Faculty of Engineering and Technology

School of Computing

SUBJECT CODE / TITLE: 18CSC302J/ Computer Networks

Prepared by

Dr. T.R. Saravanan , Asssitant Professor , Computational Intelligence
Dr. L. Kavisankar, Assistant Professor , Computing Technologies

SRM Institute of Science and Technology

SRM Nagar, Kattankulathur-603203

INDEX

SLNO	NAME OF EXPERIMENT
1.	Study of necessary header files with respect to socket programming
2.	Study of Basic Functions of Socket Programming
3.	Simple TCP/IP Client Server Communication
4.	UDP Echo Client Server Communication
5.	Concurrent TCP/IP Day-Time Server
6.	Half Duplex Chat Using TCP/IP
7.	Full Duplex Chat Using TCP/IP
8.	Implementation of File Transfer Protocol
9.	Remote Command Execution Using UDP
10.	ARP Implementation Using UDP
11.	Study of IPV6 Addressing & Subnetting
12.	Implementation of Network Address Translation
13.	Implementation of VPN
14.	Communication Using HDLC
15.	Communication Using PPP

Ex.No:3	SIMPLE TCP/IP CLIENT SERVER COMMUNICATION
Date:	

Aim:

Server:

Client:

Source Code:

Server:

```
#include <stdio.h>
#include <netdb.h>
#include <netinet/in.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>

#define MAX 80
#define PORT 2212
#define SA struct sockaddr

void func(int connfd)
{
	char buff[MAX];
	int n;
	for (;;) {
		bzero(buff, MAX);
		read(connfd, buff, sizeof(buff));
		printf("From client: %s\t To client : ", buff);
		bzero(buff, MAX);
		n = 0;
		while ((buff[n++] = getchar()) != '\n')
			;
		write(connfd, buff, sizeof(buff));
		if (strncmp("Exit", buff, 4) == 0) {
			printf("Server Exit...\n");
			break;
		}
	}
}

int main()
{
```

```
    int sockfd, connfd, len;
    struct sockaddr_in servaddr, cli;
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd == -1) {
        printf("Socket creation failed...\n");
        exit(0);
    }
    else
        printf("Socket successfully created..\n");
    bzero(&servaddr, sizeof(servaddr));
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
    servaddr.sin_port = htons(PORT);
    if ((bind(sockfd, (SA*)&servaddr, sizeof(servaddr))) != 0) {
        printf("Socket bind failed...\n");
        exit(0);
    }
    else
        printf("Socket successfully binded..\n");
    if ((listen(sockfd, 5)) != 0) {
        printf("Listen failed...\n");
        exit(0);
    }
    else
        printf("Server listening..\n");
    len = sizeof(cli);
    connfd = accept(sockfd, (SA*)&cli, &len);
    if (connfd < 0) {
        printf("Server accept failed...\n");
        exit(0);
    }
    else
        printf("Server accept the client...\n");
    func(connfd);
    close(sockfd);
}
```

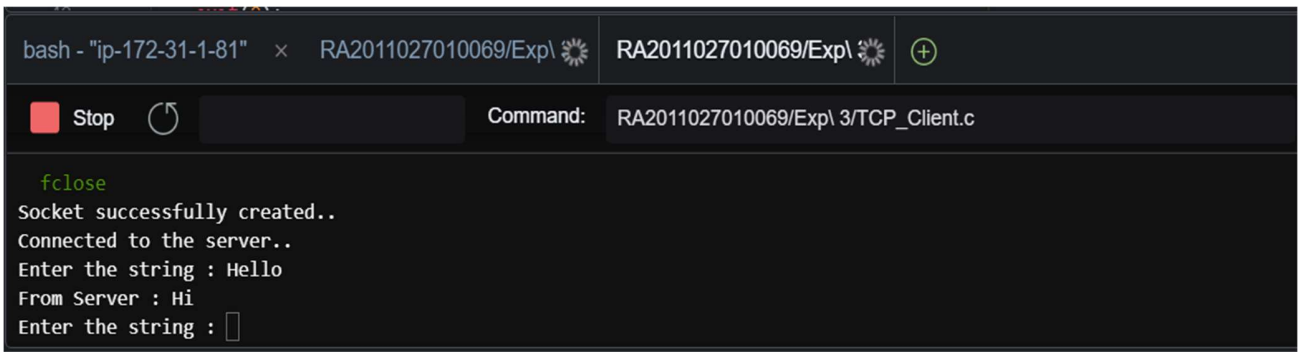
Client:

```
#include <netdb.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>

#define MAX 80
#define PORT 2212
#define SA struct sockaddr

void func(int sockfd)
{
	char buff[MAX];
	int n;
	for (;;) {
		bzero(buff, sizeof(buff));
		printf("Enter the string : ");
		n = 0;
		while ((buff[n++] = getchar()) != '\n')
			;
		write(sockfd, buff, sizeof(buff));
		bzero(buff, sizeof(buff));
		read(sockfd, buff, sizeof(buff));
		printf("From Server : %s", buff);
		if ((strncmp(buff, "exit", 4)) == 0) {
			printf("Client Exit...\n");
			break;
		}
	}
}

int main()
{
	int sockfd, connfd;
	struct sockaddr_in servaddr, cli;
```

```
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd == -1) {
        printf("Socket creation failed...\n");
        exit(0);
    }
    else
        printf("Socket successfully created..\n");
    bzero(&servaddr, sizeof(servaddr));
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
    servaddr.sin_port = htons(PORT);
    if (connect(sockfd, (SA*)&servaddr, sizeof(servaddr)) != 0) {
        printf("Connection with the server failed...\n");
        exit(0);
    }
    else
        printf("Connected to the server..\n");
    func(sockfd);
    close(sockfd);
}
```

OUTPUT:

Client Output:



Server Output:

Result:

Ex.No:3	UDP ECHO CLIENT SERVER COMMUNICATION
Date:	

Aim:

METHODOLOGY:

Server:

Client:

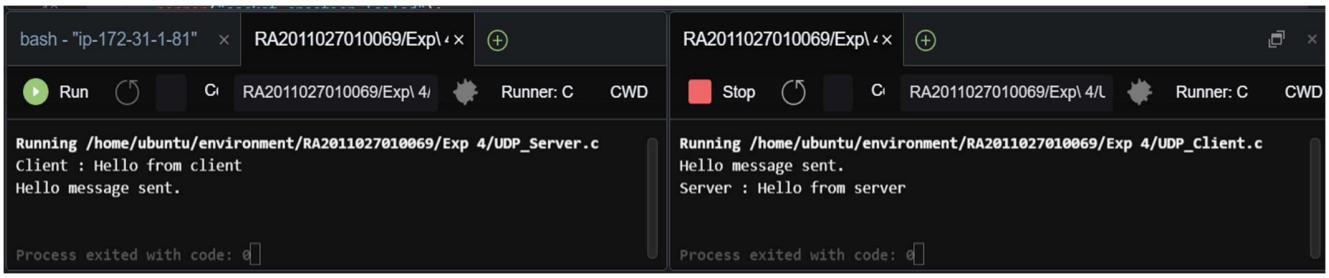
Source Code:
Server:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#define PORT          2002
#define MAXLINE 1024

int main() {
        int sockfd;
        char buffer[MAXLINE];
        char *hello = "Hello from server";
        struct sockaddr_in servaddr, cliaddr;
        if ( (sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0 ) {
                perror("socket creation failed");
                exit(EXIT_FAILURE);
        }
        memset(&servaddr, 0, sizeof(servaddr));
        memset(&cliaddr, 0, sizeof(cliaddr));
        servaddr.sin_family = AF_INET;
        servaddr.sin_addr.s_addr = INADDR_ANY;
        servaddr.sin_port = htons(PORT);

        if ( bind(sockfd, (const struct sockaddr *)&servaddr,
                        sizeof(servaddr)) < 0 )
        {
                perror("bind failed");
                exit(EXIT_FAILURE);
        }
        int len, n;
        len = sizeof(cliaddr);
        n = recvfrom(sockfd, (char *)buffer, MAXLINE,MSG_WAITALL, ( struct sockaddr *)
&cliaddr,&len);
        buffer[n] = '\0';
        printf("Client : %s\n", buffer);
        sendto(sockfd, (const char *)hello, strlen(hello),MSG_CONFIRM, (const struct sockaddr *)
&cliaddr,len);
        printf("Hello message sent.\n");
        return 0;
}
```

Client:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#define PORT    2002
#define MAXLINE 1024

int main()
{
   int sockfd;
   char buffer[MAXLINE];
   char *hello = "Hello from client";
   struct sockaddr_in     servaddr;
   if ( (sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0 ) {
      perror("socket creation failed");
      exit(EXIT_FAILURE);
   }
   memset(&servaddr, 0, sizeof(servaddr));
   servaddr.sin_family = AF_INET;
   servaddr.sin_port = htons(PORT);
   servaddr.sin_addr.s_addr = INADDR_ANY;
   int n, len;
   sendto(sockfd, (const char *)hello, strlen(hello), MSG_CONFIRM, (const struct sockaddr *)
&servaddr, sizeof(servaddr));
   printf("Hello message sent.\n");
   n = recvfrom(sockfd, (char *)buffer, MAXLINE, MSG_WAITALL, (struct sockaddr *) &servaddr,
&len);
   buffer[n] = '\0';
   printf("Server : %s\n", buffer);
   close(sockfd);
   return 0;
}
```

Output:



Result:

Ex.No:5	CONCURRENT TCP/IP DAY-TIME SERVER
Date:	

Aim:

METHODOLOGY:
Server :

Client :

Source Code:

