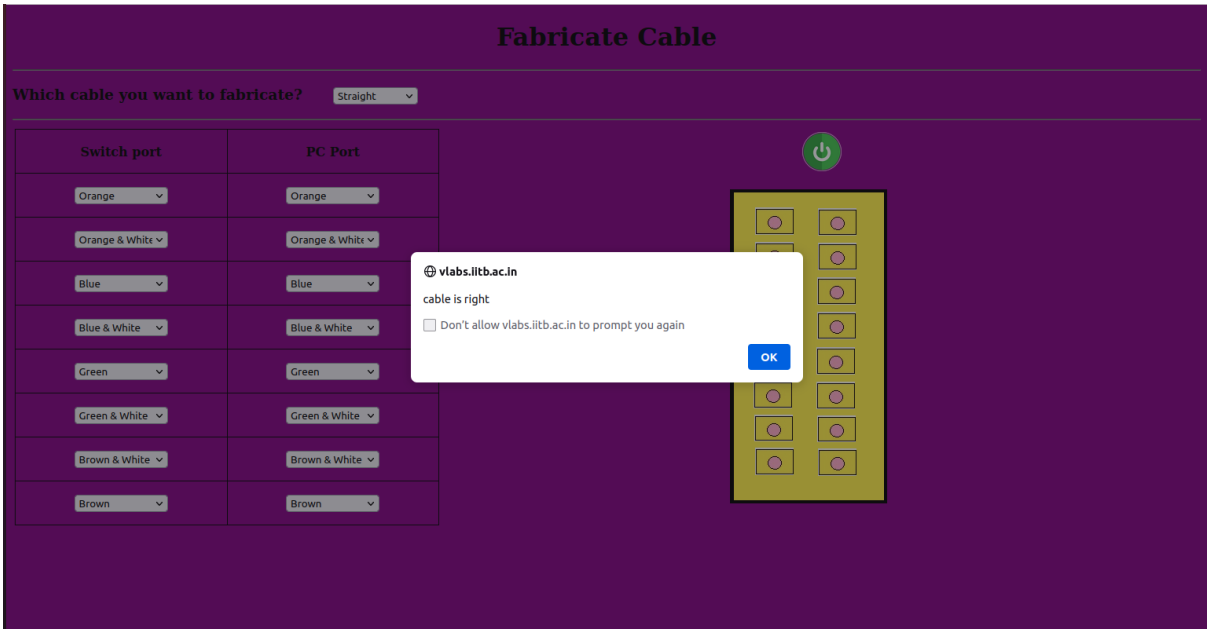
A twisted pair consists of two insulated conductors twisted together in the shape of a spiral. It can be shielded or unshielded. The unshielded twisted pair cables are very cheap and easy to install. But they are very badly affected by the electromagnetic noise interference.
Twisting of wires will reduce the effect of noise or external interference. The induced emf into the two wires due to interference tends to cancel each other due to twisting. Number of twists per unit length will determine the quality of cable. More twists mean better quality.

There are 3 types of UTP cables: -

1) Straight-through cable
2) Crossover cable
3) Roll-over cable

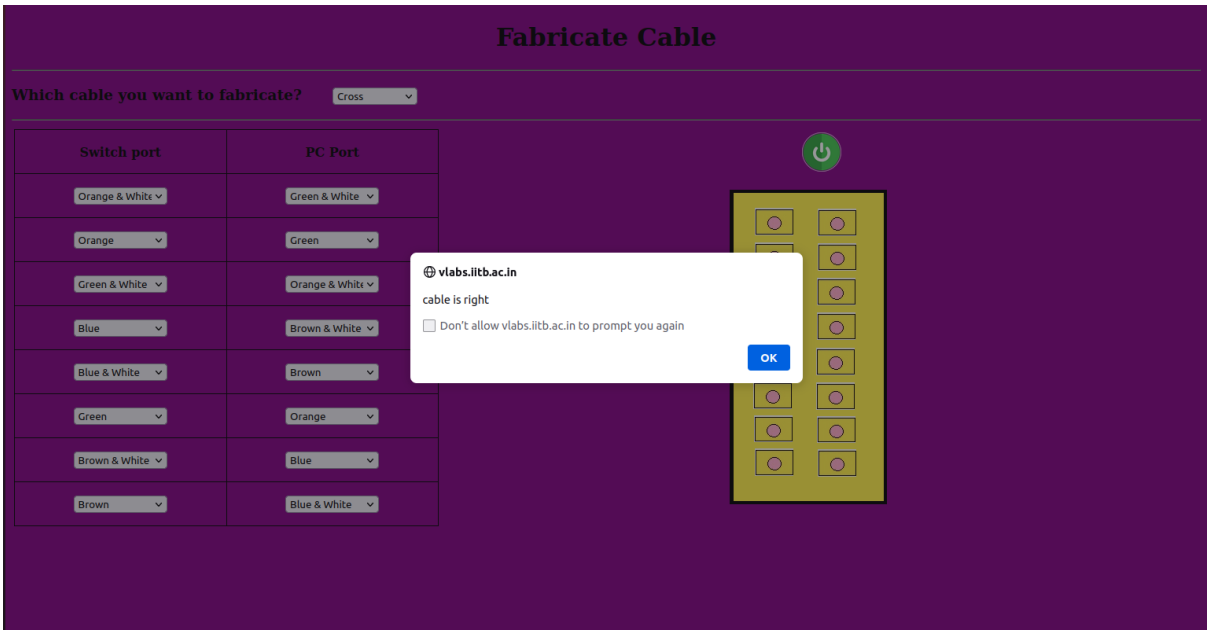
A. Straight-through cable

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



B. Crossover cable

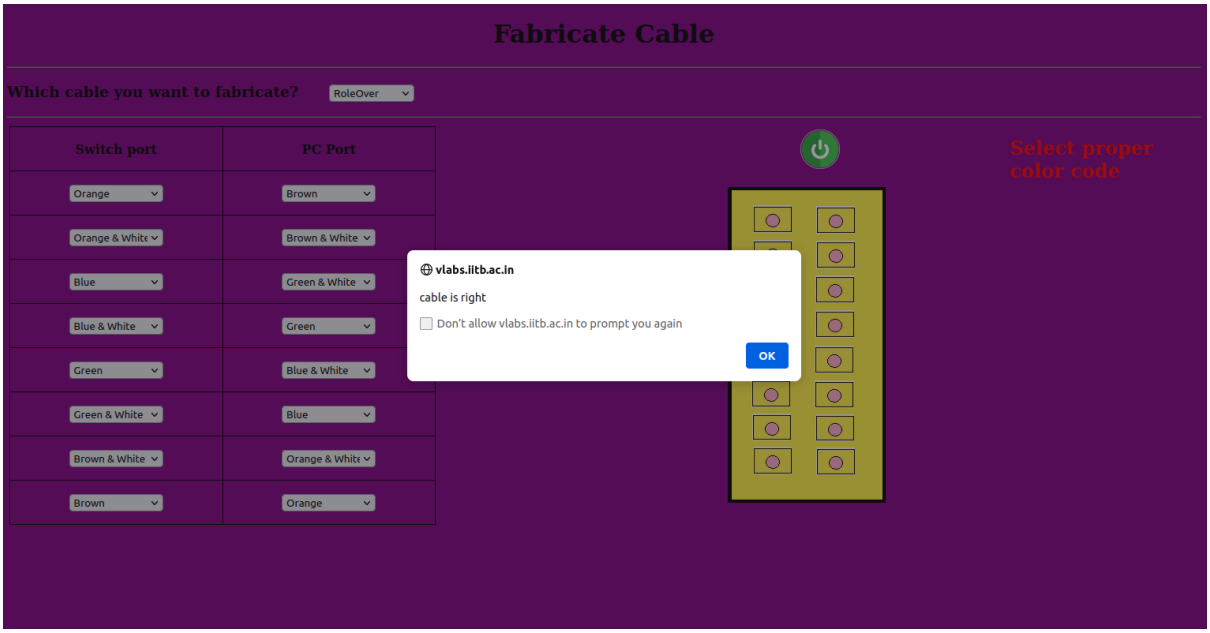
Crossover wired cables (commonly called crossover cables) are very much like Straight-Through cables with the exception that TX and RX lines are crossed (they are at opposite positions on either end of the cable. Using the 568-B standard as an example below you will see that Pin 1 on connector A goes to Pin 3 on connector B. Pin 2 on connector A goes to Pin 6 on connector B etc. Crossover cables are most commonly used to connect two hosts directly. Examples would be connecting a computer directly to another computer, connecting a switch directly to another switch, or connecting a router to a router. Note: While in the past when connecting two host devices directly a crossover cable was required. Now days most devices have auto sensing technology that detects the cable and device and crosses pairs when needed.



C. Roll-over cable

Rollover wired cables most commonly called rollover cables, have opposite Pin assignments on each end of the cable or in other words it is "rolled over". Pin 1 of connector A would be connected to Pin 8 of connector B. Pin 2 of connector A would be connected to Pin 7 of connector B and so on. Rollover

cables, sometimes referred to as Yost cables are most commonly used to connect to a devices console port to make programming changes to the device. Unlike crossover and straight-wired cables, rollover cables are not intended to carry data but instead create an interface with the device.



RESULT:

Thus, an UDP Echo Server was created and fabrication of UTP cables have been done and learnt successfully.

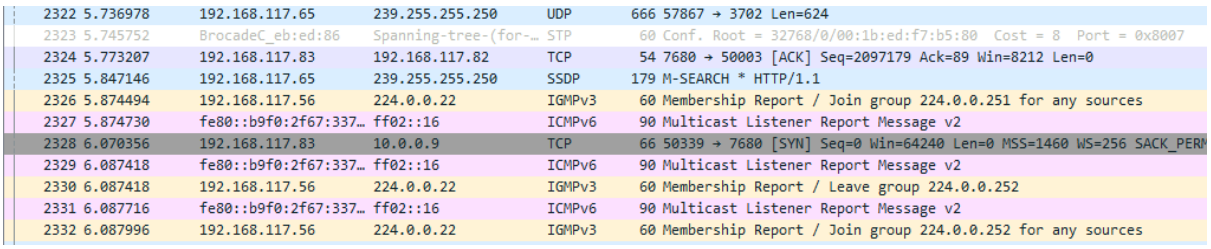
EXP NO: 4 DATE:	WIRESHARK LAB

AIM:

To capture packet transfer data in wireshark and study the wireshark application.

EXERCISE:

1. List 3 different protocols that appear in the protocol column in the unfiltered packet-listing window. Support your answer with an appropriate screenshot from your computer.

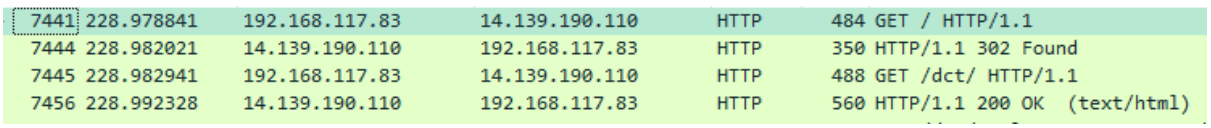


2322	5.736978	192.168.117.65	239.255.255.250	UDP	666 57867 → 3702 Len=624
2323	5.745752	BrocadeC_eb:ed:86	Spanning-tree-(for-…	STP	60 Conf. Root = 32768/0/00:1b:ed:f7:b5:80 Cost = 8 Port = 0x8007
2324	5.773207	192.168.117.83	192.168.117.82	TCP	54 7680 → 50003 [ACK] Seq=2097179 Ack=89 Win=8212 Len=0
2325	5.847146	192.168.117.65	239.255.255.250	SSDP	179 M-SEARCH * HTTP/1.1
2326	5.874494	192.168.117.56	224.0.0.22	IGMPv3	60 Membership Report / Join group 224.0.0.251 for any sources
2327	5.874730	fe80::b9f0:2f67:337…	ff02::16	ICMPv6	90 Multicast Listener Report Message v2
2328	6.070356	192.168.117.83	10.0.0.9	TCP	66 50339 → 7680 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
2329	6.087418	fe80::b9f0:2f67:337…	ff02::16	ICMPv6	90 Multicast Listener Report Message v2
2330	6.087418	192.168.117.56	224.0.0.22	IGMPv3	60 Membership Report / Leave group 224.0.0.252
2331	6.087716	fe80::b9f0:2f67:337…	ff02::16	ICMPv6	90 Multicast Listener Report Message v2
2332	6.087996	192.168.117.56	224.0.0.22	IGMPv3	60 Membership Report / Join group 224.0.0.252 for any sources

Different protocols available are **UDP, TCP, SSDP, ICMPv3, ICMPv6**

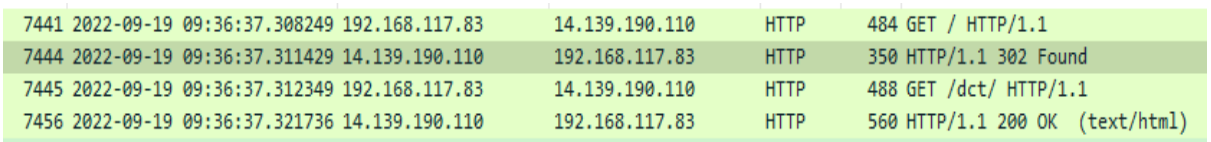
2. How long did it take from when the HTTP GET message was sent until the HTTP OK reply was received? (By default, the value of the Time column in the packet-listing window is the amount of time, in seconds, since Wireshark tracing began. To display the Time field in time-of-day format, select the Wireshark View pull down menu, then select Time Display Format, then select Time-of-day.)

Original view:



7441	228.978841	192.168.117.83	14.139.190.110	HTTP	484 GET / HTTP/1.1
7444	228.982021	14.139.190.110	192.168.117.83	HTTP	350 HTTP/1.1 302 Found
7445	228.982941	192.168.117.83	14.139.190.110	HTTP	488 GET /dct/ HTTP/1.1
7456	228.992328	14.139.190.110	192.168.117.83	HTTP	560 HTTP/1.1 200 OK (text/html)

View after changing of time:



7441	2022-09-19 09:36:37.308249	192.168.117.83	14.139.190.110	HTTP	484 GET / HTTP/1.1
7444	2022-09-19 09:36:37.311429	14.139.190.110	192.168.117.83	HTTP	350 HTTP/1.1 302 Found
7445	2022-09-19 09:36:37.312349	192.168.117.83	14.139.190.110	HTTP	488 GET /dct/ HTTP/1.1
7456	2022-09-19 09:36:37.321736	14.139.190.110	192.168.117.83	HTTP	560 HTTP/1.1 200 OK (text/html)

HTTP GET Request:

```
Frame 7441: 484 bytes on wire (3872 bits), 484 bytes captured (387
  Interface id: 0 (\Device\NPF_{41B2F740-6177-4753-A091-2C9C35AF2
      Interface name: \Device\NPF_{41B2F740-6177-4753-A091-2C9C35A
      Interface description: Ethernet
    Encapsulation type: Ethernet (1)
    Arrival Time: Sep 19, 2022 09:36:37.308249000 India Standard Ti
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1663560397.308249000 seconds
    [Time delta from previous captured frame: 0.000293000 seconds]
    [Time delta from previous displayed frame: 0.000000000 seconds]
    [Time since reference or first frame: 228.978841000 seconds]
    Frame Number: 7441
```

HTTP OK Reply:

```
Frame 7444: 350 bytes on wire (2800 bits), 350 bytes captured (2800
  Interface id: 0 (\Device\NPF_{41B2F740-6177-4753-A091-2C9C35AF23
      Interface name: \Device\NPF_{41B2F740-6177-4753-A091-2C9C35AF
      Interface description: Ethernet
    Encapsulation type: Ethernet (1)
    Arrival Time: Sep 19, 2022 09:36:37.311429000 India Standard Tim
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1663560397.311429000 seconds
    [Time delta from previous captured frame: 0.001746000 seconds]
    [Time delta from previous displayed frame: 0.003180000 seconds]
    [Time since reference or first frame: 228.982021000 seconds]
    Frame Number: 7444
```

The difference of these 2 times gives .982021 - .978841= **0.03180 seconds**

3. What is the Internet address of the ct.mitindia.edu? What is the Internet address of your computer? Support your answer with an appropriate screenshot from your computer.

If we look at the IP section of the GET request, the source and destination are shown. The source is the local machine's address and the destination is the web server's public address.

Source	Destination	Protocol
192.168.117.83	14.139.190.110	HTTP

My (local machine's) address = 192.168.117.83

IP address 128.119.245.12 = http://www.ct.mitindia.edu/dct/#/home

4. Print the two HTTP messages (GET and OK) referred to in question 2 above. To do so, select Print from the Wireshark File command menu, and select the "Selected Packet Only" and "Print as displayed" radial buttons, and then click OK.