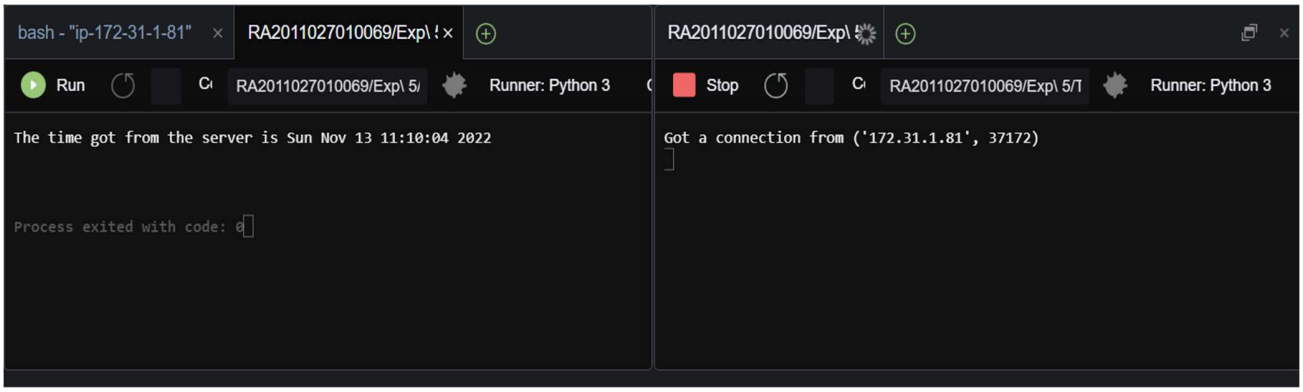
Server:

```
import socket
import time

serversocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
host = socket.gethostname()
port = 9099
serversocket.bind((host, port))
serversocket.listen(5)

while True:
   clientsocket,addr = serversocket.accept()
   print("Got a connection from %s" % str(addr))
   currentTime = time.ctime(time.time()) + "\r\n"
   clientsocket.send(currentTime.encode('ascii'))
   clientsocket.close()
```

Client:

```
import socket

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
host = socket.gethostname()
port = 9099
s.connect((host, port))
tm = s.recv(1024)

s.close()
print("The time got from the server is %s" % tm.decode('ascii'))
```

Output:



Result :

Ex.No:6	HALF DUPLEX CHAT USING TCP/IP
Date:	

Aim :

Methodlogy:
Server:

Client:

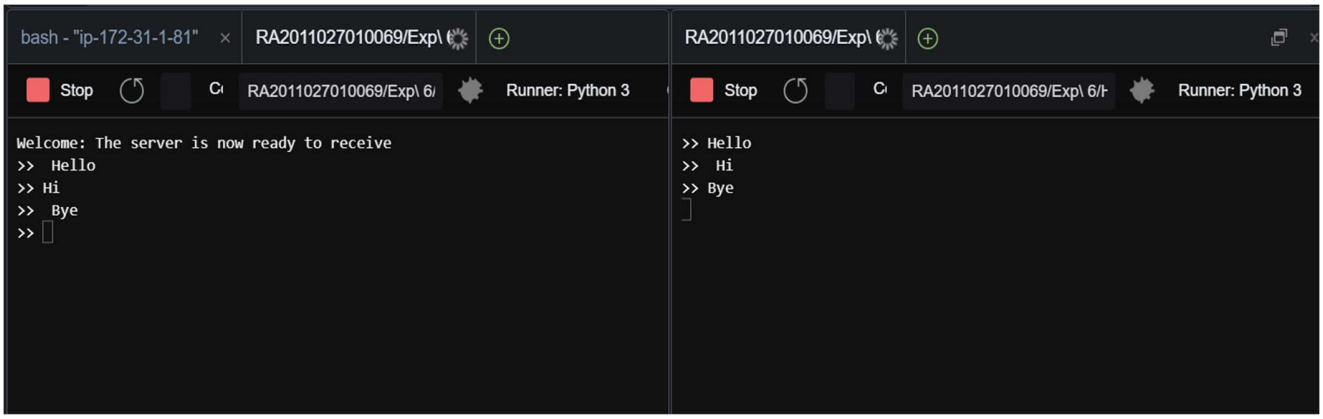
Source Code:
Server:

```
from socket import *
server_port = 6500
server_socket = socket(AF_INET,SOCK_STREAM)
server_socket.bind(('',server_port))
server_socket.listen(1)
print ("Welcome: The server is now ready to receive")
connection_socket, address = server_socket.accept()
while True:
  sentence = connection_socket.recv(2048).decode()
  print('>> ',sentence)
  message = input(">> ")
  connection_socket.send(message.encode())
  if(message == 'Bye'):
    connectionSocket.close()
```

Client:

```
from socket import *
server_name = 'localhost'
server_port = 6500
client_socket = socket(AF_INET, SOCK_STREAM)
client_socket.connect((server_name,server_port))

while True:
  sentence = input(">> ")
  client_socket.send(sentence.encode())
  message = client_socket.recv(2048)
  print (">> ", message.decode())
  if(sentence == 'Bye'):
    client_socket.close()
```

Output:



Result :

Ex.No:7	FULL DUPLEX CHAT USING TCP/IP
Date:	

Aim:

Methodology:
Server:

Client:

Source Code:
Server:

```
#include <stdio.h>
#include <netdb.h>
#include <netinet/in.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#define MAX 80
#define PORT 8080
#define SA struct sockaddr

void func(int connfd)
{
char buff[MAX];
int n;
for (;;) {
bzero(buff, MAX);

read(connfd, buff, sizeof(buff));
printf("From client: %s\t To client : ", buff);
bzero(buff, MAX);
n = 0;
while ((buff[n++] = getchar()) != '\n')
;

write(connfd, buff, sizeof(buff));

if (strncmp("exit", buff, 4) == 0) {
printf("Server Exit...\n");
break;
}
}
}

int main()
{
int sockfd, connfd, len;
struct sockaddr_in servaddr, cli;
sockfd = socket(AF_INET, SOCK_STREAM, 0);
```

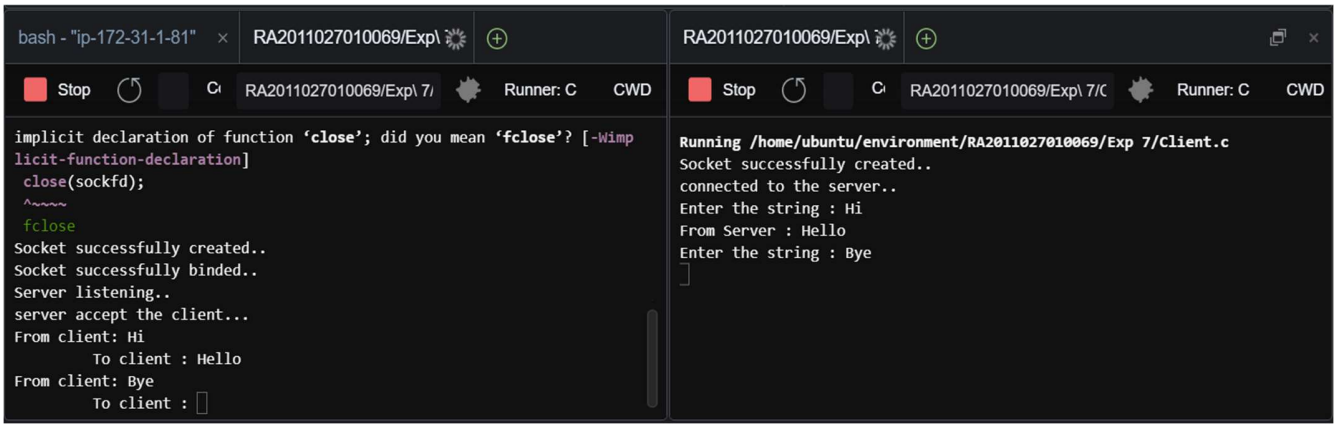
```
if (sockfd == -1) {
printf("socket creation failed...\n");
exit(0);
}
else
printf("Socket successfully created..\n");
bzero(&servaddr, sizeof(servaddr));

servaddr.sin_family = AF_INET;
servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
servaddr.sin_port = htons(PORT);

if ((bind(sockfd, (SA*)&servaddr, sizeof(servaddr))) != 0) {
printf("socket bind failed...\n");
exit(0);
}
else
printf("Socket successfully binded..\n");

if ((listen(sockfd, 5)) != 0) {
printf("Listen failed...\n");
exit(0);
}
else

printf("Server listening..\n");
len = sizeof(cli);

connfd = accept(sockfd, (SA*)&cli, &len);
if (connfd < 0) {
printf("server accept failed...\n");
exit(0);
}
else
printf("server accept the client...\n");

func(connfd);

close(sockfd);
}
```

Client:

```
#include <arpa/inet.h>
#include <netdb.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <strings.h>
#include <sys/socket.h>
#include <unistd.h>
#define MAX 80
#define PORT 8080
#define SA struct sockaddr

void func(int sockfd)
{
        char buff[MAX];
        int n;
        for (;;) {
        bzero(buff, sizeof(buff));
        printf("Enter the string : ");
        n = 0;
        while ((buff[n++] = getchar()) != '\n')
        ;
        write(sockfd, buff, sizeof(buff));
        bzero(buff, sizeof(buff));
        read(sockfd, buff, sizeof(buff));
        printf("From Server : %s", buff);
        if ((strncmp(buff, "exit", 4)) == 0) {
                printf("Client Exit...\n");
                break;
        }
        }
}

int main()
{
        int sockfd, connfd;
        struct sockaddr_in servaddr, cli;
        sockfd = socket(AF_INET, SOCK_STREAM, 0);
        if (sockfd == -1) {
        printf("socket creation failed...\n");
        exit(0);
```

```
        }
        else
        printf("Socket successfully created..\n");
        bzero(&servaddr, sizeof(servaddr));

        servaddr.sin_family = AF_INET;
        servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
        servaddr.sin_port = htons(PORT);

        if (connect(sockfd, (SA*)&servaddr, sizeof(servaddr))
        != 0) {
        printf("connection with the server failed...\n");
        exit(0);
        }
        else
        printf("connected to the server..\n");

        func(sockfd);

        close(sockfd);
}
```

Output:



Result:

Ex.No:8	IMPLEMENTATION OF FILE TRANSFER PROTOCOL
Date:	

Aim :

METHODOLOGY:

Server :

Client :

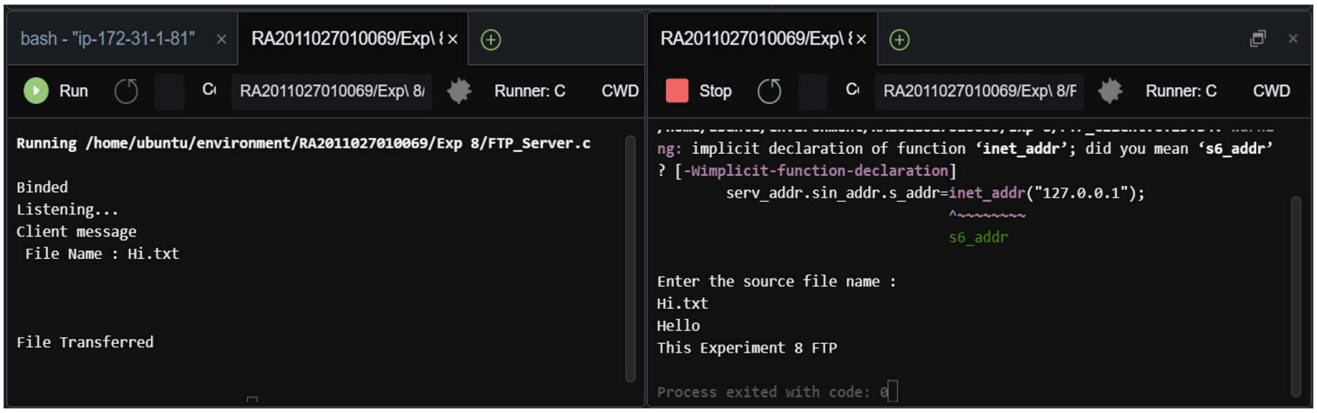
Source Code:
Server:

```
#include<stdio.h>
#include<arpa/inet.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<netdb.h>
#include<stdlib.h>
#include<string.h>
#include<unistd.h>
#define SERV_TCP_PORT 5035
#define MAX 60
int i, j, tem;
char buff[4096], t;
FILE *f1;
int main(int afg, char *argv)
{
    int sockfd, newsockfd, clength;
    struct sockaddr_in serv_addr,cli_addr;
    char t[MAX], str[MAX];
    strcpy(t,"exit");
    sockfd=socket(AF_INET, SOCK_STREAM,0);
    serv_addr.sin_family=AF_INET;
    serv_addr.sin_addr.s_addr=INADDR_ANY;
    serv_addr.sin_port=htons(SERV_TCP_PORT);
    printf("\nBinded");
    bind(sockfd,(struct sockaddr*)&serv_addr, sizeof(serv_addr));
    printf("\nListening...");
    listen(sockfd, 5);
    clength=sizeof(cli_addr);
    newsockfd=accept(sockfd,(struct sockaddr*) &cli_addr,&clength);
    close(sockfd);
    read(newsockfd, &str, MAX);
    printf("\nClient message\n File Name : %s\n", str);
    f1=fopen(str, "r");
    while(fgets(buff, 4096, f1)!=NULL)
    {
       write(newsockfd, buff,MAX);
       printf("\n");
    }
    fclose(f1);
    printf("\nFile Transferred\n");
    return 0;
}
```

Client:

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<netdb.h>
#include<stdlib.h>
#include<string.h>
#include<unistd.h>
#define SERV_TCP_PORT 5035
#define MAX 60
int main(int arg,char*argv[])
{
    int sockfd,n;
    struct sockaddr_in serv_addr;
    struct hostent*server;
    char send[MAX],recvline[MAX],s[MAX],name[MAX];
    sockfd=socket(AF_INET,SOCK_STREAM,0);
    serv_addr.sin_family=AF_INET;
    serv_addr.sin_addr.s_addr=inet_addr("127.0.0.1");
    serv_addr.sin_port=htons(SERV_TCP_PORT);
    connect(sockfd,(struct sockaddr*)&serv_addr,sizeof(serv_addr));
    printf("\nEnter the source file name : \n");
    scanf("%s",send);
    write(sockfd,send,MAX);
    while((n=read(sockfd,recvline,MAX))!=0)
    {
      printf("%s",recvline);
    }
    close(sockfd);
    return 0;
}
```

Output:



Result:

Ex.No:9	REMOTE COMMAND EXECUTION USING UDP
Date:	

Aim:

METHODOLOGY:

Server:

.

Client:

Source Code:

Server:

```
#include<sys/types.h>
#include<sys/socket.h>
#include<stdio.h>
#include<netdb.h>
#include<netinet/in.h>
#include<string.h>
#include<sys/stat.h>
#include<arpa/inet.h>
#include<unistd.h>
int main(int argc,char* argv[])
{
 int sd,size;
 char buff[1024],file[10000];
 struct sockaddr_in cliaddr,servaddr;
 FILE *fp;
 struct stat x;
 socklen_t clilen;
 clilen=sizeof(cliaddr);
 bzero(&servaddr,sizeof(servaddr));
 servaddr.sin_family=AF_INET;
 servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
 servaddr.sin_port=htons(9976);
 sd=socket(AF_INET,SOCK_DGRAM,0);
 if(sd<0)
 {
  printf("Socket CReation Error");
 }
 bind(sd,(struct sockaddr *)&servaddr,sizeof(servaddr));
 while(1)
 {
  bzero(buff,sizeof(buff));
  recvfrom(sd,buff,sizeof(buff),0,(struct sockaddr *)&cliaddr,&clilen);
  strcat(buff,">Hi.text");
  system(buff);
  fp=fopen("Hi.text","r");
  stat("Hi.text",&x);
  size=x.st_size;
  fread(file,size,1,fp);
  sendto(sd,file,sizeof(file),0,(struct sockaddr *)&cliaddr,sizeof(cliaddr));
  printf("Data Sent to UDPCLIENT %s",buff);
 }
 close(sd);
return 0;
}
```

Client:

```
#include<sys/types.h>
#include<sys/socket.h>
#include<stdio.h>
#include<unistd.h>
#include<netdb.h>
#include<netinet/in.h>
#include<string.h>
#include<arpa/inet.h>
#include<sys/stat.h>
int main(int argc,char* argv[])
{
 int sd;
 char buff[1024],file[10000];
 struct sockaddr_in cliaddr,servaddr;
 struct hostent *h;
 socklen_t servlen;
 servlen=sizeof(servaddr);
 h=gethostbyname(argv[1]);
 bzero(&servaddr,sizeof(servaddr));
 servaddr.sin_family=h->h_addrtype;
 memcpy((char *)&servaddr.sin_addr,h->h_addr_list[0],h->h_length);
 servaddr.sin_port=htons(9976);
 sd=socket(AF_INET,SOCK_DGRAM,0);
 if(sd<0)
 {
  printf("Socket CReation Error");
 }
 bind(sd,(struct sockaddr *)&servaddr,sizeof(servaddr));
 while(1)
 {
  printf("\nEnter the command to be executed");
  fgets(buff,1024,stdin);
  sendto(sd,buff,strlen(buff)+1,0,(struct sockaddr *)&servaddr,sizeof(servaddr));
  printf("\nData Sent");
  recvfrom(sd,file,strlen(file)+1,0,(struct sockaddr *)&servaddr,&servlen);
  printf("Recieved From UDPSERVER %s",file);
 }
 return 0;
}
```

Output:

Server:

```
RA1911026010114:~/environment/RA1911026010114/CN LAB 9/SERVER (master) $ cc server.c
RA1911026010114:~/environment/RA1911026010114/CN LAB 9/SERVER (master) $ ./a.out
Command Executed ... vi text.txt
a.out  server.c  text.txt
 Command Executed ... ls
Hi my name is Amulya
Nice to meet you.
How are you?
Have a great day.
Bye bye

 Command Executed ... cat text.txt
```

Client:

```
RA1911026010114:~/environment/RA1911026010114/CN LAB 9/CLIENT (master) $ cc client.c
RA1911026010114:~/environment/RA1911026010114/CN LAB 9/CLIENT (master) $ ./a.out

COMMAND FOR EXECUTION ... vi text.txt

Data Sent !UDP SERVER : Command Successfully executed !
COMMAND FOR EXECUTION ... ls

Data Sent !UDP SERVER : Command Successfully executed !
COMMAND FOR EXECUTION ... cat text.txt

Data Sent !UDP SERVER : Command Successfully executed !
COMMAND FOR EXECUTION ...
```

Result:

Ex.No:10	ARP IMPLEMENTATION USING UDP
Date:	

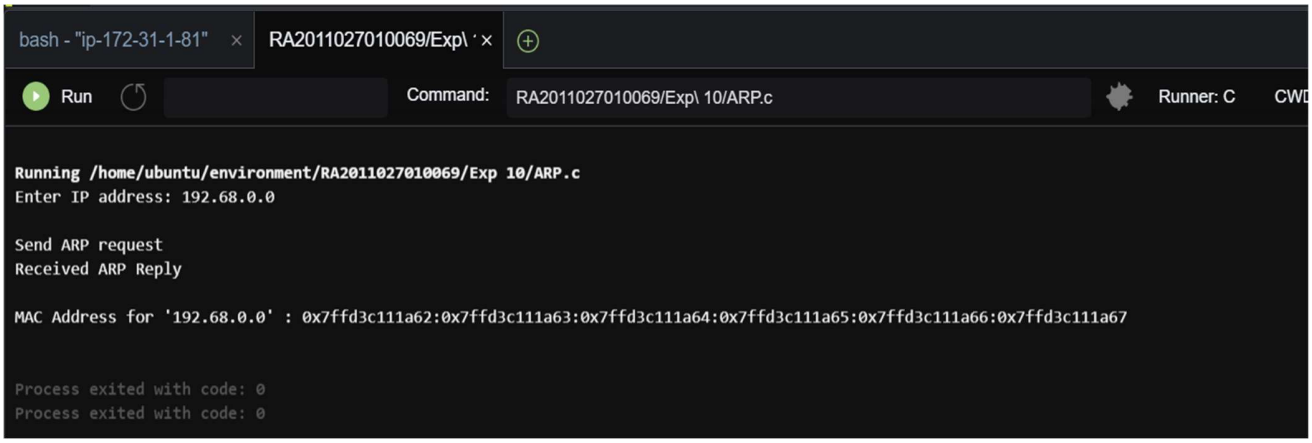
Aim :

METHODOLOGY:

Source Code:

```
#include<sys/types.h>
#include<sys/socket.h>
#include<net/if_arp.h>
#include<sys/ioctl.h>
#include<stdio.h>
#include<string.h>
#include<unistd.h>
#include<math.h>
#include<complex.h>
#include<arpa/inet.h>
#include<netinet/in.h>
#include<netinet/if_ether.h>
#include<net/ethernet.h>
#include<stdlib.h>
int main()
{
struct sockaddr_in sin={0};
struct arpreq myarp={{0}};
unsigned char *ptr;
int sd;
sin.sin_family=AF_INET;
printf("Enter IP address: ");
char ip[20];
scanf("%s", ip);

if(inet_pton(AF_INET,ip,&sin.sin_addr)==0)
{
printf("IP address Entered '%s' is not valid \n",ip);
exit(0);
}
memcpy(&myarp.arp_pa,&sin,sizeof(myarp.arp_pa));
strcpy(myarp.arp_dev,"echo");
sd=socket(AF_INET,SOCK_DGRAM,0);
printf("\nSend ARP request\n");
if(ioctl(sd,SIOCGARP,&myarp)==1)
{
printf("No Entry in ARP cache for '%s'\n",ip);
exit(0);
}
ptr=&myarp.arp_pa.sa_data[0];
printf("Received ARP Reply\n");
printf("\nMAC Address for '%s' : ",ip);
printf("%p:%p:%p:%p:%p:%p\n",ptr,(ptr+1),(ptr+2),(ptr+3),(ptr+4),(ptr+5));
return 0;
}
```

Output:



Result:

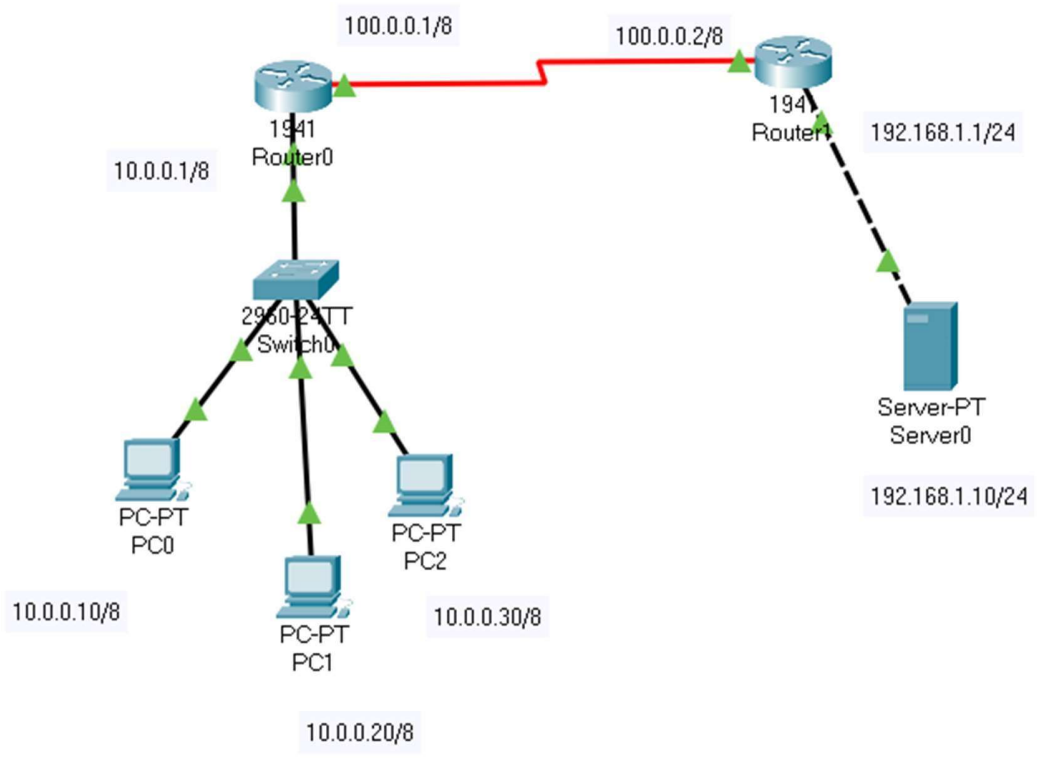
Ex.No:11	STUDY OF IPV6 ADDRESSING & SUBNETTING
Date:	

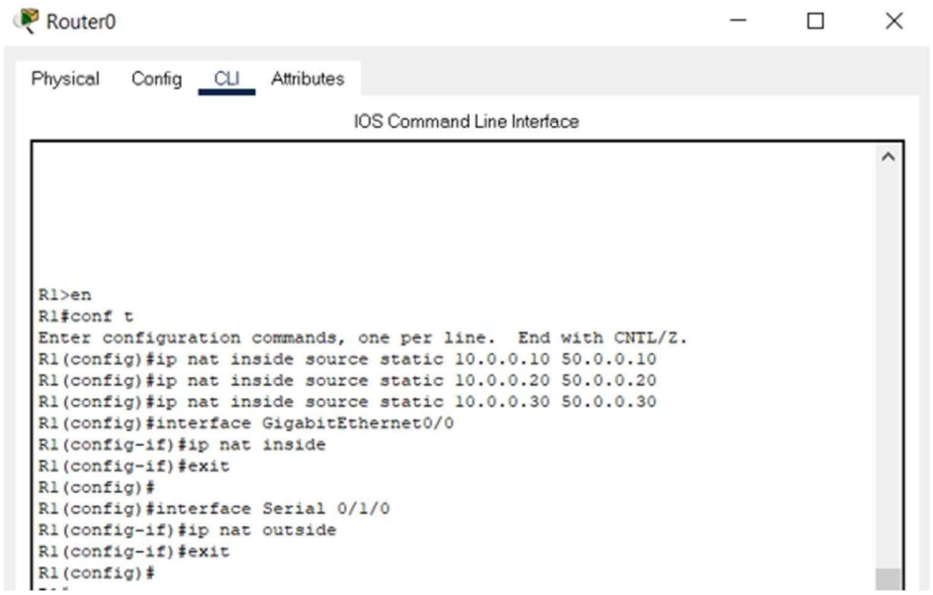
AIM:

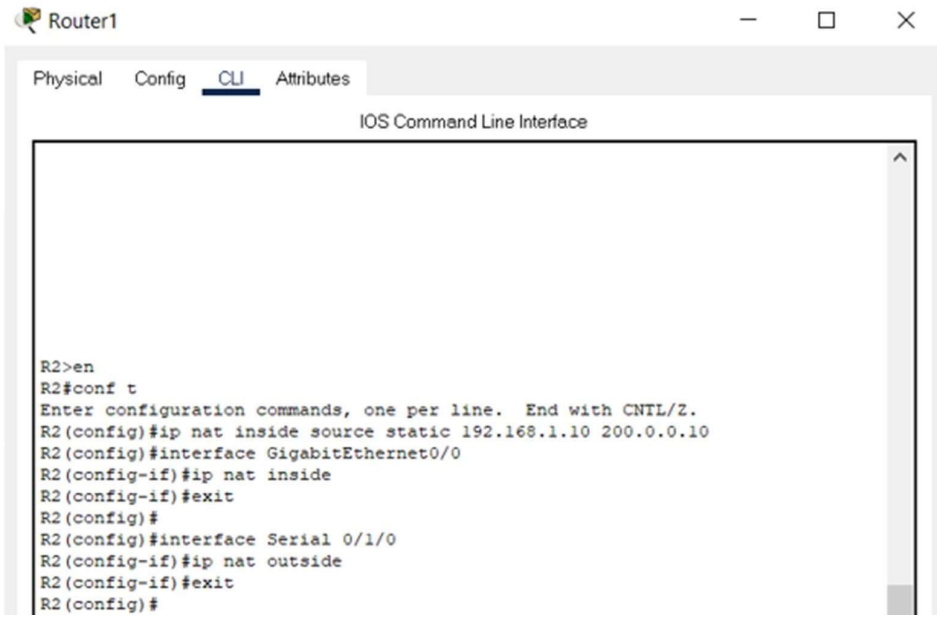
Ex.No:12	IMPLEMENTATION OF NETWORK ADDRESS TRANSLATION
Date:	

Aim:

Procedure:



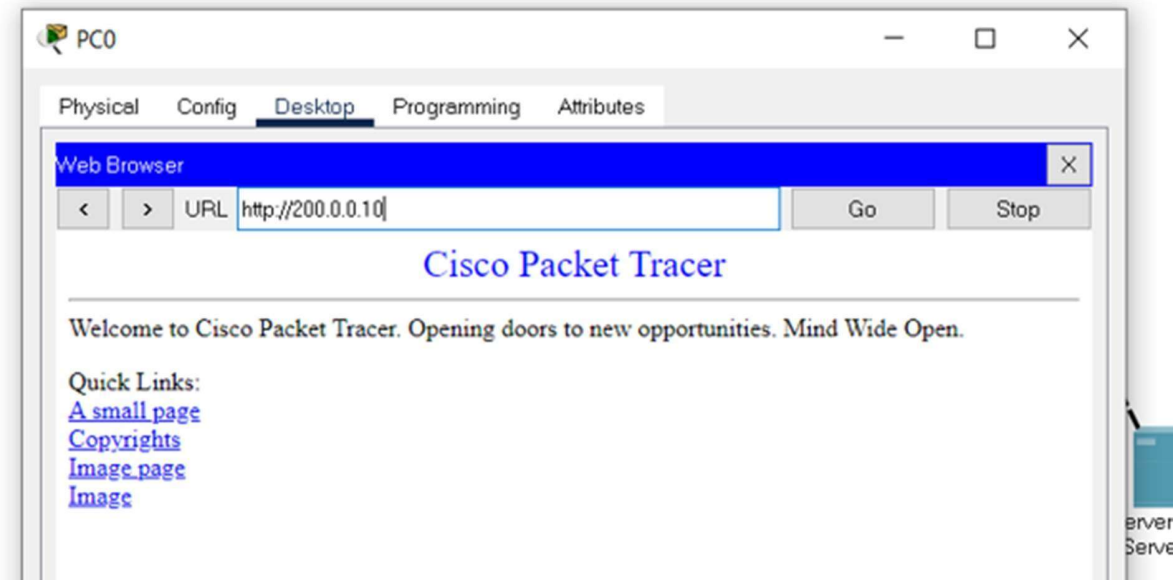

Router0
Physical
Config
CLI
Attributes
IOS Command Line Interface
R1>en
R1#conf t
Enter configuration commands, one per line. End with CNTL/Z.
R1(config)#ip nat inside source static 10.0.0.10 50.0.0.10
R1(config)#ip nat inside source static 10.0.0.20 50.0.0.20
R1(config)#ip nat inside source static 10.0.0.30 50.0.0.30
R1(config)#interface GigabitEthernet0/0
R1(config-if)#ip nat inside
R1(config-if)#exit
R1(config)#
R1(config)#interface Serial 0/1/0
R1(config-if)#ip nat outside
R1(config-if)#exit
R1(config)#

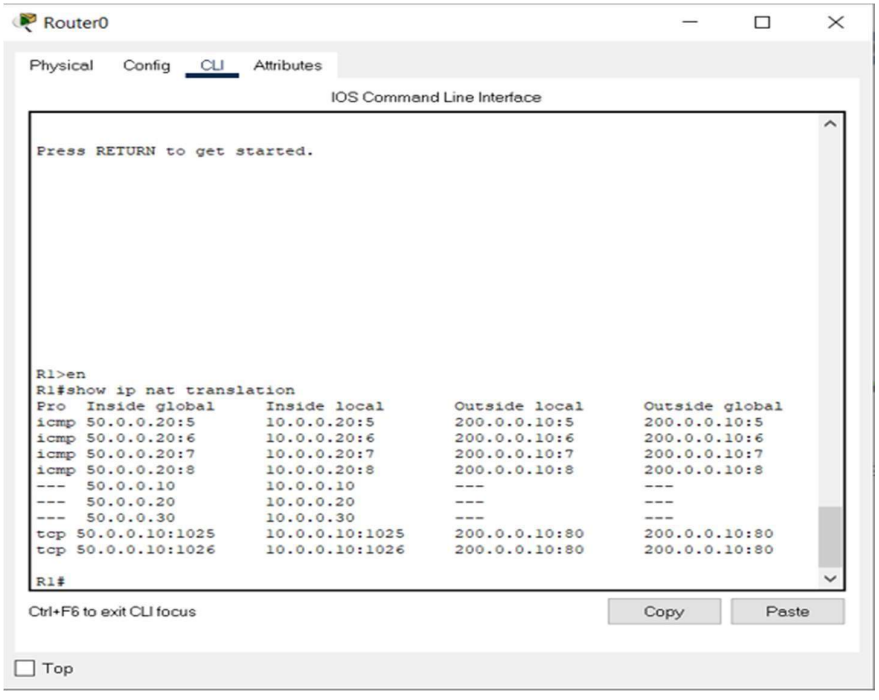

Router1
Physical
Config
CLI
Attributes
IOS Command Line Interface
R2>en
R2#conf t
Enter configuration commands, one per line. End with CNTL/Z.
R2(config)#ip nat inside source static 192.168.1.10 200.0.0.10
R2(config)#interface GigabitEthernet0/0
R2(config-if)#ip nat inside
R2(config-if)#exit
R2(config)#
R2(config)#interface Serial 0/1/0
R2(config-if)#ip nat outside
R2(config-if)#exit
R2(config)#