HTTP GET:

```
No.     Time           Source                Destination           Protocol Length Info
    165 2.816497       192.168.117.173       14.139.190.110        HTTP     405    GET /dct/12-es2015.1d!
Frame 165: 405 bytes on wire (3240 bits), 405 bytes captured (3240 bits) on interface \Device\NPF_{4C09D:
E637975929B0}, id 0
Ethernet II, Src: EliteGro_ca:31:3c (94:c6:91:ca:31:3c), Dst: PrimaryA_69:8c:00 (00:20:9c:69:8c:00)
Internet Protocol Version 4, Src: 192.168.117.173, Dst: 14.139.190.110
Transmission Control Protocol, Src Port: 54524, Dst Port: 80, Seq: 1, Ack: 1, Len: 351
Hypertext Transfer Protocol
```

HTTP OK:

```
No.     Time           Source                Destination           Protocol Length Info
    747 2.847724       14.139.190.110        192.168.117.173       HTTP     649    HTTP/1.1 200 OK  (application/java
Frame 747: 649 bytes on wire (5192 bits), 649 bytes captured (5192 bits) on interface \Device\NPF_{4C09D3FA-74A8-4687
E637975929B0}, id 0
Ethernet II, Src: PrimaryA_69:8c:00 (00:20:9c:69:8c:00), Dst: EliteGro_ca:31:3c (94:c6:91:ca:31:3c)
Internet Protocol Version 4, Src: 14.139.190.110, Dst: 192.168.117.173
Transmission Control Protocol, Src Port: 80, Dst Port: 54521, Seq: 173741, Ack: 351, Len: 595
[120 Reassembled TCP Segments (174335 bytes): #168(1460), #174(1460), #178(1460), #185(1460), #190(1460), #198(1460),
#214(1460), #220(1460), #229(1460), #242(1460), #247(1460), #254(1460), #261(1460), #269(1460), #273(1460), #2]
```

5. Write the exact packet capture filter expressions to accomplish the following:
   a) Capture all TCP traffic to/from Facebook, during the time when you log in to your Facebook account host www.facebook.com and tcp

ip.dst == 142.250.196.14 && tcp

No.	Time	Source	Destination	Protocol	Length	Info
2488	10:45:27.789892	192.168.117.162	142.250.196.14	TCP	55	56825 → 4
3767	10:46:02.081408	192.168.117.162	142.250.196.14	TCP	54	56825 → 4
3778	10:46:02.083143	192.168.117.162	142.250.196.14	TCP	54	56825 → 4

Communication between Facebook and System

   b) Capture all HTTP traffic to/from Facebook, when you log in to your Facebook accounthost www.facebook.com and (port 80 || port 443)

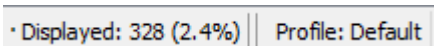
ip.dst == 142.250.196.14 && tcp.port==443

No.	Time	Source	Destination	Protocol	Length	Info
2488	10:45:27.789892	192.168.117.162	142.250.196.14	TCP	55	56825 → 443 [ACK] Seq=1
3767	10:46:02.081408	192.168.117.162	142.250.196.14	TCP	54	56825 → 443 [FIN, ACK] S
3778	10:46:02.083143	192.168.117.162	142.250.196.14	TCP	54	56825 → 443 [ACK] Seq=3

   c) Find a popular YouTube video and play it while capturing all traffic to/from YouTube ip.host == "youtube.com"

ip.dst == 172.217.160.142

No.	Time	Source	Destination	Protocol	Length	Info
4143	11:16:43.913679	192.168.117.162	172.217.160.142	UDP	1292	50725 → 443 Len=1250
4144	11:16:43.913688	192.168.117.162	172.217.160.142	UDP	781	50725 → 443 Len=739
4148	11:16:43.917973	192.168.117.162	172.217.160.142	UDP	75	50725 → 443 Len=33
4152	11:16:43.970165	192.168.117.162	172.217.160.142	UDP	81	50725 → 443 Len=39
4153	11:16:43.971344	192.168.117.162	172.217.160.142	UDP	75	50725 → 443 Len=33
6045	11:16:52.288991	192.168.117.162	172.217.160.142	UDP	690	50725 → 443 Len=648
6047	11:16:52.293157	192.168.117.162	172.217.160.142	UDP	75	50725 → 443 Len=33
6050	11:16:52.328306	192.168.117.162	172.217.160.142	UDP	81	50725 → 443 Len=39
6051	11:16:52.329681	192.168.117.162	172.217.160.142	UDP	75	50725 → 443 Len=33
8683	11:17:02.292618	192.168.117.162	172.217.160.142	UDP	954	50725 → 443 Len=912
8685	11:17:02.297365	192.168.117.162	172.217.160.142	UDP	75	50725 → 443 Len=33
8694	11:17:02.329575	192.168.117.162	172.217.160.142	UDP	81	50725 → 443 Len=39
8696	11:17:02.331349	192.168.117.162	172.217.160.142	UDP	75	50725 → 443 Len=33
10477	11:17:10.595961	192.168.117.162	172.217.160.142	UDP	1285	50725 → 443 Len=1243
10478	11:17:10.595981	192.168.117.162	172.217.160.142	UDP	1292	50725 → 443 Len=1250
10479	11:17:10.595991	192.168.117.162	172.217.160.142	UDP	547	50725 → 443 Len=505
10486	11:17:10.600433	192.168.117.162	172.217.160.142	UDP	75	50725 → 443 Len=33
10657	11:17:10.636755	192.168.117.162	172.217.160.142	UDP	81	50725 → 443 Len=39
10662	11:17:10.640013	192.168.117.162	172.217.160.142	UDP	75	50725 → 443 Len=33
13112	11:17:15.291795	192.168.117.162	172.217.160.142	UDP	696	50725 → 443 Len=654

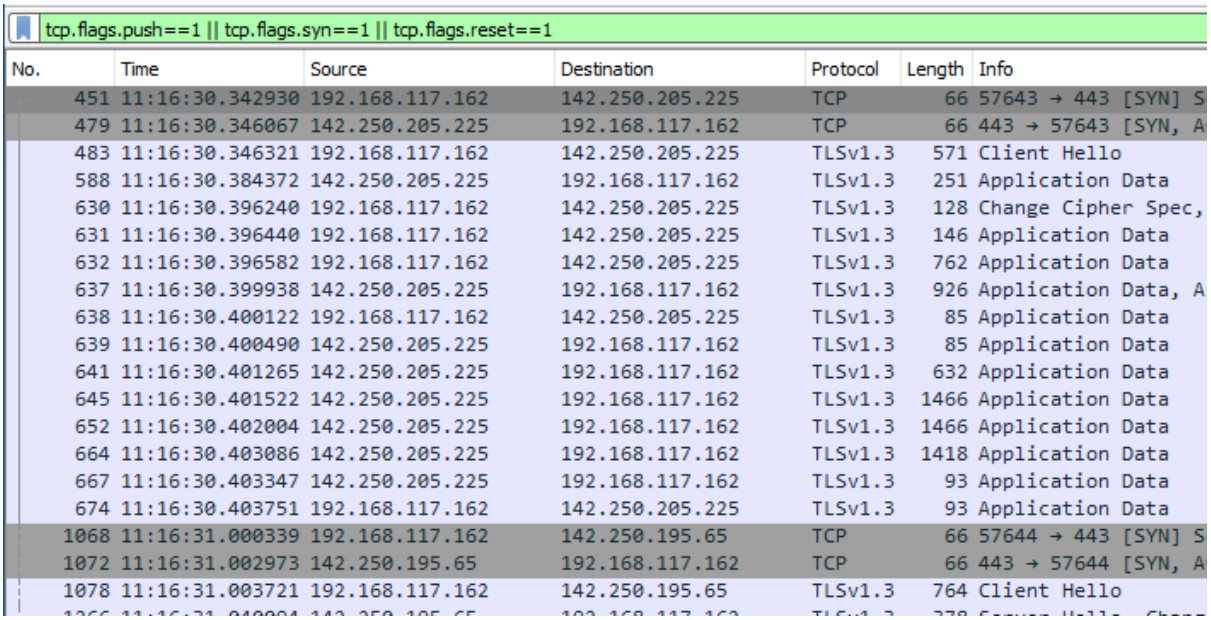


Total packets = 328
While playing a popular youtube video 328 udp packets were transferred

6. After you run Wireshark with the above capture filters and collected the data, do the following:

   I. Write a DISPLAY filter expression to count all TCP packets (captured under 5a) that have the flags SYN, PSH, and RST set. Show the fraction of packets that had each flag set.

   - TYPE "tcp.flags.push==1 || tcp.flags.syn==1 || tcp.flags.reset==1" in display filter



   - Enter "Ctrl+Alt+Shift+C" to view statistics which displays the number of packets displayed and the fraction of the packets that had each flag set.

File

Name:	C:\Users\dcl\AppData\Local\Temp\wireshark_EthernetRZ2KS1.pcapng
Length:	55 MB
Hash (SHA256):	ec0c58fc0a81016e730e046c3a5ab6d6ba10f19d38fcc2e945bcb4b49c151110
Hash (RIPEMD160):	be73b195666f66889f95cd100d8a587b49715b0e
Hash (SHA1):	a447ba577cd04def77c52755face3b45f1a63f83
Format:	Wireshark/... - pcapng
Encapsulation:	Ethernet

Time

First packet:	2022-09-24 11:16:29
Last packet:	2022-09-24 11:23:48
Elapsed:	00:07:18

Capture

Hardware:	Intel(R) Core(TM) i7-9700 CPU @ 3.00GHz (with SSE4.2)
OS:	64-bit Windows 10 (20H2), build 19042
Application:	Dumpcap (Wireshark) 3.6.8 (v3.6.8-0-gd25900c51508)

Interfaces

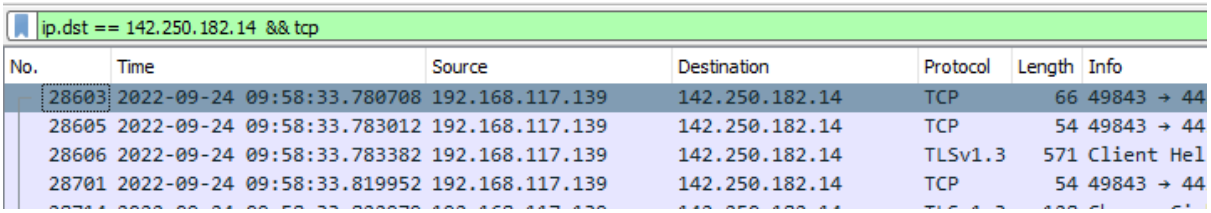
Interface	Dropped packets	Capture filter	Link type	Packet size limit (snaplen)
Ethernet	Unknown	none	Ethernet	262144 bytes

Statistics

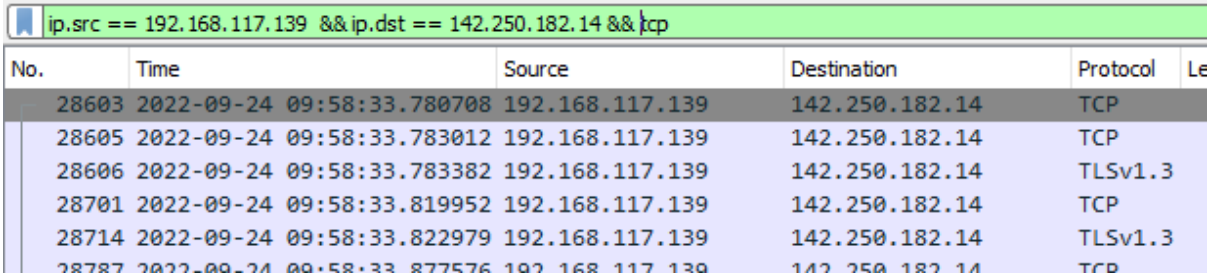
Measurement	Captured	Displayed	Marked
Packets	55852	418 (0.7%)	—
Time span, s	438.529	437.156	—
Average pps	127.4	1.0	—
Average packet size, B	958	237	—
Bytes	53530304	98958 (0.2%)	0
Average bytes/s	122 k	226	—
Average bits/s	976 k	1810	—

II. Use a DISPLAY filter expression to separate the packets sent by your computer vs. received from Facebook and YouTube in 5(b) and 5(c) above. Show the fractions for each type. For fractions and statistics, status bar in bottom can be viewed:

- For packets sent from my computer,



- Type ip.src == 192.168.1.12



From system to facebook- ip for facebook - 31.13.79.35

ip.src == 31.13.79.35

No.	Time	Source	Destination	Protocol	Length	Info
2152…	2022-09-24 10:21:43.517464	31.13.79.35	192.168.117.139	QUIC	1274	Initial, SCID=4dbd054f6da37b11, PKN:
2152…	2022-09-24 10:21:43.519280	31.13.79.35	192.168.117.139	QUIC	1274	Initial, SCID=4dbd054f6da37b11, PKN:
2152…	2022-09-24 10:21:43.519280	31.13.79.35	192.168.117.139	QUIC	228	Handshake, SCID=4dbd054f6da37b11
2152…	2022-09-24 10:21:43.519552	31.13.79.35	192.168.117.139	QUIC	90	Protected Payload (KP0)
2152…	2022-09-24 10:21:43.519552	31.13.79.35	192.168.117.139	QUIC	122	Protected Payload (KP0)
2152…	2022-09-24 10:21:43.546293	31.13.79.35	192.168.117.139	QUIC	81	Handshake, SCID=4dbd054f6da37b11
2152…	2022-09-24 10:21:43.546293	31.13.79.35	192.168.117.139	QUIC	154	Protected Payload (KP0)
2152…	2022-09-24 10:21:43.546644	31.13.79.35	192.168.117.139	QUIC	90	Protected Payload (KP0)
2152…	2022-09-24 10:21:43.547396	31.13.79.35	192.168.117.139	QUIC	90	Protected Payload (KP0)
2152…	2022-09-24 10:21:43.549864	31.13.79.35	192.168.117.139	QUIC	314	Protected Payload (KP0)
2152…	2022-09-24 10:21:43.682130	31.13.79.35	192.168.117.139	QUIC	1274	Protected Payload (KP0)
2152…	2022-09-24 10:21:43.682130	31.13.79.35	192.168.117.139	QUIC	1274	Protected Payload (KP0)
2152…	2022-09-24 10:21:43.682130	31.13.79.35	192.168.117.139	QUIC	378	Protected Payload (KP0)
2152…	2022-09-24 10:21:43.733431	31.13.79.35	192.168.117.139	QUIC	90	Protected Payload (KP0)

- Enter “Ctrl+Alt+Shift+C” view statistics of fraction of total sent from my computer

File

Name:	C:\Users\student\AppData\Local\Temp\wireshark_EthernetW2TMS1.pcapng
Length:	187 MB
Hash (SHA256):	ea86b47736a6cd218861da4e8570758476499523bc94ed2bfbbdab8bdb39ce32
Hash (RIPEMD160):	1ba757c3f9caa5fb6195d55e40f065f53403d308
Hash (SHA1):	8c13197755b2ee80eb7eee20287e69ebbe9e3323
Format:	Wireshark/… - pcapng
Encapsulation:	Ethernet

Time

First packet:	2022-09-24 09:55:48
Last packet:	2022-09-24 10:17:15
Elapsed:	00:21:26

Capture

Hardware:	Intel(R) Core(TM) i7-9700 CPU @ 3.00GHz (with SSE4.2)
OS:	64-bit Windows 10 (20H2), build 19042
Application:	Dumpcap (Wireshark) 3.6.8 (v3.6.8-0-gd25900c51508)

Interfaces

Interface	Dropped packets	Capture filter	Link type	Packet size limit (snaplen)
Ethernet	Unknown	none	Ethernet	262144 bytes

Statistics

Measurement	Captured	Displayed	Marked
Packets	209970	57473 (27.4%)	—
Time span, s	1286.935	1277.958	—
Average pps	163.2	45.0	—
Average packet size, B	858	223	—
Bytes	180180531	12835122 (7.1%)	0
Average bytes/s	140 k	10 k	—
Average bits/s	1120 k	80 k	—

- For packets sent from facebook and youtube,

File

Name:	C:\Users\student\AppData\Local\Temp\wireshark_EthernetW2TMS1.pcapng
Length:	187 MB
Hash (SHA256):	52de1ad760e23b5fb238fa432c64ff0e3a1c5b33caff0a79a0cafa97b523b91e
Hash (RIPEMD160):	5f353195fca7c5358e05983eea7cd511ca265431
Hash (SHA1):	ff679df040a2c19c829d1d8956ff82c086cb6f18
Format:	Wireshark/... - pcapng
Encapsulation:	Ethernet

Time

First packet:	2022-09-24 09:55:48
Last packet:	2022-09-24 10:19:04
Elapsed:	00:23:15

Capture

Hardware:	Intel(R) Core(TM) i7-9700 CPU @ 3.00GHz (with SSE4.2)
OS:	64-bit Windows 10 (20H2), build 19042
Application:	Dumpcap (Wireshark) 3.6.8 (v3.6.8-0-gd25900c51508)

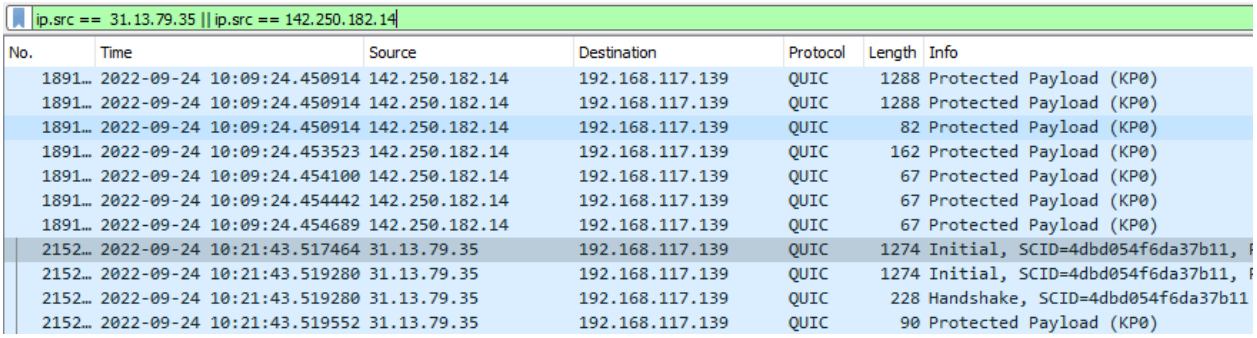
Interfaces

Interface	Dropped packets	Capture filter	Link type	Packet size limit (snaplen)
Ethernet	Unknown	none	Ethernet	262144 bytes

Statistics

Measurement	Captured	Displayed	Marked
Packets	211627	171 (0.1%)	—
Time span, s	1395.331	650.683	—
Average pps	151.7	0.3	—
Average packet size, B	854	700	—
Bytes	180738087	119684 (0.1%)	0
Average bytes/s	129 k	183	—
Average bits/s	1036 k	1471	—

- Type “ip.src == “www.facebook.com” || ip.src == ”www.youtube.com”



- Enter “Ctrl+Alt+Shift+C” to view statistics of fraction of total sent from my computer(displayed in percentage).