```
R1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#ip route 200.0.0.0 255.255.255.0 100.0.0.2
R1(config)#no shutdown
                ^
```

```
R2(config)#
R2(config)#ip route 50.0.0.0 255.0.0.0 100.0.0.1
```


PC0
Physical
Config
Desktop
Programming
Attributes
Command Prompt
Reply from 10.0.0.1: Destination host unreachable.
Ping statistics for 192.168.1.10:
Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
C:\>ipconfig
FastEthernet0 Connection:(default port)
Connection-specific DNS Suffix..:
Link-local IPv6 Address.....: FE80::260:47FF:FE93:623B
IPv6 Address.....: ::
IPv4 Address.....: 10.0.0.10
Subnet Mask.....: 255.0.0.0
Default Gateway.....: ::
10.0.0.1
Bluetooth Connection:
Connection-specific DNS Suffix..:
Link-local IPv6 Address.....: ::
IPv6 Address.....: ::
IPv4 Address.....: 0.0.0.0
Subnet Mask.....: 0.0.0.0
Default Gateway.....: ::
0.0.0.0
C:\>ping 200.0.0.10
Pinging 200.0.0.10 with 32 bytes of data:
Reply from 200.0.0.10: bytes=32 time=10ms TTL=126
Reply from 200.0.0.10: bytes=32 time=1ms TTL=126
Reply from 200.0.0.10: bytes=32 time=2ms TTL=126
Reply from 200.0.0.10: bytes=32 time=8ms TTL=126
Ping statistics for 200.0.0.10:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 1ms, Maximum = 10ms, Average = 5ms
C:\>ping 192.168.1.10
Pinging 192.168.1.10 with 32 bytes of data:
Reply from 10.0.0.1: Destination host unreachable.
Reply from 10.0.0.1: Destination host unreachable.
Reply from 10.0.0.1: Destination host unreachable.
Reply from 10.0.0.1: Destination host unreachable.
Ping statistics for 192.168.1.10:
Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
C:\>
Top


PC0
Physical
Config
Desktop
Programming
Attributes
Web Browser
URL
http://200.0.0.10
Go
Stop
Cisco Packet Tracer
Welcome to Cisco Packet Tracer. Opening doors to new opportunities. Mind Wide Open.
Quick Links:
A small page
Copyrights
Image page
Image


Router0
Physical
Config
CLI
Attributes
IOS Command Line Interface
Press RETURN to get started.
R1>en
R1#show ip nat translation
Pro Inside global Inside local Outside local Outside global
icmp 50.0.0.20:5 10.0.0.20:5 200.0.0.10:5 200.0.0.10:5
icmp 50.0.0.20:6 10.0.0.20:6 200.0.0.10:6 200.0.0.10:6
icmp 50.0.0.20:7 10.0.0.20:7 200.0.0.10:7 200.0.0.10:7
icmp 50.0.0.20:8 10.0.0.20:8 200.0.0.10:8 200.0.0.10:8
--- 50.0.0.10 10.0.0.10 --- ---
--- 50.0.0.20 10.0.0.20 --- ---
--- 50.0.0.30 10.0.0.30 --- ---
tcp 50.0.0.10:1025 10.0.0.10:1025 200.0.0.10:80 200.0.0.10:80
tcp 50.0.0.10:1026 10.0.0.10:1026 200.0.0.10:80 200.0.0.10:80
R1#
Ctrl+F6 to exit CLI focus
Copy
Paste
Top

For router1:

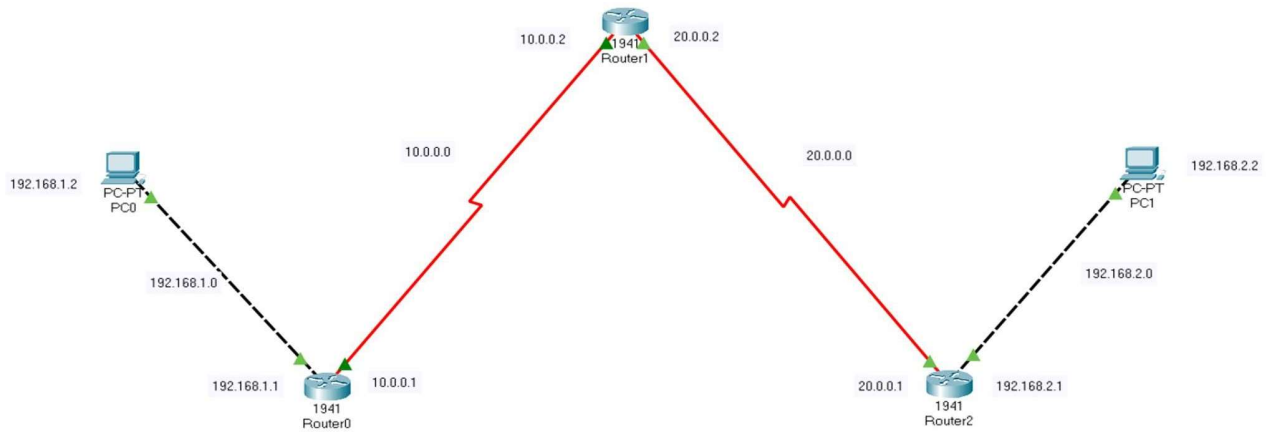
```
R2#show ip nat translation
Pro  Inside global     Inside local       Outside local      Outside global
---  200.0.0.10        192.168.1.10       ---                ---
tcp 200.0.0.10:80      192.168.1.10:80    50.0.0.10:1025     50.0.0.10:1025
tcp 200.0.0.10:80      192.168.1.10:80    50.0.0.10:1026     50.0.0.10:1026

R2#
```

Result:

Ex.No:13	IMPLEMENTATION OF VPN
Date:	

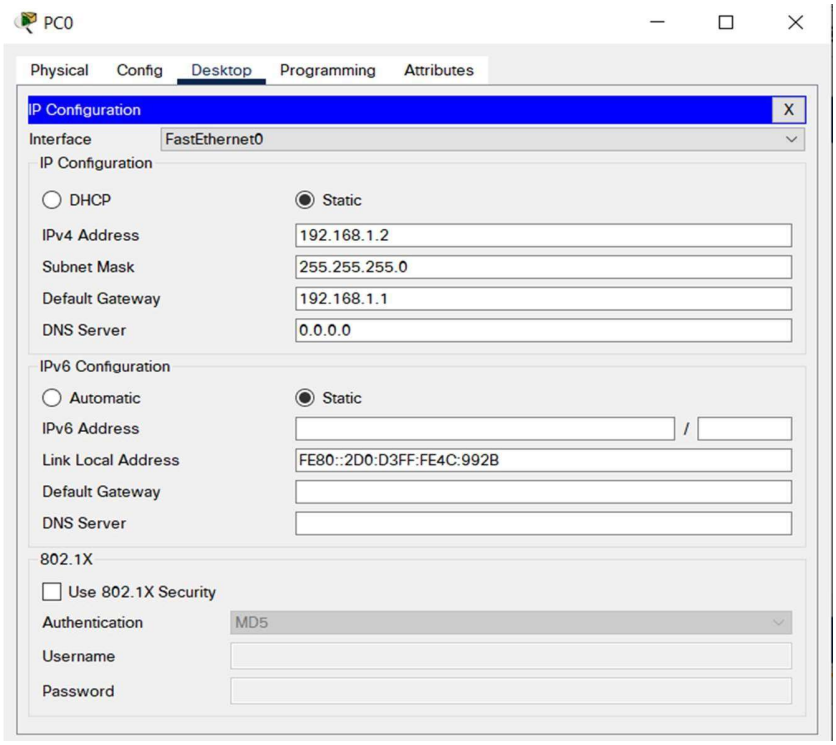
AIM :

PROCEDURE :



2 . Initial IP configuration.

Device / Interface	IP Address	Connected with




```
Router>enable

Router#config t

Router(config)#int gig0/0

Router(config-if)#ip add 192.168.1.1 255.255.255.0

Router(config-if)#no shut

Router(config-if)#exit

Router(config)#int se0/1/0

Router(config-if)#ip address 10.0.0.1 255.0.0.0

Router(config-if)#no shut
```

```
Router>enable
Router#config t
Router(config)#int se0/1/0
Router(config-if)#ip add 10.0.0.2 255.0.0.0
Router(config-if)#no shut
Router(config-if)#exit
Router(config)#int se0/1/1
Router(config-if)#ip add 20.0.0.1 255.0.0.0
Router(config-if)#no shut
```

CONFIGURATION ON ROUTER3:

```
Router>enable
Router#config t
Router(config)#int se0/1/0
Router(config-if)#ip add 20.0.0.2 255.0.0.0
Router(config-if)#no shut
Router(config-if)#exit
Router(config)#int gig0/0
Router(config-if)#ip add 192.168.2.1 255.255.255.0
Router(config-if)#no shut
```

```
Router>enable
Router#config t
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#ip route 0.0.0.0 0.0.0.0 10.0.0.2
Router(config)#
```

```
Router>enable
Router#config t
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#ip route 0.0.0.0 0.0.0.0 20.0.0.1
Router(config)#
```

```
Router#ping 20.0.0.2
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 20.0.0.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 26/28/33 ms
```

Now we go to Router3 and test the network by pinging Router1 interface.

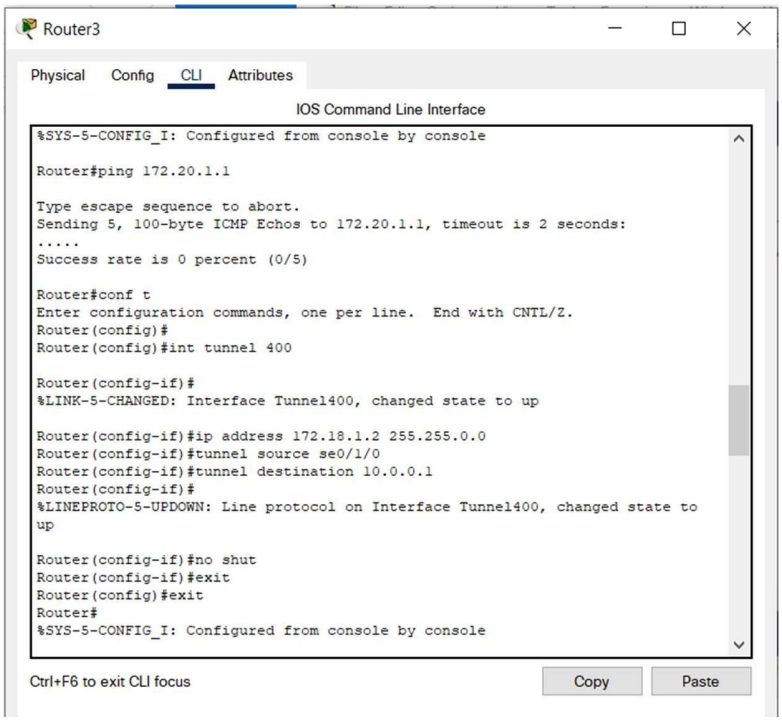
```
Router#ping 10.0.0.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.0.0.1, timeout is 2 seconds:
!!!!!
```

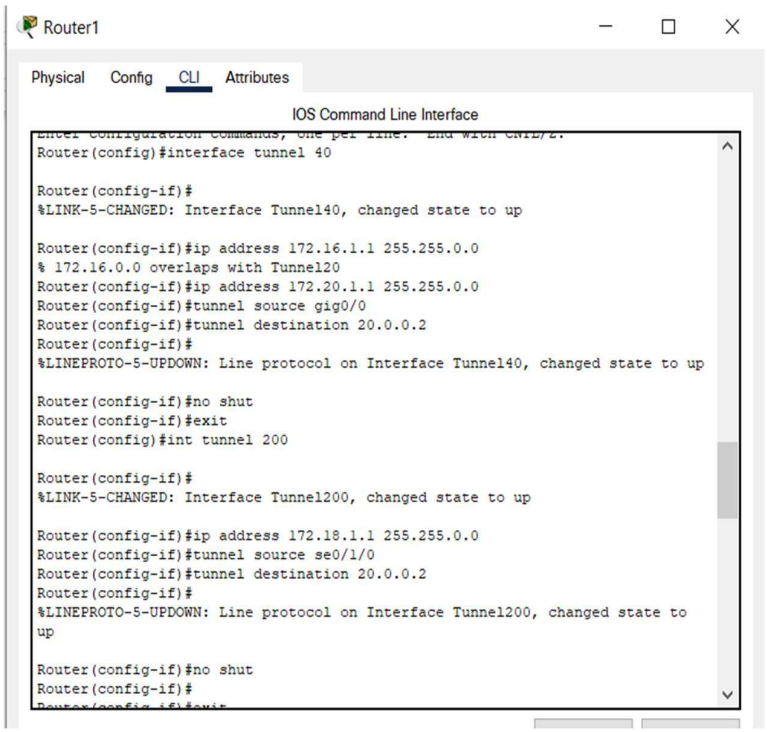
```
Success rate is 100 percent (5/5), round-trip min/avg/max = 25/28/32 ms
```

You can clearly see both routers pinging each other successfully.

```
Router#config t
Router(config)#interface tunnel 200
Router(config-if)#ip address 172.18.1.1 255.255.0.0
Router(config-if)#tunnel source se0/1/0
Router(config-if)#tunnel destination 20.0.0.2
Router(config-if)#no shut
```

```
Router#config t
Router(config)#interface tunnel 400
Router(config-if)#ip address 172.18.1.2 255.255.0.0
Router(config-if)#tunnel source se0/1/0
Router(config-if)#tunnel destination 10.0.0.1
Router(config-if)#no shut
```


Router3
Physical
Config
CLI
Attributes
IOS Command Line Interface
%SYS-5-CONFIG_I: Configured from console by console
Router#ping 172.20.1.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.20.1.1, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
Router#conf t
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#
Router(config)#int tunnel 400
Router(config-if)#
%LINK-5-CHANGED: Interface Tunnel400, changed state to up
Router(config-if)#ip address 172.18.1.2 255.255.0.0
Router(config-if)#tunnel source se0/1/0
Router(config-if)#tunnel destination 10.0.0.1
Router(config-if)#
%LINEPROTO-5-UPDOWN: Line protocol on Interface Tunnel400, changed state to up
Router(config-if)#no shut
Router(config-if)#exit
Router(config)#exit
Router#
%SYS-5-CONFIG_I: Configured from console by console
Ctrl+F6 to exit CLI focus
Copy
Paste


Router1
Physical
Config
CLI
Attributes
IOS Command Line Interface
Router(config)#interface tunnel 40
Router(config-if)#
%LINK-5-CHANGED: Interface Tunnel40, changed state to up
Router(config-if)#ip address 172.16.1.1 255.255.0.0
% 172.16.0.0 overlaps with Tunnel20
Router(config-if)#ip address 172.20.1.1 255.255.0.0
Router(config-if)#tunnel source gig0/0
Router(config-if)#tunnel destination 20.0.0.2
Router(config-if)#
%LINEPROTO-5-UPDOWN: Line protocol on Interface Tunnel40, changed state to up
Router(config-if)#no shut
Router(config-if)#exit
Router(config)#int tunnel 200
Router(config-if)#
%LINK-5-CHANGED: Interface Tunnel200, changed state to up
Router(config-if)#ip address 172.18.1.1 255.255.0.0
Router(config-if)#tunnel source se0/1/0
Router(config-if)#tunnel destination 20.0.0.2
Router(config-if)#
%LINEPROTO-5-UPDOWN: Line protocol on Interface Tunnel200, changed state to up
Router(config-if)#no shut
Router(config-if)#

8.

Router1

```
Router#ping 172.18.1.2
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.18.1.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 30/32/36 ms
Router#
```

Router2

```
Router#ping 172.18.1.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.18.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 33/45/83 ms
```

```
Router(config)#ip route 192.168.2.0 255.255.255.0 172.18.1.2
Router(config)#ip route 192.168.1.0 255.255.255.0 172.18.1.1
```