Details

File

Name:	C:\Users\dcl\AppData\Local\Temp\wireshark_EthernetRZ2KS1.pcapng
Length:	55 MB
Hash (SHA256):	ec0c58fc0a81016e730e046c3a5ab6d6ba10f19d38fcc2e945bcb4b49c151110
Hash (RIPEMD160):	be73b195666f66889f95cd100d8a587b49715b0e
Hash (SHA1):	a447ba577cd04def77c52755face3b45f1a63f83
Format:	Wireshark/... - pcapng
Encapsulation:	Ethernet

Time

First packet:	2022-09-24 11:16:29
Last packet:	2022-09-24 11:23:48
Elapsed:	00:07:18

Capture

Hardware:	Intel(R) Core(TM) i7-9700 CPU @ 3.00GHz (with SSE4.2)
OS:	64-bit Windows 10 (20H2), build 19042
Application:	Dumpcap (Wireshark) 3.6.8 (v3.6.8-0-gd25900c51508)

Interfaces

Interface	Dropped packets	Capture filter	Link type	Packet size limit (snaplen)
Ethernet	Unknown	none	Ethernet	262144 bytes

Statistics

Measurement	Captured	Displayed	Marked
Packets	55852	418 (0.7%)	—
Time span, s	438.529	437.156	—
Average pps	127.4	1.0	—
Average packet size, B	958	237	—
Bytes	53530304	98958 (0.2%)	0
Average bytes/s	122 k	226	—
Average bits/s	976 k	1810	—

7. Count how many TCP packets you received from / sent to Facebook or YouTube, and how many of each were also HTTP packets.

Received from YouTube TCP:

ip.dst == 142.250.196.14 && tcp

No.	Time	Source	Destination	Protocol	Length	Info
2488	10:45:27.789892	192.168.117.162	142.250.196.14	TCP	55	56825 → 4
3767	10:46:02.081408	192.168.117.162	142.250.196.14	TCP	54	56825 → 4
3778	10:46:02.083143	192.168.117.162	142.250.196.14	TCP	54	56825 → 4

Statistics

Measurement	Captured	Displayed	Marked
Packets	293253	16 (0.0%)	—
Time span, s	3960.638	290.056	—
Average pps	74.0	0.1	—
Average packet size, B	771	401	—
Bytes	226133149	6422 (0.0%)	0
Average bytes/s	57 k	22	—
Average bits/s	456 k	177	—

Sent to YouTube TCP:

ip.dst == 142.250.196.14 && tcp.port==443

No.	Time	Source	Destination	Protocol	Length	Info
2488	10:45:27.789892	192.168.117.162	142.250.196.14	TCP	55	56825 → 443 [ACK] Seq=1
3767	10:46:02.081408	192.168.117.162	142.250.196.14	TCP	54	56825 → 443 [FIN, ACK] S
3778	10:46:02.083143	192.168.117.162	142.250.196.14	TCP	54	56825 → 443 [ACK] Seq=3

Statistics

Measurement	Captured	Displayed	Marked
Packets	304291	14 (0.0%)	—
Time span, s	4032.574	290.056	—
Average pps	75.5	0.0	—
Average packet size, B	778	98	—
Bytes	236799209	1365 (0.0%)	0
Average bytes/s	58 k	4	—
Average bits/s	469 k	37	—

8. Analyze if during the course of a video session your client connected to multiple Youtube servers. Indicate approximately on the timeline where this occurred. Did packets with SYN or PSH flags occur at about the same time when your server changed? Provide some explanation as to why SYN/PSH packets were sent at all and if they were correlated with the server switching.

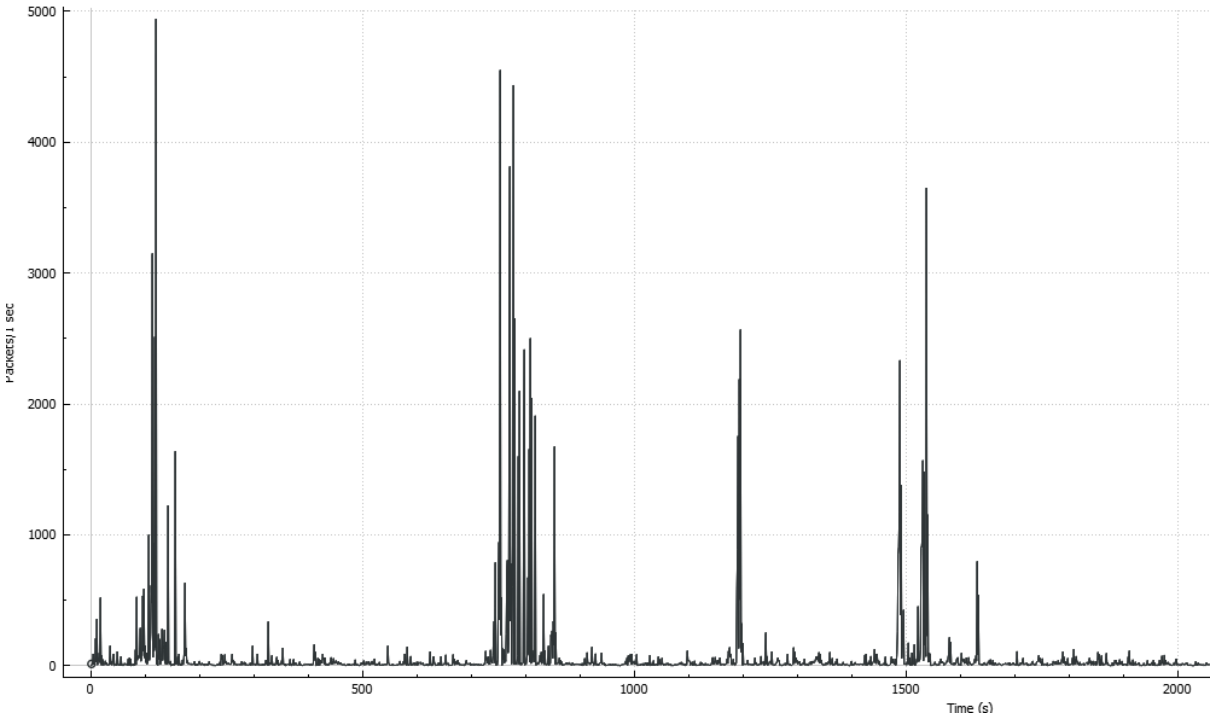
Syn Flag:

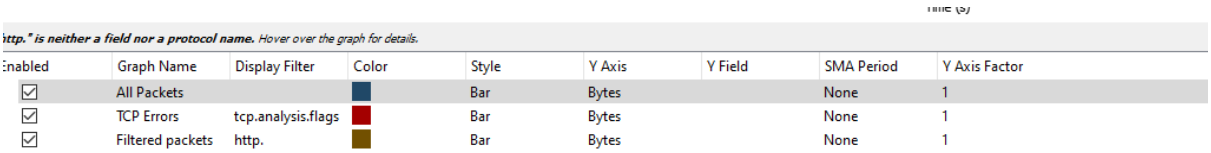
4918	89.975843	192.168.117.126	142.250.182.97	TCP	66 50109 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
4919	89.975903	192.168.117.126	142.250.182.97	TCP	66 50110 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
4920	89.975959	192.168.117.126	142.250.182.97	TCP	66 50111 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
4921	89.976013	192.168.117.126	142.250.182.97	TCP	66 50112 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1

Psh Flag:

5155	90.014147	142.250.182.97	192.168.117.126	TCP	1466 443 → 50111 [PSH, ACK] Seq=5649 Ack=518 Win=66816 Len=1412 [TCP segment of a reassembled PDU]
5157	90.014147	142.250.182.97	192.168.117.126	TLSv1.3	812 Application Data
5162	90.014368	142.250.182.97	192.168.117.126	TCP	1466 443 → 50110 [PSH, ACK] Seq=5649 Ack=518 Win=66816 Len=1412 [TCP segment of a reassembled PDU]

9. Analyze the Youtube packet sizes. Draw a histogram showing how many packets were received within a range of sizes. E.g., how many packets had length 0 - 100 bytes, 100 - 200 bytes, 200 - 300 bytes, etc. Indicate the packet size units (in bytes) on the horizontal axis.





ttp." is neither a field nor a protocol name. *Hover over the graph for details.*

Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Period	Y Axis Factor
☑	All Packets			Bar	Bytes		None	1
☑	TCP Errors	tcp.analysis.flags		Bar	Bytes		None	1
☑	Filtered packets	http.		Bar	Bytes		None	1

RESULT:

Thus the basic functionalities and features of Wireshark have been learnt successfully and implemented.

EXP NO: 5 DATE:	IMPLEMENTATION OF HTTP SERVER USING SOCKETS

AIM:

To create a Http Server using Sockets in Java.

ALGORITHM:

- Get the port number from the user.
- Create a Socket using ServerSocket java method.
- Connect the socket to the network using .accept().
- Read and write about the localhost site details using buffered reader and writer.
- Write required html code to run/print on the local website.
- Output is displayed in the localhost website.

CODE:

SETTING THE SERVER USING SOCKETS:

```
import java.net.*;
import java.io.*;
import java.util.Scanner;
class sock {
public static void main(String[] args) throws Exception {
System.out.println("Enter port : ");
    Scanner dop=new Scanner(System.in);
    int port=dop.nextInt();
ServerSocketserverSocket = new ServerSocket(port);
System.err.println("Local Host Server running at : " + port);
    while (true) {
        Socket clientSocket = serverSocket.accept();
System.err.println("Server connected!");
BufferedReader in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
BufferedWriter out = new BufferedWriter(new
OutputStreamWriter(clientSocket.getOutputStream()));
        String s;
        while ((s = in.readLine()) != null) {
System.out.println(s);
            if (s.isEmpty()) {
                break;
```

```
            }
        }

out.write("HTTP/1.0 200 OK\r\n");
out.write("Server: Apache/0.8.4\r\n");
out.write("Content-Type: text/html\r\n");
out.write("Content-Length: 59\r\n");
out.write("\r\n");
out.write("<TITLE>SuriyaaSite</TITLE>");
out.write("<P>Hello Broskies, This is Suriyaa 2020503550 here!!</P>");
out.write("<P>"+s+"</P>");
System.err.println("Server Connection Closed\n");
out.close();
in.close();
clientSocket.close();
    }
}
}
```

CUSTOM HTML INPUT FROM USER:

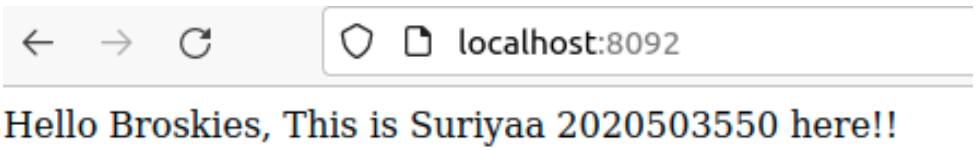
```
import java.net.*;
import java.io.*;
import java.util.*;
public class Server{
    public static void main(String[] args) throws IOException {
System.out.print("Enter port number : ");
        Scanner inn  = new Scanner(System.in);
        int port = inn.nextInt();
ServerSocketserverSocket = new ServerSocket(port);
System.err.println("Server is running on port: "+port);
        while(true){
            Socket clientSocket = serverSocket.accept();
System.err.println("Client connected");
BufferedReader in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
            String s;
while((s = in.readLine())!=null){
System.out.println(s);
                if(s.isEmpty()){
                    break;
                }
            }
            String str = "";
            str = inn.nextLine();
OutputStreamclientOutput = clientSocket.getOutputStream();
clientOutput.write("HTTP/1.1 200 OK\r\n".getBytes());
clientOutput.write("\r\n".getBytes());
clientOutput.write(str.getBytes());
clientOutput.write("\r\n\r\n".getBytes());
clientOutput.flush();
```

```
System.err.println("-----------------------------------Client connection
closed!----------------------------");
in.close();
clientOutput.close();
        }
    }
}
```

OUTPUT:

SETTING THE SERVER USING SOCKETS:

```
student@student-Veriton-M4660G:~/Desktop/3550$ java sock
Enter port :
8092
Local Host Server running at : 8092
Server connected!
GET / HTTP/1.1
Host: localhost:8092
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:104.0) Gecko/20100101 Firefo
x/104.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp
,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1

Server Connection Closed
```



Hello Broskies, This is Suriyaa 2020503550 here!!

CUSTOM HTML INPUT FROM USER:

```
^Cstudent@student-Veriton-M4660G:~$ javac Server.java
student@student-Veriton-M4660G:~$ java Server
Enter port number : 8083
Server is running on port: 8083
<h1 style="background-color:DodgerBlue;">Hello World</h1>
Client connected
GET / HTTP/1.1
Host: localhost:8083
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:104.0) Gecko/20100101 Firefox/104.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1

-----------------------------------Client connection closed!----------------------------
```

RESULT:

Thus, Http server at local host using Sockets was created and implemented using Java Programming.

EXP NO: 6 DATE:	IMPLEMENTATION OF SMTP SERVER

AIM:

To implement Simple message transfer protocol server in Java

ALGORITHM:

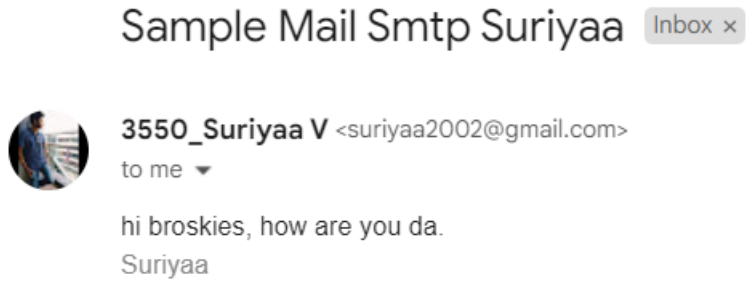
- Create properties to define and authorize a mail ID using smtp protocol
- Create a new mail sending session
- Javax mail authorization done
- Set recipients along with subject and the message and send the mail

CODE:

```
import java.util.Properties;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.PasswordAuthentication;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;
public class Smtp {
    public static void main(String[] args) {
        final String username = "suriyaa2002@gmail.com";
        final String password = "xxwdrnlyupwqkdsd";
        Properties props = new Properties();
        props.put("mail.smtp.auth", "true");
        props.put("mail.smtp.starttls.enable", "true");
        props.put("mail.smtp.host", "smtp.gmail.com");
        props.put("mail.smtp.port", "587");
        Session session = Session.getInstance(props,
                new javax.mail.Authenticator() {
                    protected PasswordAuthentication
getPasswordAuthentication() {
                        return new PasswordAuthentication(username, password);
                    }
                });
        try {
```

```
            Message message = new MimeMessage(session);
            message.setFrom(new InternetAddress("suriyaa2002@gmail.com"));
            message.setRecipients(Message.RecipientType.TO,
                    InternetAddress.parse("suriyaa2002@gmail.com"));
            message.setSubject("Sample Mail Smtp Suriyaa");
            message.setText("hi broskies, how are you da. \n Suriyaa");
            Transport.send(message);
            System.out.println("Mail sent to suriyaa2002@gmail.com");
        } catch (MessagingException e) {
            throw new RuntimeException(e);
        }
    }
}
```

OUTPUT:



RESULT:

Thus, Smtp mail Server has been implemented successfully.

EXP NO: 7 DATE:	DOMAIN NAME SERVER

AIM:

To implement domain name server (DNS) in Java.