```
Router#show interfaces Tunnel 200
Tunnel200 is up, line protocol is up (connected)
Hardware is Tunnel
Internet address is 172.18.1.1/16
```

MTU 17916 bytes, BW 100 Kbit/sec, DLY 50000 usec,

reliability 255/255, txload 1/255, rxload 1/255

Encapsulation TUNNEL, loopback not set

Keepalive not set

Tunnel source 10.0.0.1 (FastEthernet0/1), destination 20.0.0.2

Tunnel protocol/transport GRE/IP

Key disabled, sequencing disabled

Checksumming of packets disabled

Tunnel TTL 255

Fast tunneling enabled

Tunnel transport MTU 1476 bytes

Tunnel transmit bandwidth 8000 (kbps)

Tunnel receive bandwidth 8000 (kbps)

Last input never, output never, output hang never

Last clearing of "show interface" counters never

Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 1

Queueing strategy: fifo

Output queue: 0/0 (size/max)

5 minute input rate 32 bits/sec, 0 packets/sec

5 minute output rate 32 bits/sec, 0 packets/sec

52 packets input, 3508 bytes, 0 no buffer

Received 0 broadcasts, 0 runts, 0 giants, 0 throttles

0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort

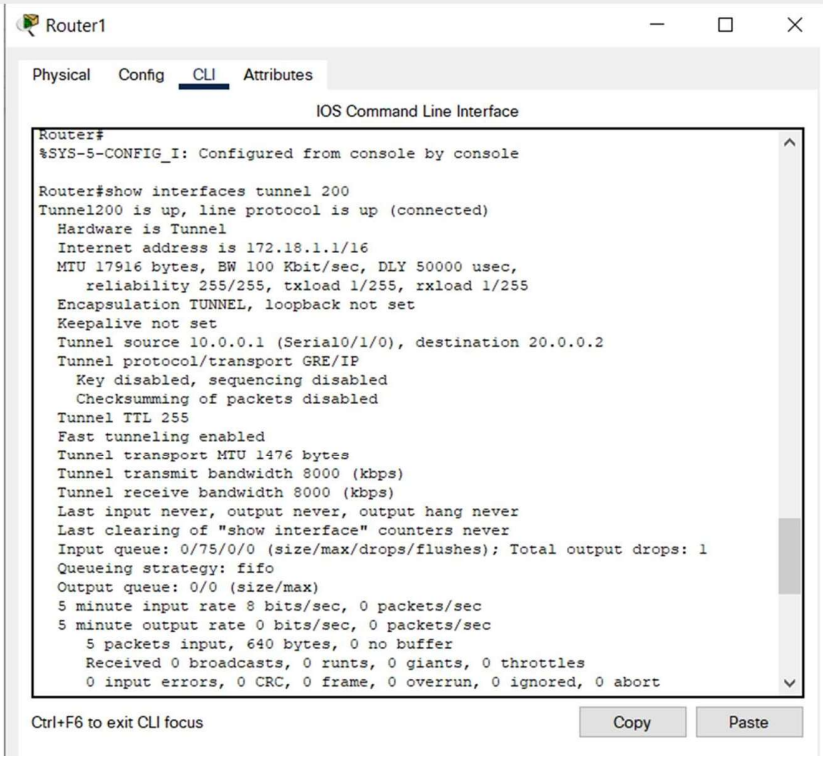
0 input packets with dribble condition detected

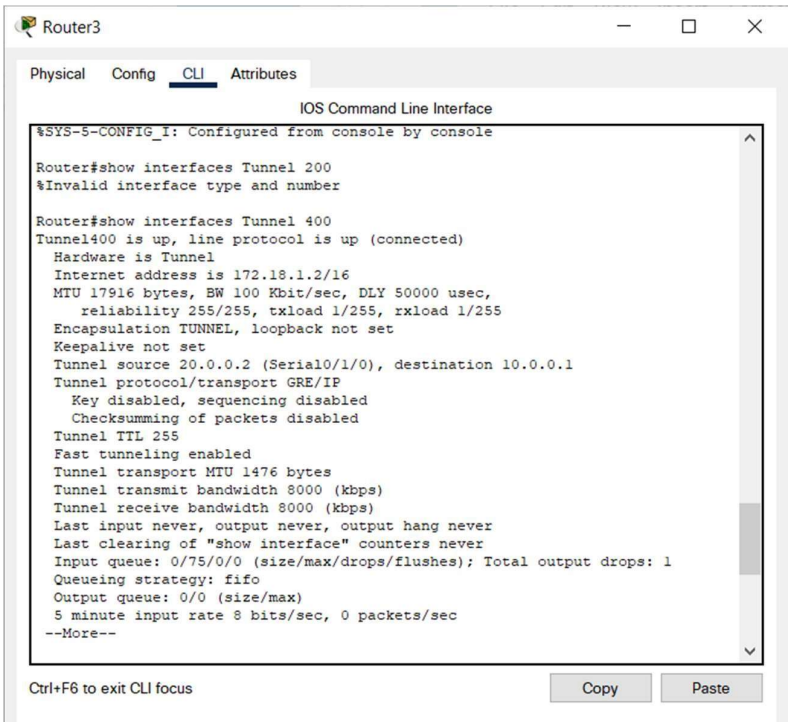
52 packets output, 3424 bytes, 0 underruns

0 output errors, 0 collisions, 0 interface resets

0 unknown protocol drops

0 output buffer failures, 0 output buffers swapped out





Now going to Router3 and test VPN Tunnel Creation:

```
Router #show interface Tunnel 400
Tunnel400 is up, line protocol is up (connected)
Hardware is Tunnel
Internet address is 172.18.1.2/16
MTU 17916 bytes, BW 100 Kbit/sec, DLY 50000 usec,
reliability 255/255, txload 1/255, rxload 1/255
Encapsulation TUNNEL, loopback not set
Keepalive not set
Tunnel source 20.0.0.2 (FastEthernet0/0), destination 10.0.0.1
Tunnel protocol/transport GRE/IP
Key disabled, sequencing disabled
Checksumming of packets disabled
Tunnel TTL 255
Fast tunneling enabled
Tunnel transport MTU 1476 bytes
Tunnel transmit bandwidth 8000 (kbps)
Tunnel receive bandwidth 8000 (kbps)
Last input never, output never, output hang never
Last clearing of "show interface" counters never
Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 1
Queueing strategy: fifo