ALGORITHM:

- Create a Socket
- Get the Inet address
- Get the desktop and Inet host address and store it in a string
- Browse and open in the web using desk.browse(URL)
- Dns has been created from the given IP

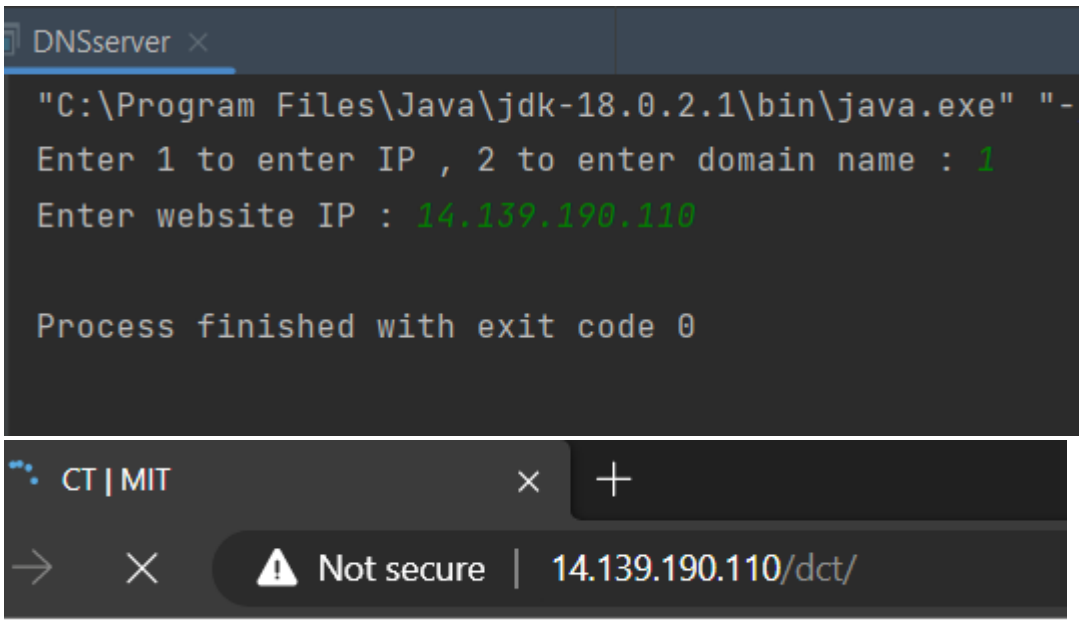
CODE:

```
import java.net.InetAddress;
import java.net.InetSocketAddress;
import java.net.URI;
import java.util.Scanner;
import java.awt.Desktop;
public class DNSserver {
    static String getHostString(InetSocketAddress socketAddress) {
        InetAddress address = socketAddress.getAddress();
        if (address == null) {
            // The InetSocketAddress was specified with a string (either a
numeric IP or a host name). If
            // it is a name, all IPs for that name should be tried. If it is an
IP address, only that IP
            // address should be tried.
            return socketAddress.getHostName();
        }
        // The InetSocketAddress has a specific address: we should only try
that address. Therefore we
        // return the address and ignore any host name that may be available.
        return address.getHostAddress();
    }

    public static void main(String[] args)throws Exception {
        Scanner in = new Scanner(System.in);
        System.out.print("Enter 1 to enter IP , 2 to enter domain name : ");
        String options = in.nextLine();
        if(options.equals("2")) {
            System.out.print("Enter website domain name : ");
        }
        else {
```

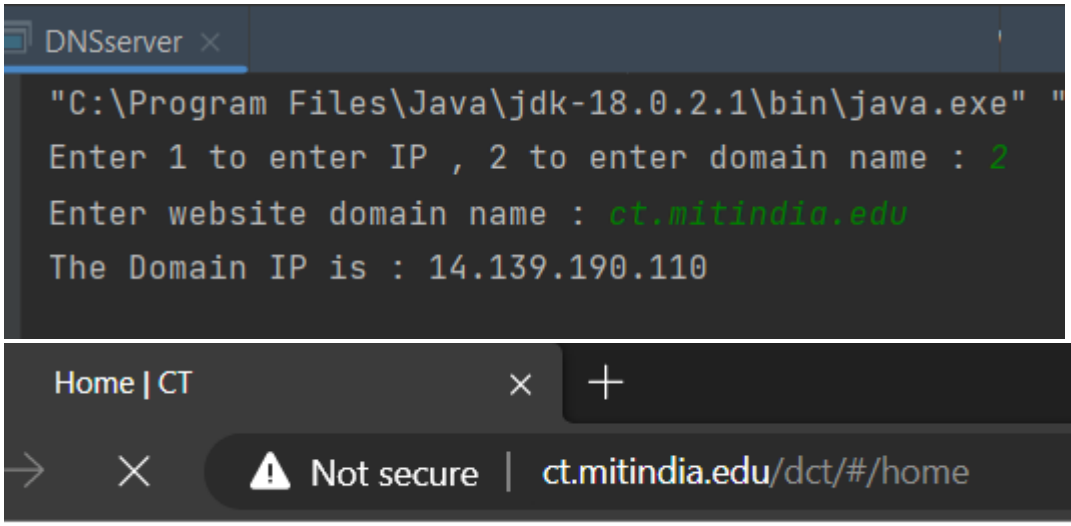
```
            System.out.print("Enter website IP : ");
        }
        String message;
        message=in.nextLine();
        Desktop desk = Desktop.getDesktop();
        InetAddress host = InetAddress.getByName(message);
        String q = host.toString();
        int k=q.indexOf('/');
        String ip = q.substring(k+1);
        if(options.equals("2")) {
            System.out.println("The Domain IP is : " + ip);
        }
        String mesage = "http://" + message;
        desk.browse(new URI(mesage));
    }
}
```

OUTPUT:

1. **IP to Domain**



2. **Domain to IP**



RESULT:

Thus, Domain name server has been implemented and checked successfully.

EXP NO: 8 DATE:	WEB CACHING

AIM:

To implement Web Caching in Domain name servers (DNS).

ALGORITHM:

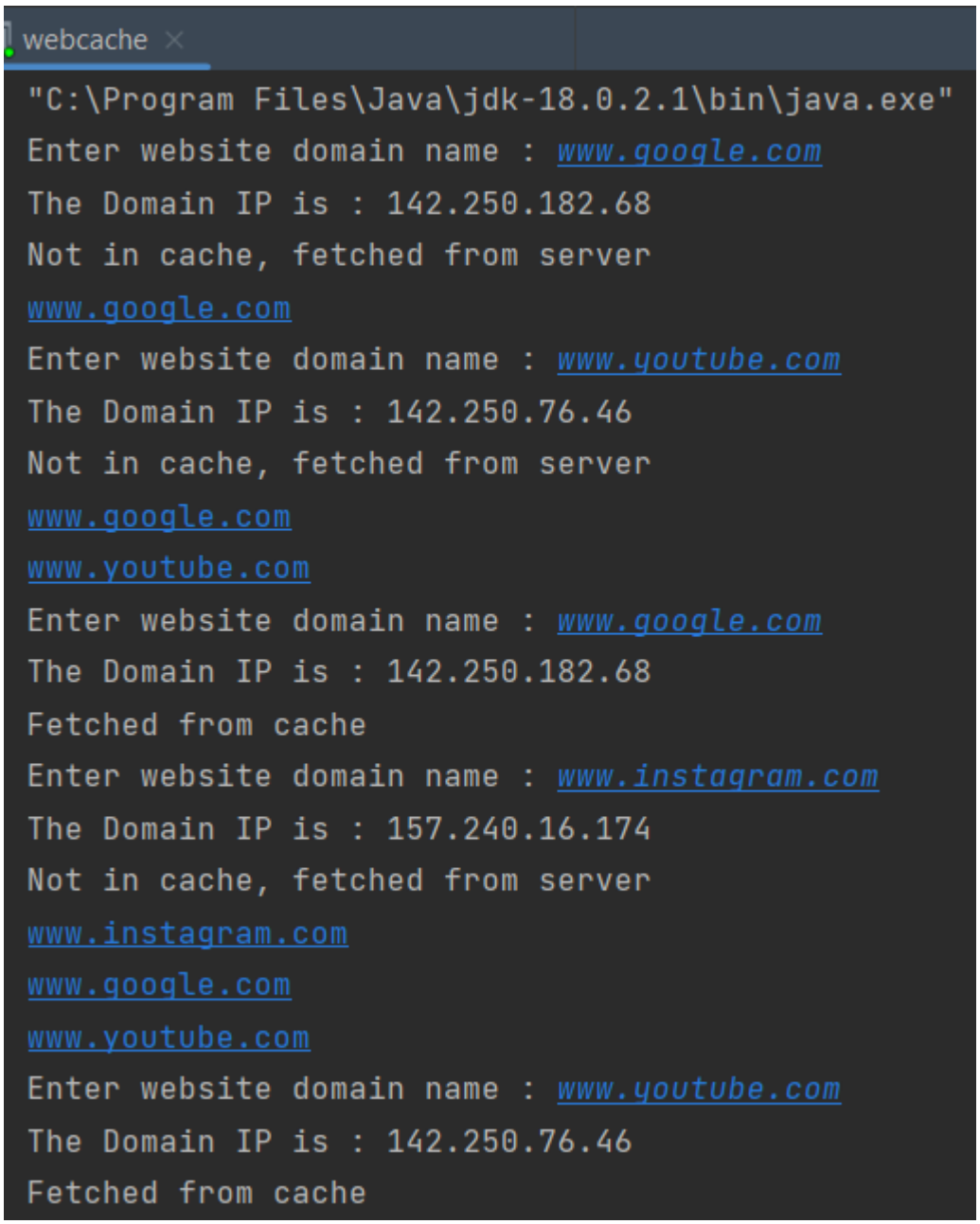
- Create a Domain Name Server.
- Enter a URL
- If present in the Cache set. Access it from there.
- If not present. Store it in the cache set.
- Next time access from the cache

CODE:

```
import java.sql.SQLOutput;
import java.util.*;
import java.net.*;
import java.awt.Desktop;

public class webcache {
    public static void main(String[] args) throws Exception {
        Scanner in=new Scanner(System.in);
        HashSet<String> set = new HashSet<String>();
        while(true) {
            System.out.print("Enter website domain name : ");
            String message;
            message = in.nextLine();
            if(message.equalsIgnoreCase("Exit")) break;
            Desktop desk = Desktop.getDesktop();
            InetAddress host = InetAddress.getByName(message);
            String msg = host.toString();
            int k = msg.indexOf('/');
            String ip = msg.substring(k + 1);

            System.out.println("The Domain IP is : " + ip);
            desk.browse(new URI(message));
            if (set.contains(message))
           {
                System.out.println("Fetched from cache");
            }
            else {
                System.out.println("Not in cache, fetched from server");
                set.add(message);
                Iterator<String> i=set.iterator();
```

```
                while(i.hasNext())
                {
                    System.out.println(i.next());
                }
            }
        }
    }
}
```

OUTPUT:



RESULT:

Thus, Web Caching in Domain name servers (DNS) has been successfully implemented.

EXP NO: 9 DATE:	TCP FLOW CONTROL

AIM:

To implement the TCP Flow control algorithms

1. Stop and Wait
2. Go Back N
3. Selective Repeat

STOP AND WAIT:

ALGORITHM:

Sender side:

1. Sender sends one data packet at a time.
2. Sender sends the next packet only when it receives the acknowledgment of the previous packet.

Receiver side:

1. Receive and then consume the data packet.
2. When the data packet is consumed, receiver sends the acknowledgment to the sender.

CODE:

Sender:

```
Import java.io.*;
import java.net.*;

public class Sender {

  Socket sender;
  ObjectOutputStream out;
  ObjectInputStream in;
  String packet, ack, str, msg;
  int n, i = 0, sequence = 0;

  Sender() {}

  Public void run() {
    try {
      BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
      System.out.println("Waiting for Connection....");
      sender = new Socket("localhost", 2005);
```

```
      sequence = 0;
      out = new ObjectOutputStream(sender.getOutputStream());
      out.flush();
      in = new ObjectInputStream(sender.getInputStream());
      str = (String) in.readObject();
      System.out.println("reciver     > " + str);
      System.out.println("Enter the data to send....");
      packet = br.readLine();
      n = packet.lengt h();
      do {
        try {
          if (i< n) {
            msg = String.valueOf(sequence);
            msg = msg.concat(packet.substring(i, i + 1));
          } elseif (i == n) {
            msg = "end";
            out.writeObject(msg);
            break;
          }
          out.writeObject(msg);
          sequence = (sequence == 0) ?1:0;
          out.flush();
          System.out.println("data sent>" + msg);
          ack = (String) in.readObject();
          System.out.println("waiting for ack.....\n\n");
          if (ack.equals(String.valueOf(sequence))) {
            i++;
            System.out.println("receiver   >  " + " packet recieved\n\n");
          } else {
            System.out.println("Time out resending data....\n\n");
            sequence = (sequence == 0) ?1:0;
          }
        } catch (Exception e) {}
      } while (i< n + 1);
      System.out.println("All data sent. exiting.");
    } catch (Exception e) {} finally {
      try {
        in.close();
        out.close();
        sender.close();
      } catch (Exception e) {}
    }
  }

  Public static void main(String args[]) {
    Sender s = new Sender();
    s.run();
  }
}
```

Receiver:

```
Import java.io.*;
```

```
Import java.net.*;

Public class Receiver {

  ServerSocket reciever;
  Socket connection = null;
  ObjectOutputStream out;
  ObjectInputStream in;
  String packet, ack, data = "";
  inti = 0, sequence = 0;

  Receiver() {}

  Public void run() {
    try {
      BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
      reciever = new ServerSocket(2005, 10);
      System.out.println("waiting for connection...");
      connection = reciever.accept();
      sequence = 0;
      System.out.println("Connection established   :");
      out = new ObjectOutputStream(connection.getOutputStream());
      out.flush();
      in = new ObjectInputStream(connection.getInputStream());
      out.writeObject("connected    .");

      do {
        try {
          packet = (String) in.readObject();
          if (Integer.valueOf(packet.substring(0, 1)) == sequence) {
            data += packet.substring(1);
            sequence = (sequence == 0) ?1:0;
            System.out.println("\n\nreceiver         >" + packet);
          } else {
            System.out.println(
              "\n\nreceiver         >" + packet + "   duplicate data"
            );
          }
          if (i<3) {
            out.writeObject(String.valueOf(sequence));
            i++;
          } else {
            out.writeObject(String.valueOf((sequence + 1) % 2));
            i = 0;
          }
        } catch (Exception e) {}
      } while(!packet.equals("end"));
      System.out.println("Data received=" + data);
      out.writeObject("connection ended    .");
    } catch (Exception e) {} finally {
      try {
        in.close();
        out.close();
        reciever.close();
```

```
      } catch (Exception e) {}
    }
  }

  Public static void main(String args[]) {
    Receiver s = new Receiver();
    while (true) {
      s.run();
    }
  }
}
```

OUTPUT:

Receiver:

```
SNW_Receiver [Java Application] C:\Program Files\Java\jdk-18
waiting for connection...
Connection established   :
receiver          >0s
receiver          >1u
receiver          >0r
receiver          >1i
receiver          >1i    duplicate data
receiver          >0y
receiver          >1a
receiver          >0a
receiver          >0a    duplicate data
Data received= suriyaa
```

Sender:

```
<terminated> SNW_Sender [Java Application] C:\Program Files\Java\
Waiting for Connection....
receiver     > connected    .
Enter the data to send....
suriyaa
data sent>0s
waiting for ack.....
receiver   >   packet recieved

data sent>1u
waiting for ack.....
receiver   >   packet recieved

data sent>0r
waiting for ack.....
receiver   >   packet recieved

data sent>1i
waiting for ack.....
Time out resending data....
data sent>1i
waiting for ack.....
receiver   >   packet recieved

data sent>0y
waiting for ack.....
receiver   >   packet recieved

data sent>1a
waiting for ack.....
receiver   >   packet recieved

data sent>0a
waiting for ack.....
Time out resending data....
data sent>0a
waiting for ack.....
receiver   >   packet recieved

All data sent. exiting.
```

GO BACK N PROTOCOL:

Algorithm:

1. Sender sends the N packet at time
2. If the acknowledgment for the first frame is received then the window moves to the next frame
3. Else entire the packet/frame in the window is resend until the acknowledgement is received
4. Repeat step 1 – 4 until all packet is send

Code:

Sender:

```
Import java.io.*;
Import java.net.*;
Import java.util.*;

Public class Go_Back_N_Client {

  Public static voidmain(String args[]) throwsException {
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    System.out.print("Enter the value of m : ");
    int m = Integer.parseInt(br.readLine());
    int x = (int) ((Math.pow(2, m)) - 1);
    System.out.print("Enter no. of frames to be sent: ");
    int count = Integer.parseInt(br.readLine());
    int data[] = new int[count];
    int h = 0;
    for (int i = 0; i< count; i++) {
      System.out.print("Enter data for frame no " + h + " => ");
      data[i] = Integer.parseInt(br.readLine());
      h = (h + 1) % x;
    }
    Socket client = new Socket("localhost", 55004);
    ObjectInputStream ois = new ObjectInputStream(client.getInputStream());
    ObjectOutputStream oos = new ObjectOutputStream(client.getOutputStream());
    System.out.println("Connected with server");

    boolean flag = false;
    GoBackNListener listener = new GoBackNListener(ois, x);
    listener.t.start();
    intstrt = 0;
    h = 0;
    oos.writeObject(x);
    do {
      int c = h;
      for (int i = h; i< count; i++) {
        System.out.print("|" + c + "|");
        c = (c + 1) % x;
      }
      System.out.println();
      System.out.println();
      h = strt;
      for (int i = strt; i< count; i++) {
        System.out.println("Sending frame:" + h);
        h = (h + 1) % x;
        System.out.println();
        oos.writeObject(i);
        oos.writeObject(data[i]);
        Thread.sleep(100);
      }
      listener.t.join(4000);
      if (listener.reply != count - 1) {
        System.out.println(
          "No reply from server in 4 seconds. Resending data from frame no " +
```

```
          (listener.reply + 1)
        );
        System.out.println(
          "Listener.reply value: " + listener.reply + "\nx-1: " + (x - 1)
        );
        System.out.println();
        strt = listener.reply + 1;
        flag = false;
      } else {
        System.out.println("All elements sent successfully. Exiting");
        flag = true;
      }
    } while(!flag);

    oos.writeObject(-1);
  }
}

Class GoBackNListener implements Runnable {

  Thread t;
  ObjectInputStream ois;
  int reply, x;

  GoBackNListener(ObjectInputStream o, int i) {
    t = new Thread(this);
    ois = o;
    reply = -2;
    x = i;
  }

  @Override
  public void run() {
    try {
      int temp = 0;
      while (reply != -1) {
        reply = (Integer) ois.readObject();
        if (reply != -1&& reply != temp + 1) reply = temp;
        if (reply != -1) {
          temp = reply;
          System.out.println(
            "Acknowledgement of frame no " + (reply % x) + " recieved."
          );
          System.out.println();
        }
      }
      reply = temp;
    } catch (Exception e) {
      System.out.println("Exception => " + e);
    }
  }
}
```

Receiver:

```
Import java.io.*;
Import java.net.*;
Import java.util.*;

public class Go_Back_N_Server {

  public static void main(String[] args) throwsException {
    ServerSocket server = new ServerSocket(55004);
    System.out.println("Server established");
    Socket client = server.accept();
    ObjectOutputStream oos = new ObjectOutputStream(client.getOutputStream());
    ObjectInputStream ois = new ObjectInputStream(client.getInputStream());
    System.out.println("Client is now connected");
    int x = (Integer) ois.readObject();
    int k = (Integer) ois.readObject();
    int j = 0;
    int i = (Integer) ois.readObject();
    boolean flag = true;
    Random r = newRandom(6);
    int mod = r.nextInt(6);
    while (mod == 1 || mod == 0) mod = r.nextInt(6);
    while (true) {
      int c = k;
      for (int h = 0; h <= x; h++) {
        System.out.print("|" + c + "|");
        c = (c + 1) % x;
      }
      System.out.println();
      System.out.println();
      if (k == j) {
        System.out.println("Frame " + k + " recieved" + "\n" + "Data:" + j);
        j++;
        System.out.println();
      } else System.out.println(
        "Frames recieved not in correct order" +
        "\n" +
        " Expected frame:" +
        j +
        "\n" +
        " Recieved frame no :" +
        k
      );
      System.out.println();
      if (j % mod == 0&& flag) {
        System.out.println("Error found. Acknowledgement not sent. ");
        flag = !flag;
        j--;
      } elseif (k == j - 1) {
        oos.writeObject(k);
        System.out.println("Acknowledgement sent");
      }
```

```
      System.out.println();
      if (j % mod == 0) flag = !flag;
      k = (Integer) ois.readObject();
      if (k == -1) break;
      i = (Integer) ois.readObject();
    }
    System.out.println("Client finished sending data. Exiting");
    oos.writeObject(-1);
  }
}
```

Output

Receiver:

```
Server established
Client is now connected
|0||1||2||3||4||5||6||0|

Frame 0 recieved
Data:0


Acknowledgement sent

|1||2||3||4||5||6||0||1|

Frame 1 recieved
Data:1


Error found. Acknowledgement not sent.

|2||3||4||5||6||0||1||2|

Frames recieved not in correct order
 Expected frame:1
 Recieved frame no :2


|3||4||5||6||0||1||2||3|

Frames recieved not in correct order
 Expected frame:1
 Recieved frame no :3


|4||5||6||0||1||2||3||4|

Frames recieved not in correct order
 Expected frame:1
 Recieved frame no :4


|1||2||3||4||5||6||0||1|
Frame 3 recieved
Data:3


Acknowledgement sent

|4||5||6||0||1||2||3||4|

Frame 4 recieved
Data:4


Acknowledgement sent

Client finished sending data. Exiting
```

Sender:

```
Enter the value of m : 3
Enter no. of frames to be sent: 5
Enter data for frame no 0 => 1
Enter data for frame no 1 => 2
Enter data for frame no 2 => 3
Enter data for frame no 3 => 4
Enter data for frame no 4 => 5
Connected with server
|0||1||2||3||4|

Sending frame:0

Acknowledgement of frame no 0 recieved.

Sending frame:1

Sending frame:2

Sending frame:3

Sending frame:4

No reply from server in 4 seconds. Resending data from frame no 1
Listener.reply value: 0
x-1: 6



Sending frame:1

Acknowledgement of frame no 1 recieved.

Sending frame:2

Acknowledgement of frame no 2 recieved.

Sending frame:3

Sending frame:4

No reply from server in 4 seconds. Resending data from frame no 3
Listener.reply value: 2
x-1: 6



Sending frame:3

 Acknowledgement of frame no 3 recieved.

 Sending frame:4

 Acknowledgement of frame no 4 recieved.

 All elements sent successfully. Exiting
```

SELECTIVE REPEAT PROTOCOL:

Algorithm:

1. Sender sends the N packet at time
2. If the acknowledgment for the first frame is received then the window moves to the next frame
3. Else particular packet/frame in the window is resend until the acknowledgement is received
4. Repeat step 1 – 4 until all packet is send

Code:

Sender:

```
Import java.lang.System;
Import java.net.*;
Import java.io.*;
Import java.util.Random;

Public class SRc {
    Static Socket connection;

    Public static void main(String a[]) throws SocketException {
        try {
            int v[] = new int[10];
            int n = 0;
            Random rands = new Random();
            int rand = 0;

            InetAddress addr = InetAddress.getByName("Localhost");
            System.out.println(addr);
            connection = new Socket(addr, 8011);
            DataOutputStream out = new DataOutputStream(
                    connection.getOutputStream());
            DataInputStream in = new DataInputStream(
                    connection.getInputStream());
            int p = in.read();
            System.out.println("No of frame is:" + p);

            for (int i = 0; i< p; i++) {
                v[i] = in.read();
                System.out.println(v[i]);
            }
            rand = rands.nextInt(p);//FRAME NO. IS RANDOMLY GENERATED
            v[rand] = -1;
            for (int i = 0; i< p; i++){
                    System.out.println("Received frame is: " + v[i]);

            }
            for (int i = 0; i< p; i++)
                if (v[i] == -1) {
                    System.out.println("Request to retransmit from packet no "
                            + (i+1) + " again!!");
                    n = i;
```

```
                    out.write(n);
                    out.flush();
                }
            System.out.println();
                v[n] = in.read();
                System.out.println("Received frame is: " + v[n]);
            System.out.println("quiting");
        }
        catch (Exception e) {
            System.out.println(e);
        }

    }
}
```

Receiver:

```
Import java.io.DataInputStream;
Import java.io.DataOutputStream;
Import java.io.IOException;
Import java.net.ServerSocket;
Import java.net.Socket;
Import java.net.SocketException;

public class SRs {
    static ServerSocket Serversocket;
    static DataInputStream dis;
    static DataOutputStream dos;

    public static void main(String[] args) throws SocketException {
        try {
            inta[] = { 30, 40, 50, 60, 70, 80, 90, 100 };
            Serversocket = new ServerSocket(8011);
            System.out.println("waiting for connection");
            Socket client = Serversocket.accept();
            dis = new DataInputStream(client.getInputStream());
            dos = new DataOutputStream(client.getOutputStream());
            System.out.println("The number of packets sent is:" + a.length);
            int y = a.length;
            dos.write(y);
            dos.flush();
            for (int i = 0; i<a.length; i++) {
                dos.write(a[i]);
                dos.flush();
            }
            int k = dis.read();
            dos.write(a[k]);
            dos.flush();
        }
        catch (IOException e){
            System.out.println(e);
        }
        finally{
```

```
            try{
                dis.close();
                dos.close();
            }
            catch (IOExceptione){
                e.printStackTrace();
            }
        }
    }
}
```

Output:

Sender:

```
Localhost/127.0.0.1
No of frame is:8
30
40
50
60
70
80
90
100
Received frame is: -1
Received frame is: 40
Received frame is: 50
Received frame is: 60
Received frame is: 70
Received frame is: 80
Received frame is: 90
Received frame is: 100
Request to retransmit from packet no 1 again!!

Received frame is: 30
quiting
```

Receiver:

```
waiting for connection
The number of packets sent is:8
```

RESULT:

Thus, the three TCP Flow control protocols have been tested and implemented successfully.

EXP NO: 10 DATE:	ROUTING ALGORITHMS

AIM:

To implement the Routing Algorithms using Java

a) Link State Routing

b) Distance Vector Routing

ALGORITHM:

LINK STATE ROUTING:

a) Get the no of nodes followed by the cost matrix.

b) Start from a node. Check all the adjacent nodes distance.

c) Get the shortest path node and generate the path.

d) Repeat until the entire graph/network is traversed.

DISTANCE VECTOR ROUTING:

a) Get the number of nodes along with the edge count.

b) Get each connected edges weight separately.

c) Get source and destination

d) Generate shortest path using Distant Vector algorithm.

CODE:

LINK STATE ROUTING:

```
import java.util.*;
public class LinkRoute {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number of nodes : ");
        int nodes = sc.nextInt();
        int[] preD = new int[nodes];
        int min = 999, nextNode = 0;
        int[] distance = new int[nodes];
        int[][] matrix = new int[nodes][nodes];
        int[] visited = new int[nodes];
        System.out.println("Enter the cost matrix");
        for (int i = 0; i < distance.length; i++) {
            visited[i] = 0;
            preD[i] = 0;
            for (int j = 0; j < distance.length; j++) {
                matrix[i][j] = sc.nextInt();
                if (matrix[i][j]==0)
                    matrix[i][j] = 999;
            }
        }
```

```
        distance = matrix[0];
        visited[0] = 1;
        distance[0] = 0;
        for (int counter = 0; counter < nodes; counter++) {
            min = 999;
            for (int i = 0; i < nodes; i++) {
                if (min > distance[i] && visited[i]!=1) {
                    min = distance[i];
                    nextNode = i;
                }
            }

            visited[nextNode] = 1;
            for (int i = 0; i < nodes; i++)
                if (visited[i]!=1)
                    if (min+matrix[nextNode][i] < distance[i]) {
                        distance[i] = min+matrix[nextNode][i];
                        preD[i] = nextNode;
                    }
        }


        int j;
        for (int i = 0; i < nodes; i++) {
            if (i!=0) {
                System.out.print("Path = " + i);
                j = i;
                do {
                    j = preD[j];
                    System.out.print(" <- " + j);
                }
                while(j != 0);
                System.out.println();
                System.out.print("Cost = " + distance[i]);
            }
            System.out.println("\n");
        }
    }
}
```

DISTANCE VECTOR ROUTING:

```
import java.io.*;
import java.util.Scanner;
public class DVR {
    static int graph[][],via[][],rt[][],v,e;
    public static void main(String args[]) throws IOException {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        System.out.println("Please enter the number of Vertices: ");
        v = Integer.parseInt(br.readLine());
        System.out.println("Please enter the number of Edges: ");
        e = Integer.parseInt(br.readLine());
        graph = new int[v][v];
```

```
        via = new int[v][v];
        rt = new int[v][v];
        for(int i = 0; i < v; i++)
            for(int j = 0; j < v; j++)
                if(i == j)
                    graph[i][j] = 0;
                else
                    graph[i][j] = 9999;
        for(int i = 0; i < e; i++) {
            System.out.println("Please enter data for Edge " + (i + 1) + ":");
            System.out.print("Source: ");
            int s = Integer.parseInt(br.readLine()); s--;
            System.out.print("Destination: ");
            int d = Integer.parseInt(br.readLine()); d--;
            System.out.print("Cost: ");
            int c = Integer.parseInt(br.readLine());
            graph[s][d] = c;
            graph[d][s] = c;
        }
        dvr_calc_disp("The Routing Tables are: ");
        Scanner in=new Scanner(System.in);
        System.out.print("Enter source: ");
        int so = in.nextInt();
        System.out.print("Enter Destination: ");
        int de = in.nextInt();
        de--; so--;
        int j = so;
        System.out.print("Path: ");
        System.out.print(so+1);
        while(j!=de){
            System.out.print("->"+(via[j][de]+1));
            j = via[j][de];
        }
        System.out.println();
    }

    static void dvr_calc_disp(String message) {
        System.out.println();
        init_tables();
        update_tables();
        System.out.println(message);
        print_tables();
        System.out.println();
    }

    static void update_table(int source) {
        for(int i = 0; i < v; i++) {
            if(graph[source][i] != 9999) {
                int dist = graph[source][i];
                for(int j = 0; j < v; j++) {
                    int inter_dist = rt[i][j];
                    if(via[i][j] == source)
                        inter_dist = 9999;
                    if(dist + inter_dist < rt[source][j]) {
```

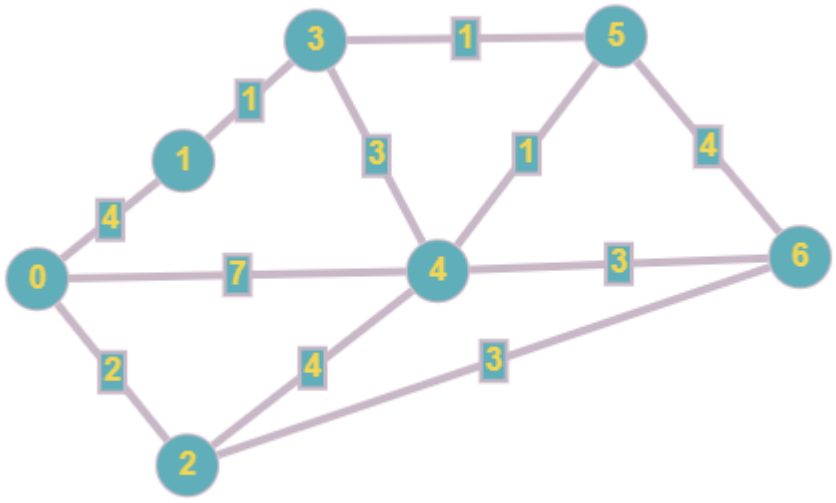
```
                        rt[source][j] = dist + inter_dist;
                        via[source][j] = i;
                    }
                }
            }
        }
    }
    static void update_tables() {
        int k = 0;
        for(int i = 0; i < 4*v; i++) {
            update_table(k);
            k++;
            if(k == v)
                k = 0;
        }
    }
static void init_tables() {
        for(int i = 0; i < v; i++)
            for(int j = 0; j < v; j++)
                if(i == j) {
                    rt[i][j] = 0;
                    via[i][j] = i;
                }
                else {
                    rt[i][j] = 9999;
                    via[i][j] = 100;
                }
    }

    static void print_tables() {
        for(int i = 0; i < v; i++) {
            for(int j = 0; j < v; j++)
                System.out.print("Dist: " + rt[i][j]+"  ");
            System.out.println();
        }
    }
}
```

OUTPUT:

LINK STATE ROUTING:

LinkRoute

```
"C:\Program Files\Java\jdk-19\bin
Enter the number of nodes : 7
Enter the cost matrix
0 4 2 1000 7 1000 1000
4 0 1000 1 1000 1000 1000
2 1000 0 1000 4 1000 3
1000 1 1000 0 3 1 1000
7 1000 4 3 0 1 3
1000 1000 1000 1 1 0 4
1000 1000 1000 1000 3 4 0


Path = 1 <- 0
Cost = 4

Path = 2 <- 0
Cost = 2

Path = 3 <- 1 <- 0
Cost = 5

Path = 4 <- 2 <- 0
Cost = 6

Path = 5 <- 3 <- 1 <- 0
Cost = 6

Path = 6 <- 2 <- 0
Cost = 5
```



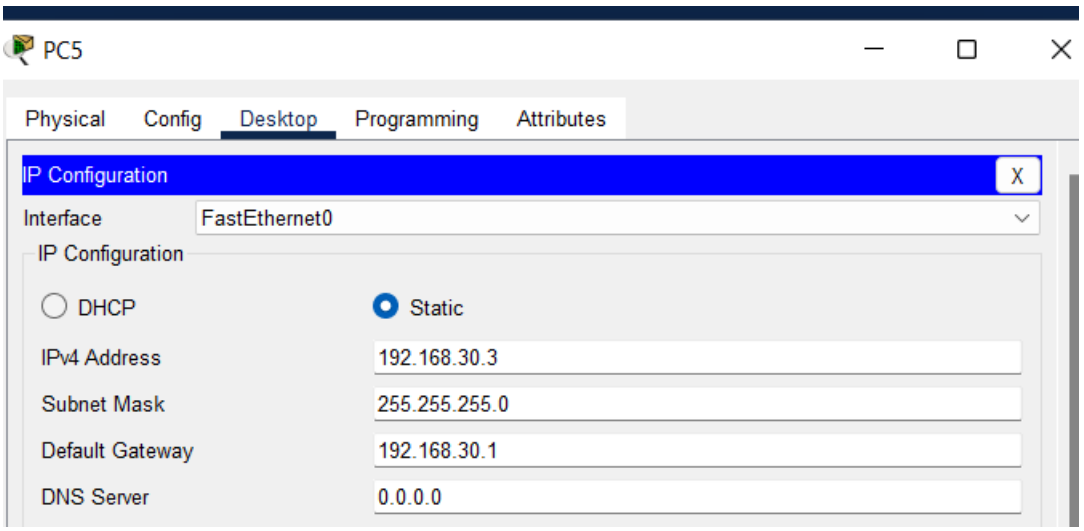
DISTANCE VECTOR ROUTING:

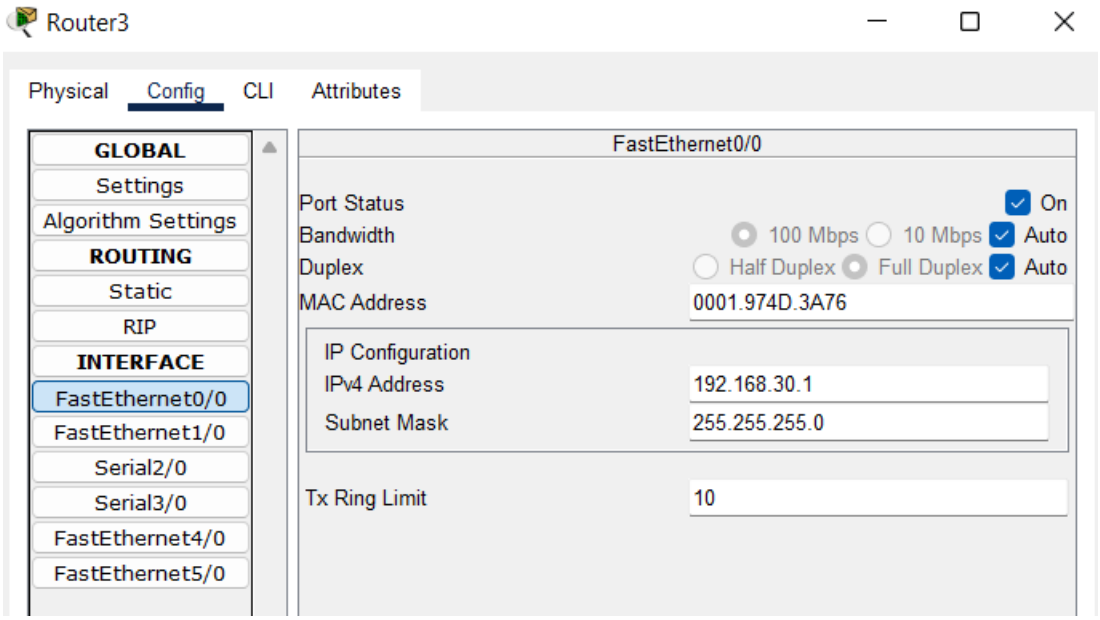
```
DVR
"C:\Program Files\Java\jdk-19\bin\java.ex
Please enter the number of Vertices:
5
Please enter the number of Edges:
5
Please enter data for Edge 1:
Source: 1
Destination: 5
Cost: 5
Please enter data for Edge 2:
Source: 2
Destination: 3
Cost: 6
Please enter data for Edge 3:
Source: 3
Destination: 4
Cost: 3
Please enter data for Edge 4:
Source: 4
Destination: 5
Cost: 7
Please enter data for Edge 5:
Source: 2
Destination: 5
Cost: 4
```

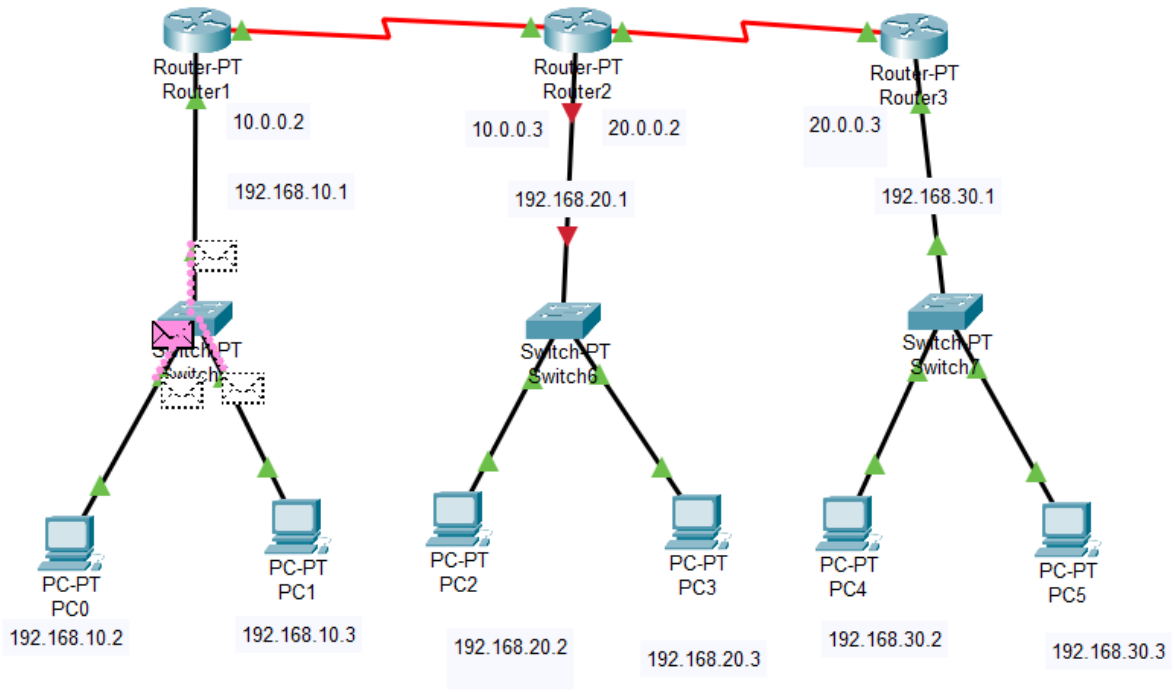
```
The Routing Tables are:
Dist: 0  Dist: 9  Dist: 15  Dist: 12  Dist: 5
Dist: 9  Dist: 0  Dist: 6  Dist: 9  Dist: 4
Dist: 15  Dist: 6  Dist: 0  Dist: 3  Dist: 10
Dist: 12  Dist: 9  Dist: 3  Dist: 0  Dist: 7
Dist: 5  Dist: 4  Dist: 10  Dist: 7  Dist: 0

Enter source: 1
Enter Destination: 3
Path: 1->5->2->3
```

ROUTING IN CISCO PACKET TRACER:

DISTANCE VECTOR ROUTING:







Simulation Panel

Event List

Vis.	Time(sec)	Last Device
	0.002	Switch5
	0.002	--
	0.002	Switch6
	0.002	--
	0.003	Switch5
	0.003	Router1
	0.004	Switch5
	0.385	--
	0.386	Switch7
	0.386	Switch7
	0.386	Switch7
Visible	0.402	--

Reset Simulation | Constant Delay | Captured to: 0.402 s

LINK STATE ROUTING:

PC0

Physical | Config | Desktop | Programming | Attributes

IP Configuration

Interface: FastEthernet0

IP Configuration

DHCP | Static

IPv4 Address: 192.168.1.2

Subnet Mask: 255.255.255.0

Default Gateway: 192.168.1.1

DNS Server: 0.0.0.0

Router0

Physical | Config | CLI | Attributes

GLOBAL
Settings
Algorithm Settings
ROUTING
Static
RIP
INTERFACE
FastEthernet0/0
FastEthernet1/0
Serial2/0
Serial3/0
FastEthernet4/0
FastEthernet5/0

FastEthernet0/0

Port Status: On

Bandwidth: 100 Mbps | 10 Mbps | Auto

Duplex: Half Duplex | Full Duplex | Auto

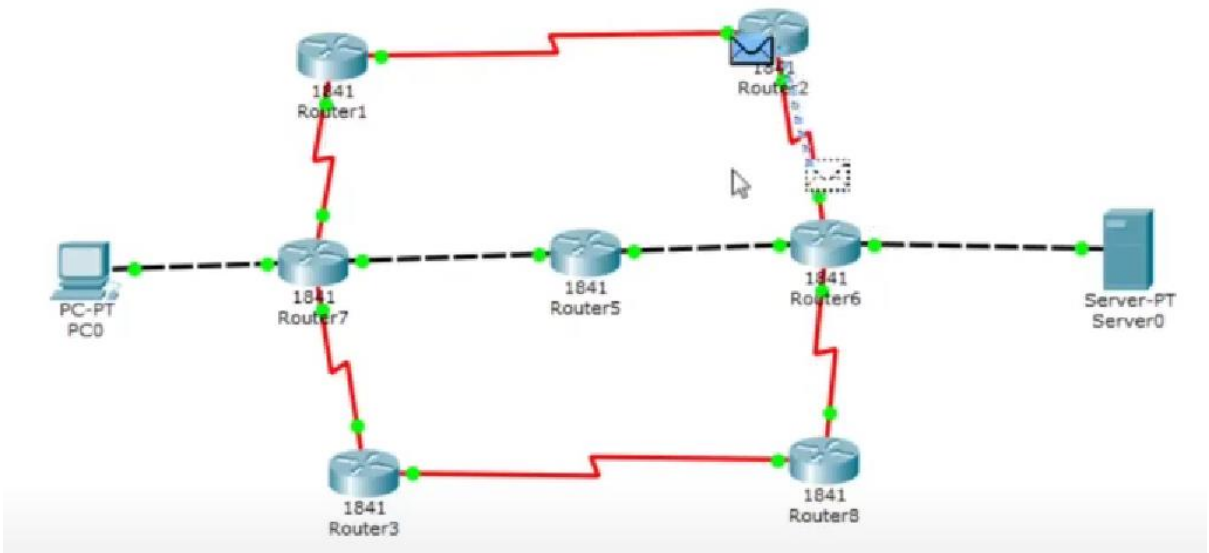
MAC Address: 0001.C9CD.D410

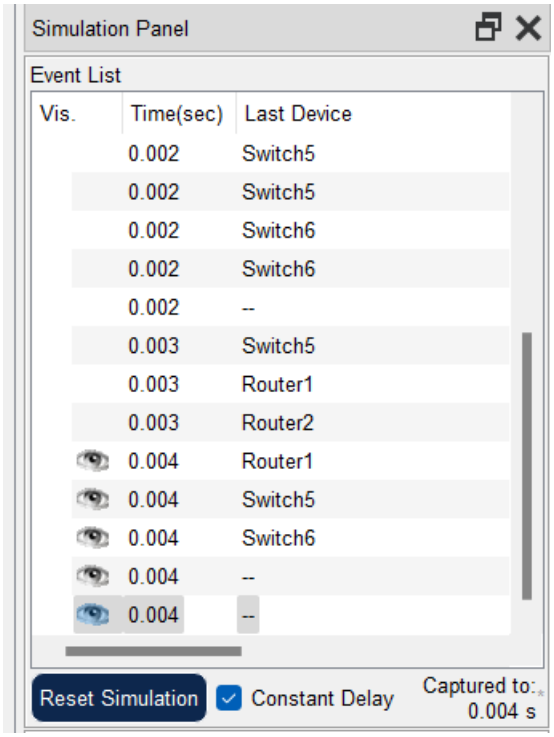
IP Configuration

IPv4 Address: 192.168.1.1

Subnet Mask: 255.255.255.0

Tx Ring Limit: 10





RESULT:

Thus. both the routing algorithms have been successfully implemented and verified.

EXP NO: 11 DATE:	IMPLEMENTATION OF ERROR DETECTION AND CORRECTION

AIM:

To implement the Routing Algorithms using Java

a) Check Sum

b) Cyclic Redundancy Check

ALGORITHM:

CHECK SUM:

a) Get the input string 1 and 2.

b) Generate its check sum by adding the ASCII values of each letter present.

c) If the checksum values are same. No single digit error.

d) If not, the strings failed checksum confirmation.

CYCLIC REDUNDANCY CHECK:

a) Get data and key from the user.

b) Perform XOR division for Data/key.

c) Find the remainder and Append it to the Data string at the last.

d) Perform XOR division again by dividing and XOR adding

e) If remainder is 0 then, No error else CRC is present.

CODE:

CHECK SUM:

```
import java.util.Scanner;
public class CheckSum {
    public static void main(String[] args){
        Scanner in = new Scanner(System.in);
        System.out.print("Enter message 1 : ");
        String s;
        s = in.nextLine();
        System.out.print("Enter message 2 : ");
        String r;
        r = in.nextLine();
        byte[] arr1 = s.getBytes();
        byte[] arr2 = r.getBytes();
        int q1 = 0 , q2 = 0;
        String w1="",w2="";
        for(int i = 0 ; i < arr1.length ; i++){
            q1 += arr1[i];
            w1 += Integer.toBinaryString(arr1[i]);
            q2 += arr2[i];
```

```
            w2 += Integer.toBinaryString(arr2[i]);
        }
        System.out.println("Binary of Input String a : "+w1);
        System.out.println("Check Sum of "+s+" : "+ Integer.toBinaryString(q1));
        System.out.println();
        System.out.println();
        System.out.println("Binary of Input String b : "+w2);
        System.out.println("Check Sum of "+r+" : "+ Integer.toBinaryString(q2));
        if((q1 == q2) && q1 != 0 && q2 != 0) {
            System.out.println("\nCheck Sum matching!! No Error");
        }
        else{
            System.out.println("\nCheck Sum not matching!! Error found");
        }
    }
}
```

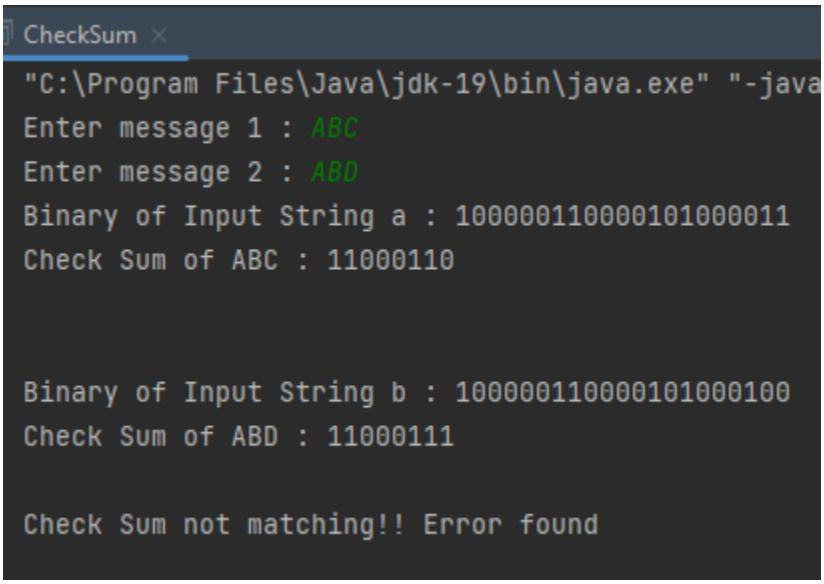
CYCLIC REDUNDANCY CHECK:

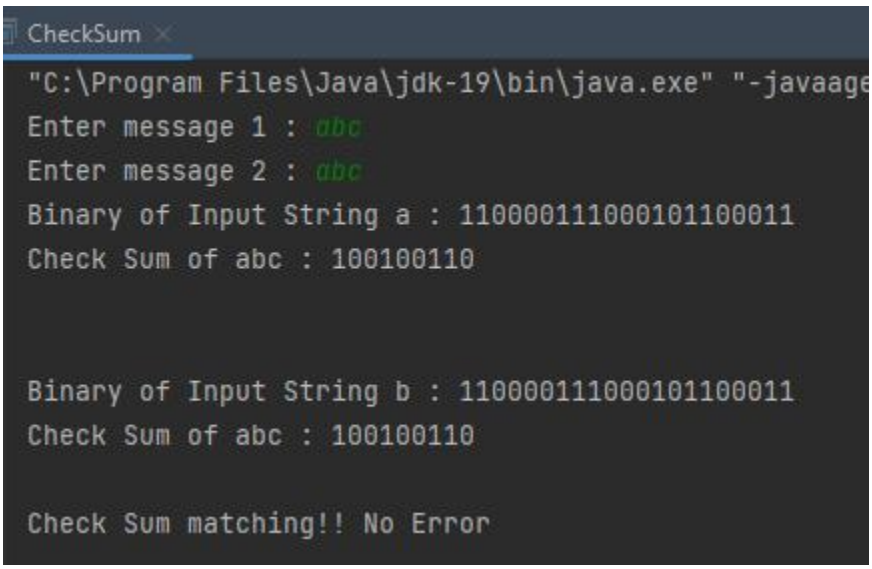
```
public class CRC {
    public static String xor(char q1 , char q2){
        if(q1 == q2) return "0";
        else return "1";
    }
    public static String bin(String divident , String divisor , int flag){
        if(flag == 1) {
            for (int j = 0; j < divisor.length(); j++) {
                divident += 0;
            }
        }
        String res = "";
        String ans = "";
        for(int i = 0 ; i < divisor.length() ; i++) {
            res += divident.charAt(i);
        }
        int i = 0;
        for(int j = divisor.length() ; j < divident.length() ; ) {
            if(res.charAt(i) == '0') {
                while(j < divident.length()&&i<res.length()&&res.charAt(i)== '0') {
                    i++;
                    res += divident.charAt(j);
                    j++;
                }
            }
            if(j >= divident.length()) break;
            for(int w = 0 ; w < divisor.length() ; w++) {
                ans += xor(res.charAt(i) , divisor.charAt(w));
                i++;
            }
            res = ans;
            ans = "";
            i = 0;
        }
        String real_ans = "";
```

```
        for(i = res.length()-divisor.length() ; i < res.length() ; i++){
            real_ans += res.charAt(i);
        }
        return real_ans;
    }
    public static void main(String[] args){
        String s = "1010101010";
        String d = "11001";
        System.out.println("Data : "+s);
        System.out.println("Key : "+d);
        System.out.println("Remainder : " + bin(s,d,1));s += bin(s,d,1);
        System.out.println("\nNew Data after appending : " + s);
        System.out.println("Again doing XOR division!\n");
        System.out.println("Remainder : " + bin(s,d,0) +"\nNo error");
    }
}
```

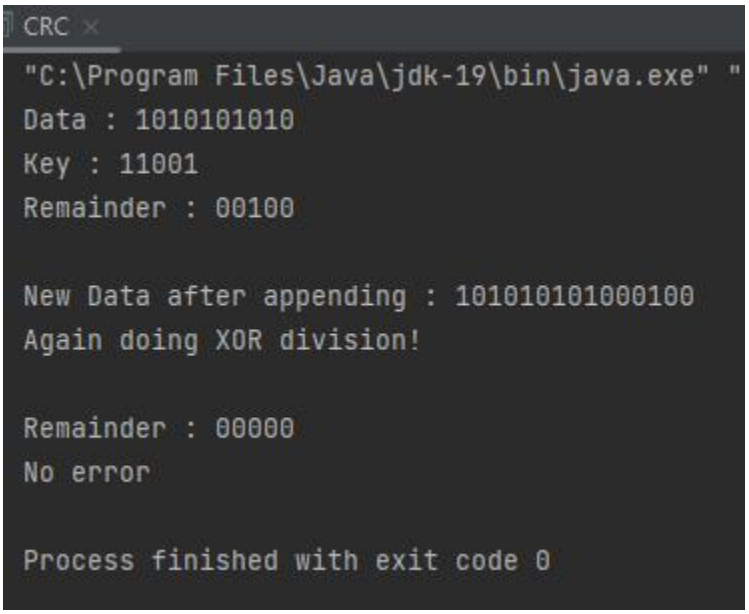
OUTPUT:

CHECK SUM:





CYCLIC REDUNDANCY CHECK:



RESULT:

Thus, Error Detection methods like Check Sum and CRC have been successfully implemented.

EXP NO: 12 DATE:	NS3 SIMULATION

(i) Basic Network construction using NS3

AIM:

Basic network Simulation using NS3 simulator.

CODE:

```
/* -- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -- */
   /*
    * This program is free software; you can redistribute it and/or modify
    * it under the terms of the GNU General Public License version 2 as
    * published by the Free Software Foundation;
    *
    * This program is distributed in the hope that it will be useful,
    * but WITHOUT ANY WARRANTY; without even the implied warranty of
    * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
   * GNU General Public License for more details.
   *
   * You should have received a copy of the GNU General Public License
   * along with this program; if not, write to the Free Software
   * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
   */

  #include "ns3/core-module.h"
  #include "ns3/network-module.h"
  #include "ns3/internet-module.h"
  #include "ns3/point-to-point-module.h"
  #include "ns3/applications-module.h"

  using namespace ns3;

  NS_LOG_COMPONENT_DEFINE ("FirstScriptExample");


  Int main (int argc, char *argv[])
  {
    CommandLine cmd;
    cmd.Parse (argc, argv);

    NS_LOG_UNCOND("Network in NS3 by Sam: ");
    Time::SetResolution (Time::NS);
    LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
    LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);

    NodeContainer nodes;
    nodes.Create (2);

    PointToPointHelper pointToPoint;
    pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
    pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));
```

```
    NetDeviceContainer devices;
    devices = pointToPoint.Install (nodes);

    InternetStackHelper stack;
    stack.Install (nodes);
    Ipv4AddressHelper address;
    address.SetBase ("10.2.3.0", "255.255.255.0");
    Ipv4InterfaceContainer interfaces = address.Assign (devices);
    UdpEchoServerHelper echoServer (9);
    ApplicationContainer serverApps = echoServer.Install (nodes.Get (1));
    serverApps.Start (Seconds (1.0));
    serverApps.Stop (Seconds (10.0));

    UdpEchoClientHelper echoClient (interfaces.GetAddress (1), 9);
    echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
    echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
    echoClient.SetAttribute ("PacketSize", UintegerValue (1024));

    ApplicationContainer clientApps = echoClient.Install (nodes.Get (0));
    clientApps.Start (Seconds (2.0));
    clientApps.Stop (Seconds (10.0));

    Simulator::Run ();
    Simulator::Destroy ();
    return 0;
  }
```

SIMULATION:

```
Scanning dependencies of target scratch_first
[  0%] Building CXX object scratch/CMakeFiles/scratch_first.dir/first.cc.o
[  0%] Linking CXX executable ../../build/scratch/ns3.37-first-default
Network in NS3
At time +2s client sent 1024 bytes to 10.2.3.2 port 9
At time +2.00369s server received 1024 bytes from 10.2.3.1 port 49153
At time +2.00369s server sent 1024 bytes to 10.2.3.1 port 49153
At time +2.00737s client received 1024 bytes from 10.2.3.2 port 9
```

```
Flow ID: 1 Src Addr 10.0.1.1 Dst Addr 10.0.1.2
Tx Packets = 39278
Rx Packets = 3310
lostPackets Packets = 35968
Throughput: 2925.27 Kbps
Flow ID: 2 Src Addr 10.0.2.1 Dst Addr 10.0.2.2
Tx Packets = 39278
Rx Packets = 3310
lostPackets Packets = 35968
Throughput: 2925.27 Kbps
```

RESULT:

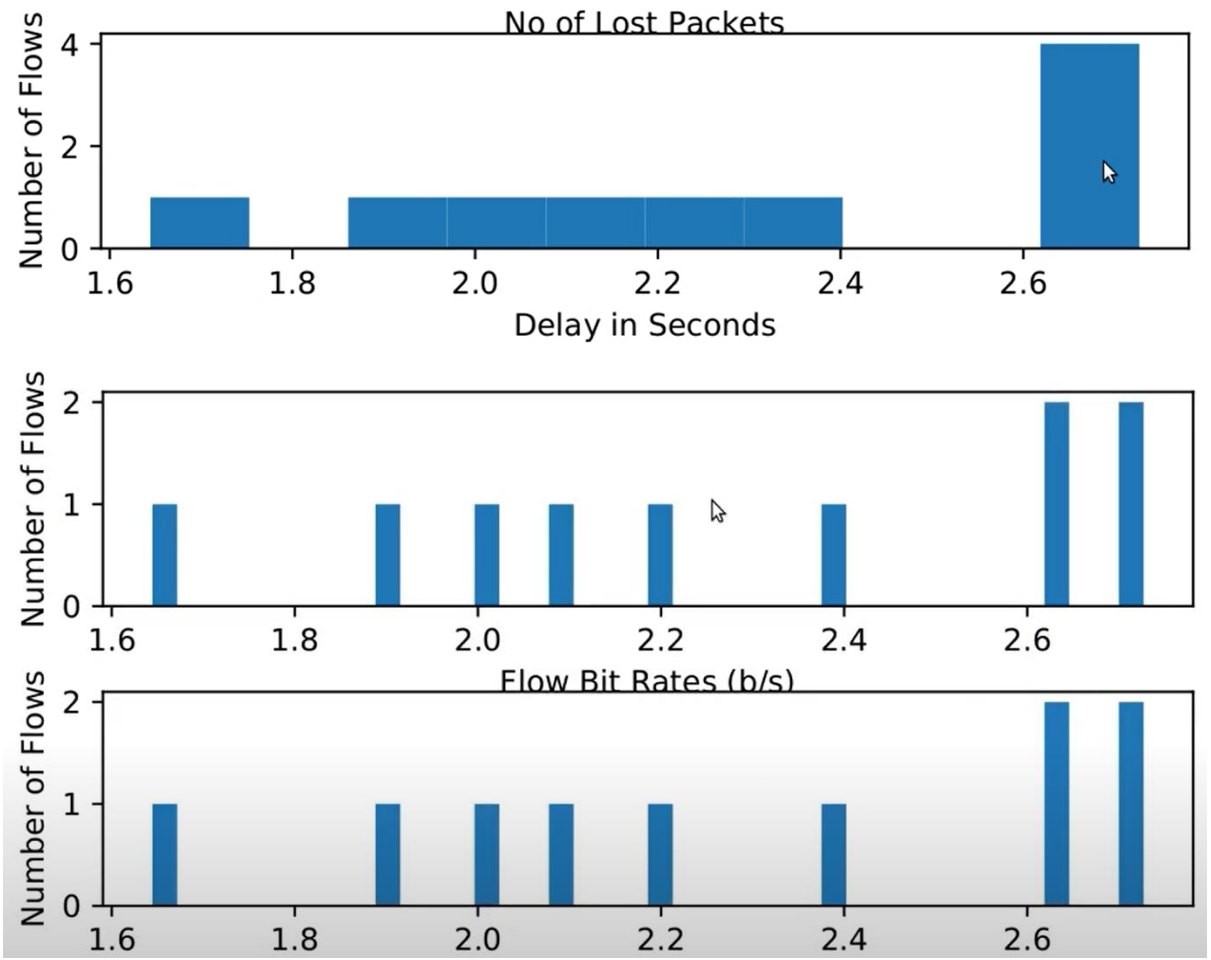
Thus, A basic c using NS3 simulator has been done successfully.

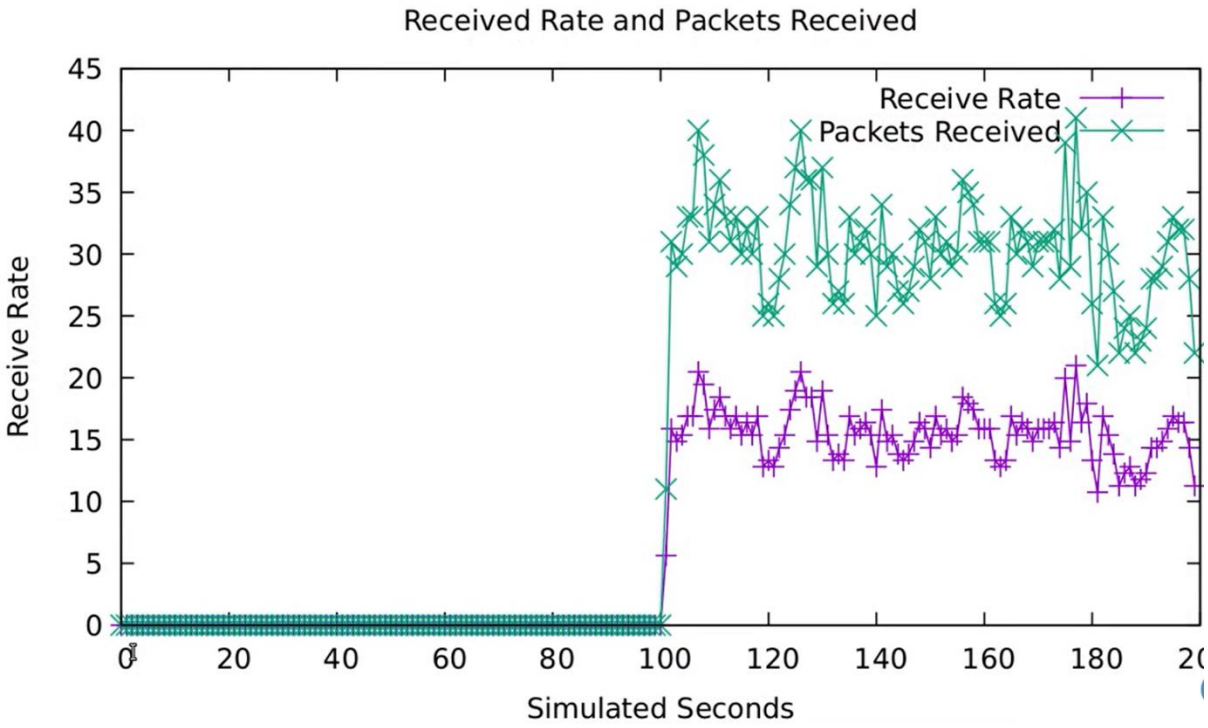
(ii) Performance analysis of different networks topologies and routing protocols using ns3 or Cisco

AIM:

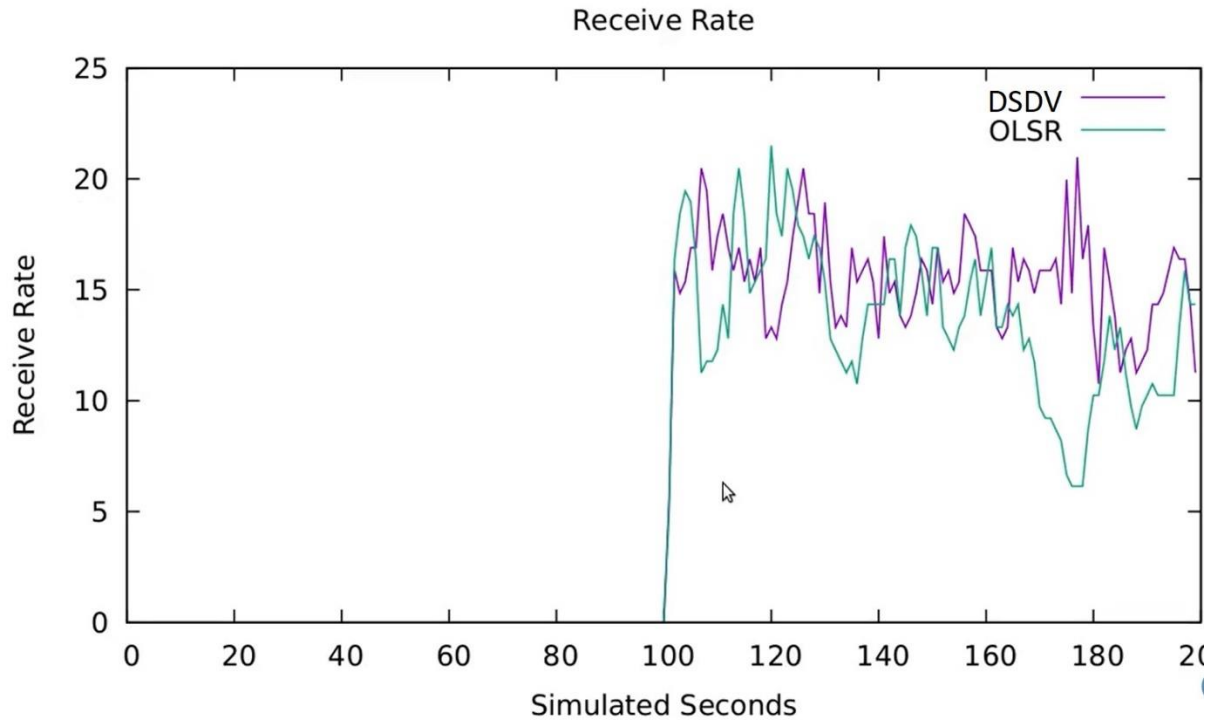
Performance analysis of different networks topologies and routing protocols using ns3.