Output queue: 0/0 (size/max)
5 minute input rate 32 bits/sec, 0 packets/sec
```

5 minute output rate 32 bits/sec, 0 packets/sec

52 packets input, 3424 bytes, 0 no buffer

Received 0 broadcasts, 0 runts, 0 giants, 0 throttles

0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort

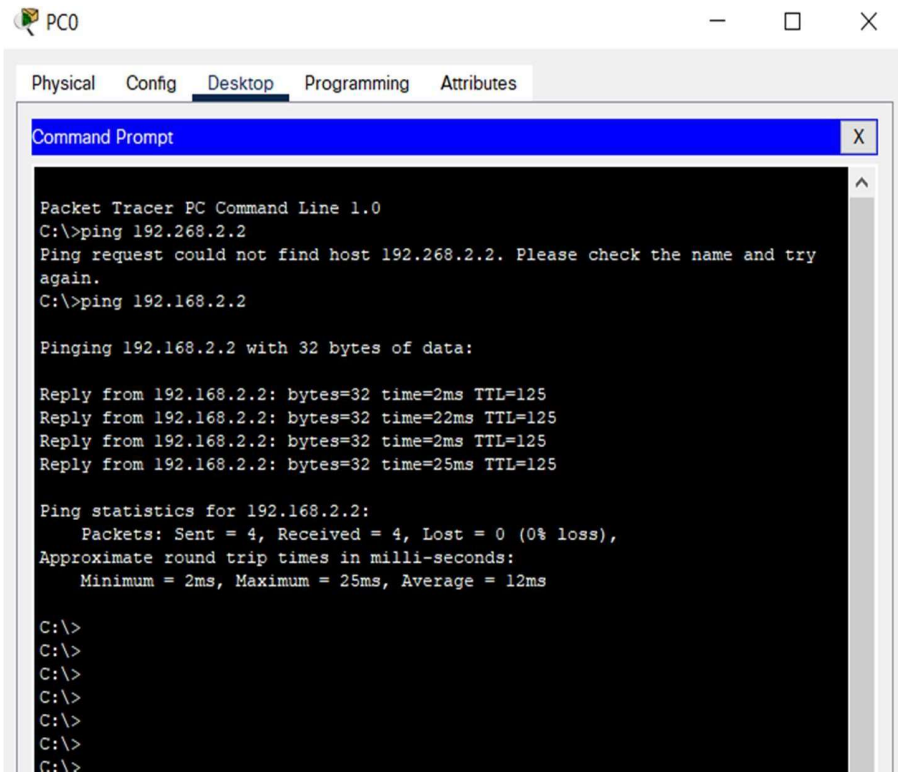
0 input packets with dribble condition detected

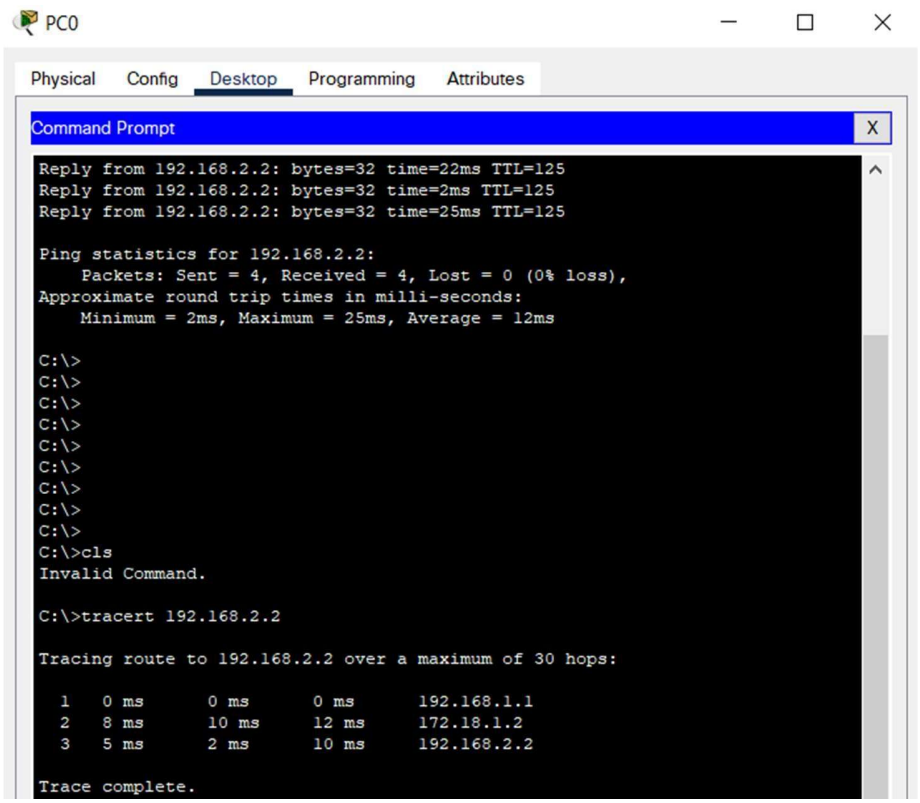
53 packets output, 3536 bytes, 0 underruns

0 output errors, 0 collisions, 0 interface resets

0 unknown protocol drops

11. Trace the VPN tunnel path.




PC0
Physical
Config
Desktop
Programming
Attributes
Command Prompt
X
Reply from 192.168.2.2: bytes=32 time=22ms TTL=125
Reply from 192.168.2.2: bytes=32 time=2ms TTL=125
Reply from 192.168.2.2: bytes=32 time=25ms TTL=125
Ping statistics for 192.168.2.2:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 2ms, Maximum = 25ms, Average = 12ms
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>cls
Invalid Command.
C:\>tracert 192.168.2.2
Tracing route to 192.168.2.2 over a maximum of 30 hops:
1 0 ms 0 ms 0 ms 192.168.1.1
2 8 ms 10 ms 12 ms 172.18.1.2
3 5 ms 2 ms 10 ms 192.168.2.2
Trace complete.

RESULT:

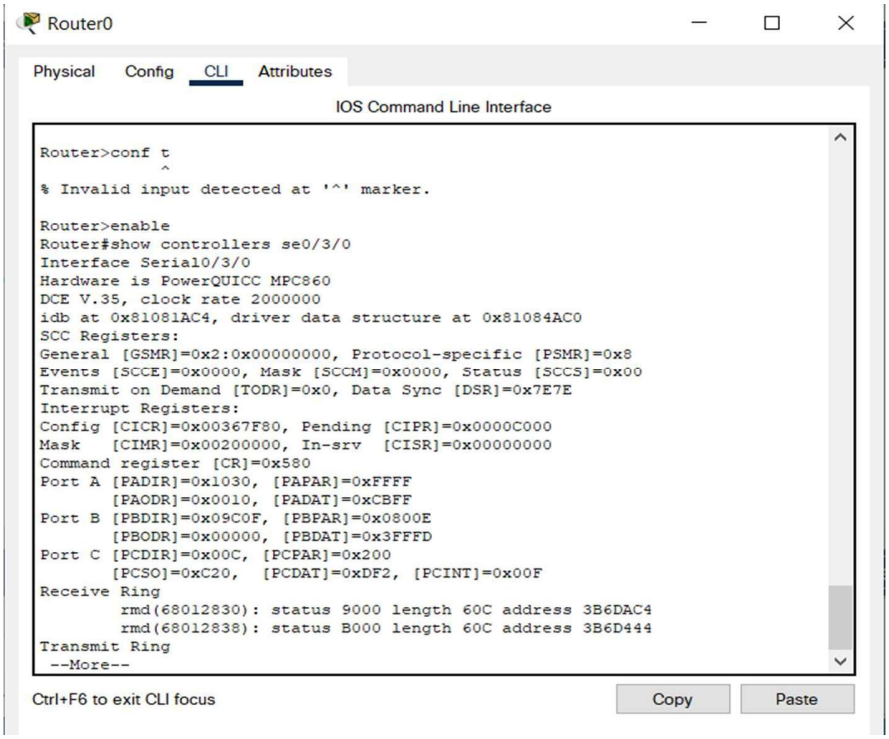
Ex.No:14	COMMUNICATION USING HDLC
Date:	

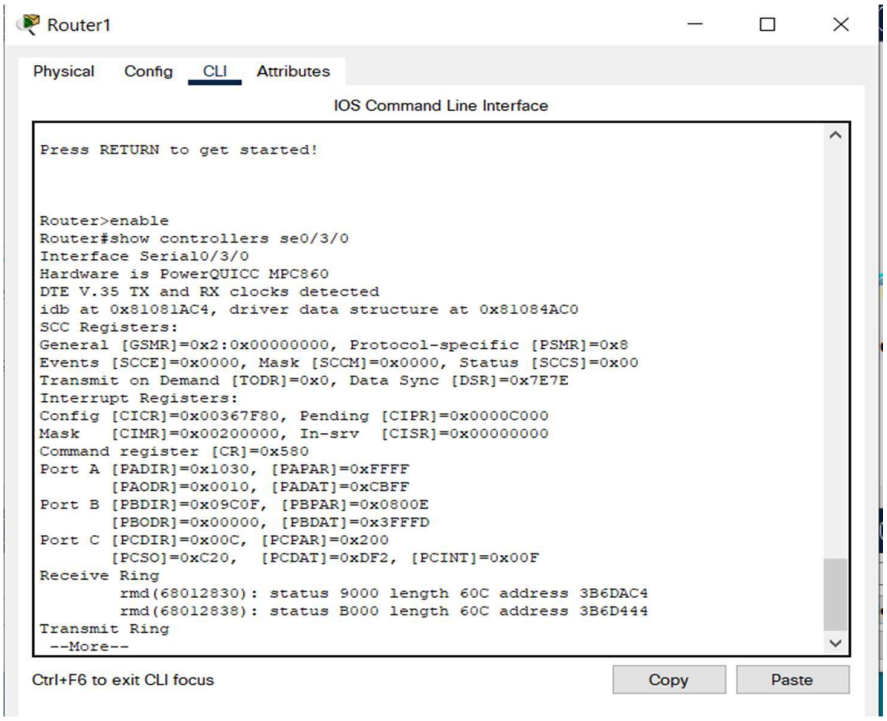
AIM:

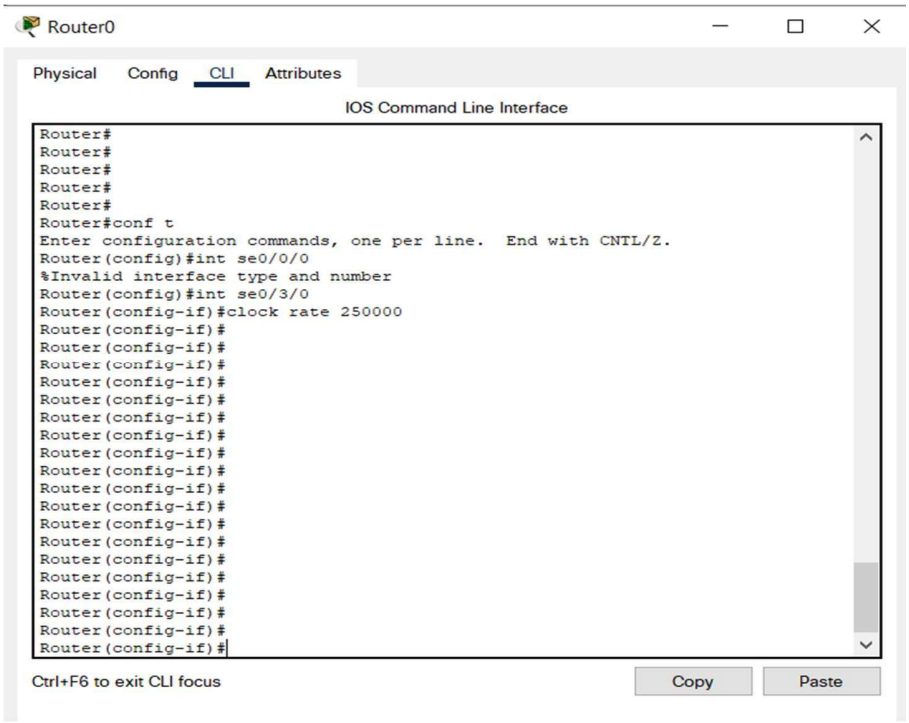
PROCEDURE:

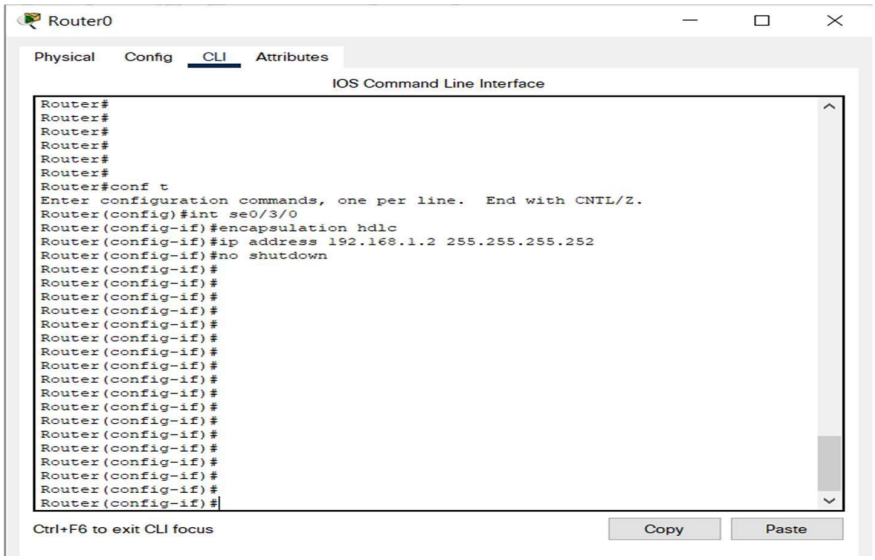
2 . Initial IP configuration.