COMPARITIVE PERFORMANCE ANALYSIS of DSDV and OLSR:



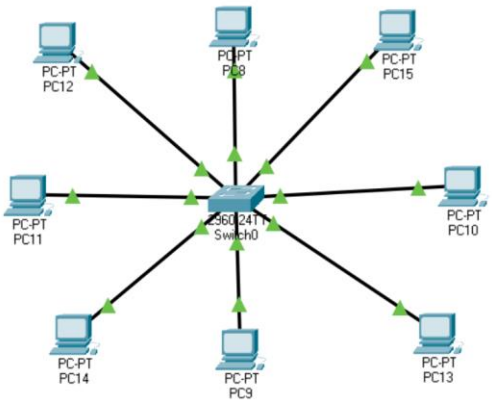


COMPARISON GRAPH:

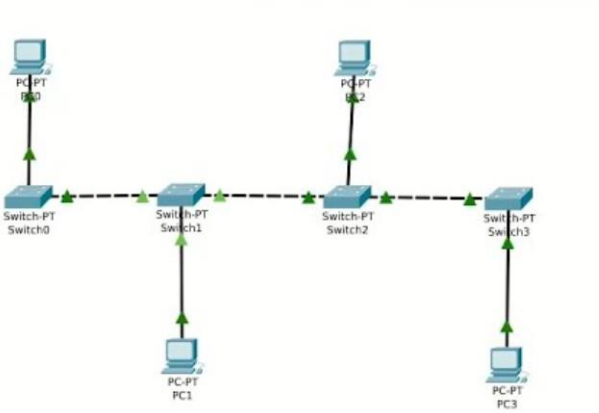


TOPOLOGY SETUP IN CISCO:

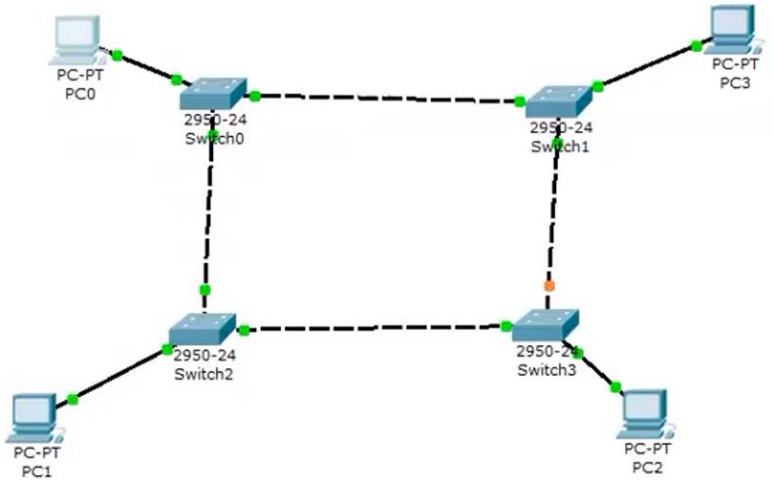
STAR:



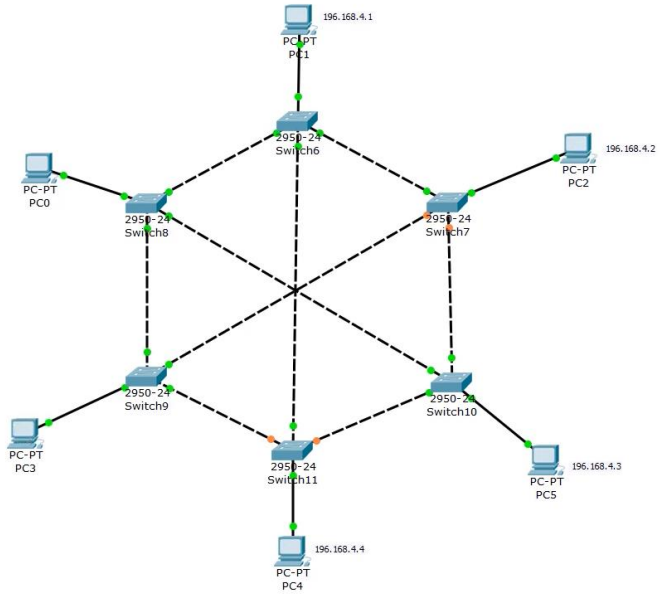
BUS:



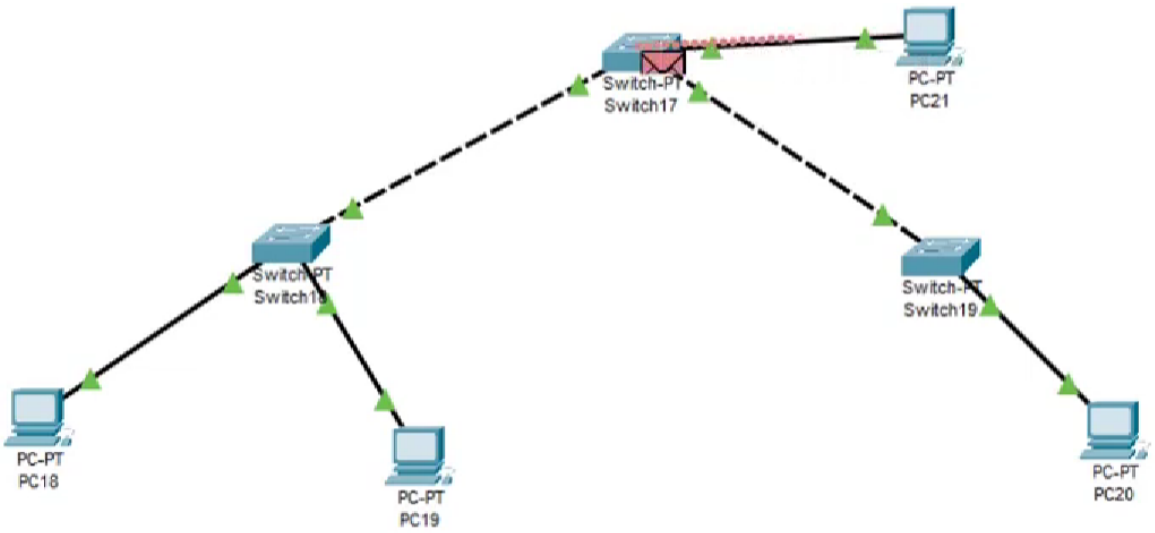
RING:



MESH:



TREE:



RESULT:

Thus, Performance analysis of different routing protocols using ns3 has been done.

(iii) Topology setup using hubs, bridges and switches using simulator.

AIM:

Network Topology setup using hubs, bridges and switches using Cisco.

THEORY:

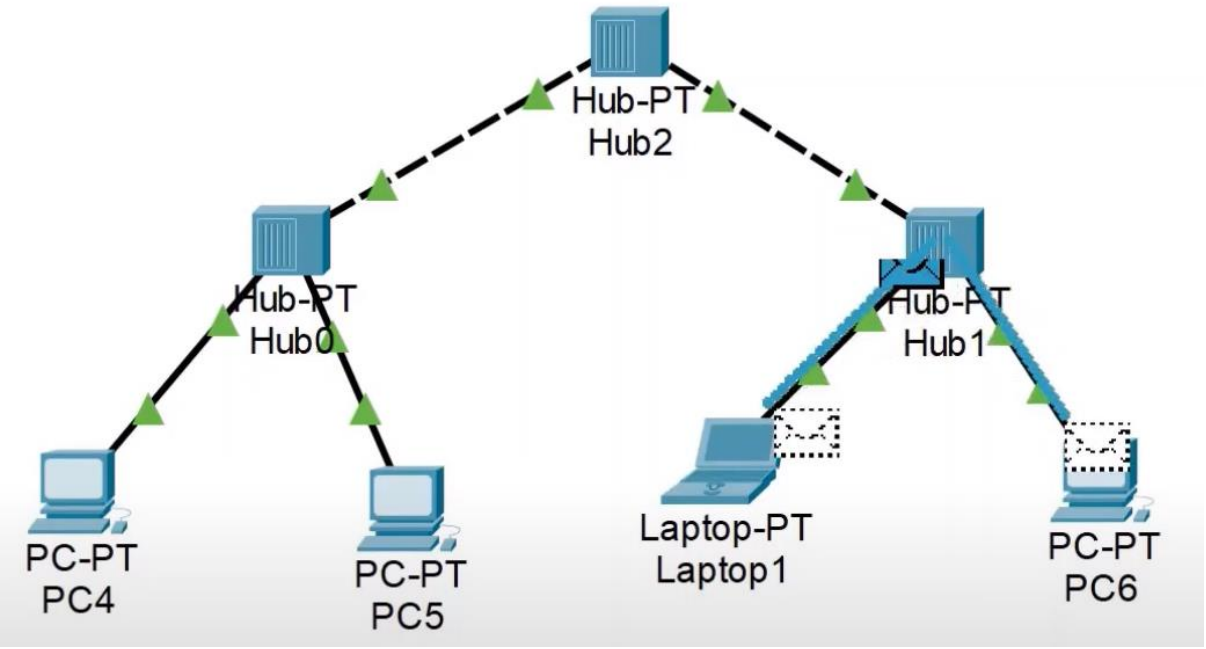
HUBS – A hub is a basically multi-port repeater. A hub connects multiple wires coming from different branches, for example, the connector in star topology which connects different stations. Hubs cannot filter data, so data packets are sent to all connected devices. Also, they do not have the intelligence to find out the best path for data packets which leads to inefficiencies and wastage.

BRIDGES – A bridge operates at the data link layer. A bridge is a repeater, with add on the functionality of filtering content by reading the MAC addresses of the source and destination. It is also used for interconnecting two LANs working on the same protocol. It has a single input and single output port, thus making it a 2 port device.

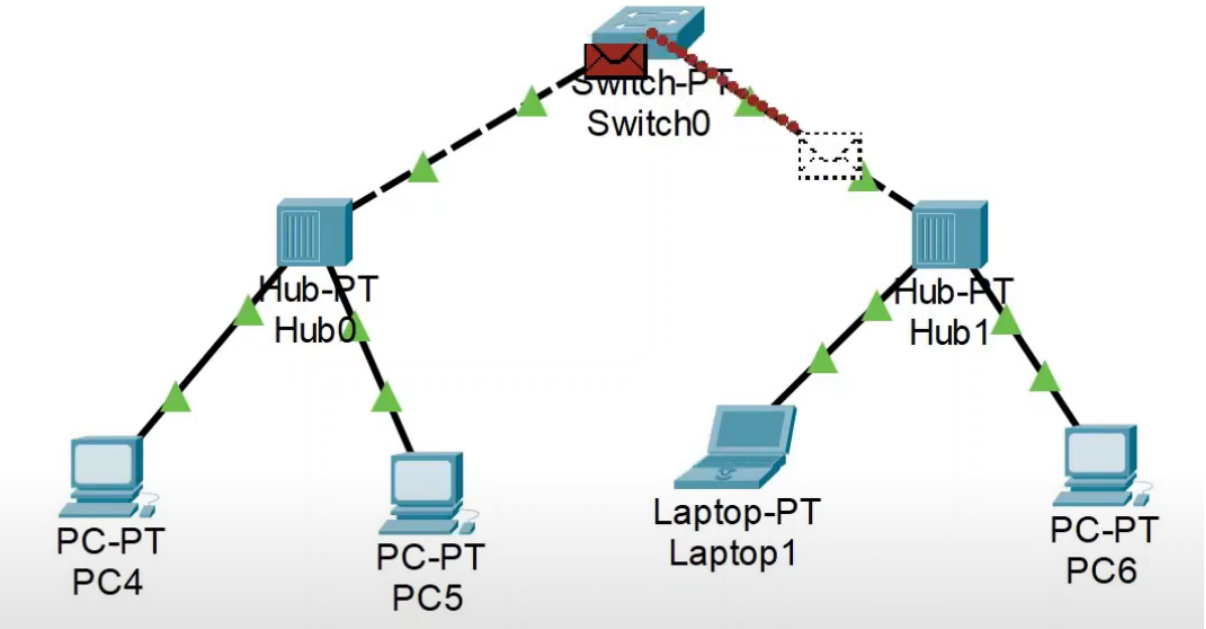
SWITCH – A switch is a multiport bridge with a buffer and a design that can boost its efficiency and performance. A switch is a data link layer device. The switch can perform error checking before forwarding data, which makes it very efficient as it does not forward packets that have errors and forward good packets selectively to the correct port only.

TOPOLOGY SETUP:

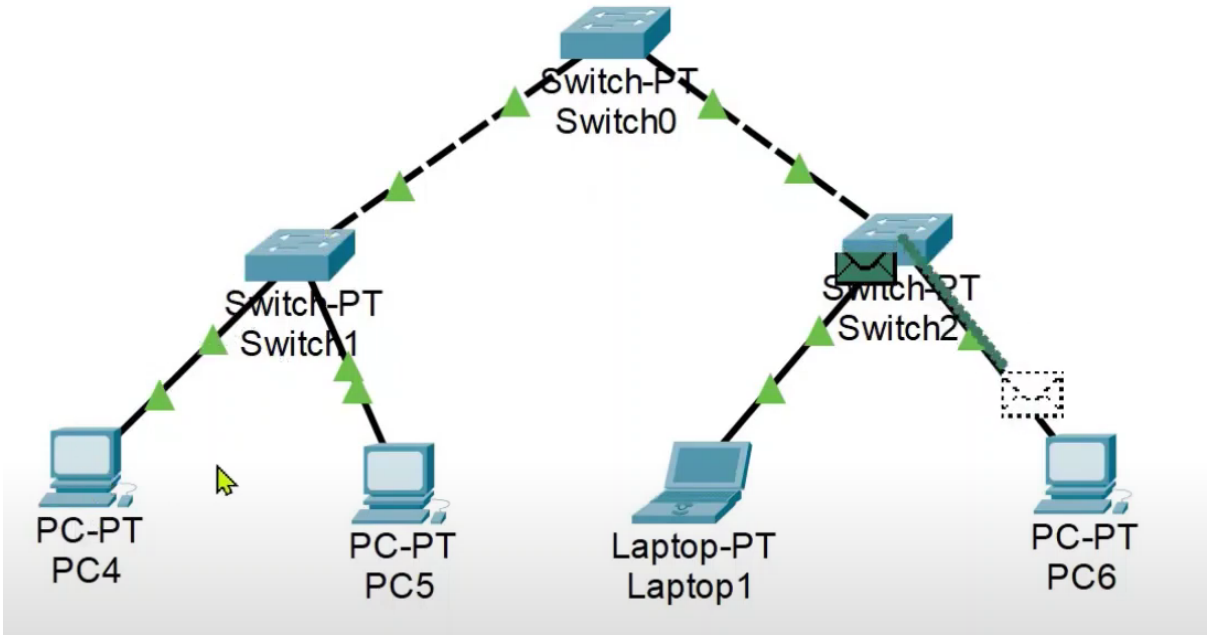
a) **Setup using only Hubs** – Hubs basically broadcast messages to all the devices connected in a network. Point to point transfer in not possible. Here Hub 1 transmits messages to both the devices.



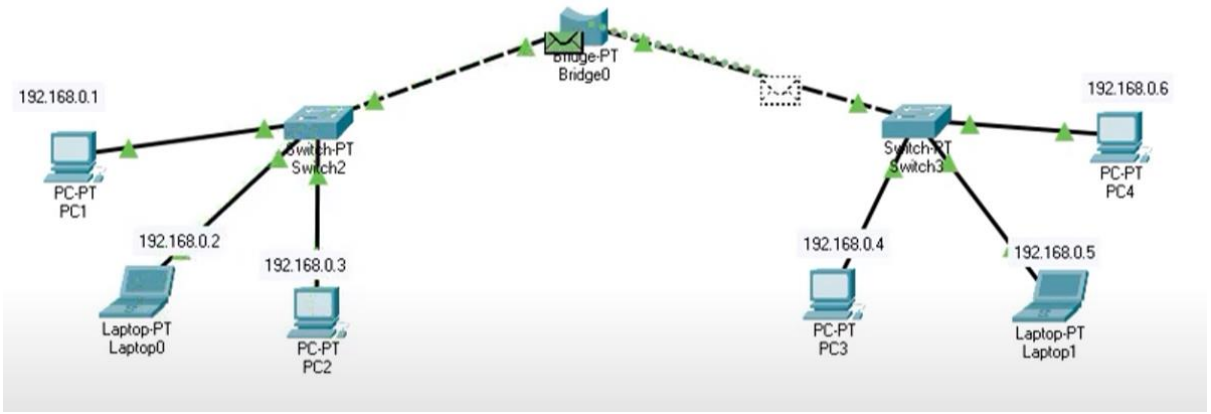
a) **Setup using one Switch and Hubs** – Switches are able to transmit messages to a particular destination as opposed to hubs. Switch0 here transmits a message to hub1 which broadcasts it to both the connected devices.

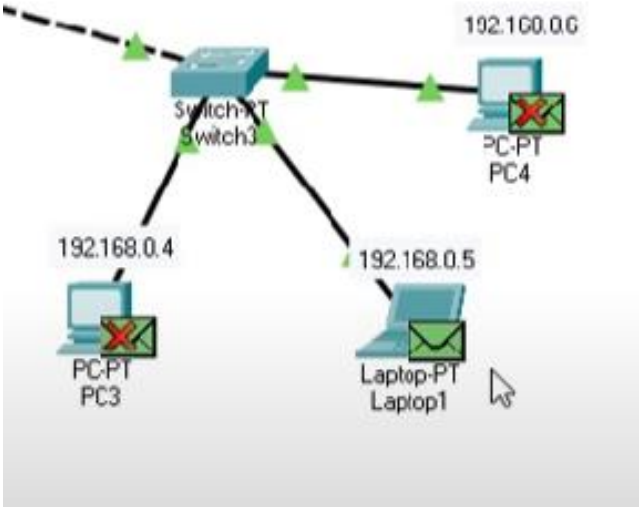


b) **Setup using switches** – Switches are able to transmit messages to a particular destination. Switch0 here transmits a message to Switch2 which transmits into PC6 it to both the connected devices.



c) **Setup using bridges** – Bridges are able to connect two different LAN networks. Bridge0 here transmits a message to Switch3 which transmits into Laptop1. All the other devices which receive the message reject it. So bridge allows efficient transfer of messages.





RESULT:

Thus, Network Topology setup and simulation with hubs, bridges and switches using Cisco has been done

(iv) Cisco packet tracer badge with name

BADGE:

CERTIFICATE: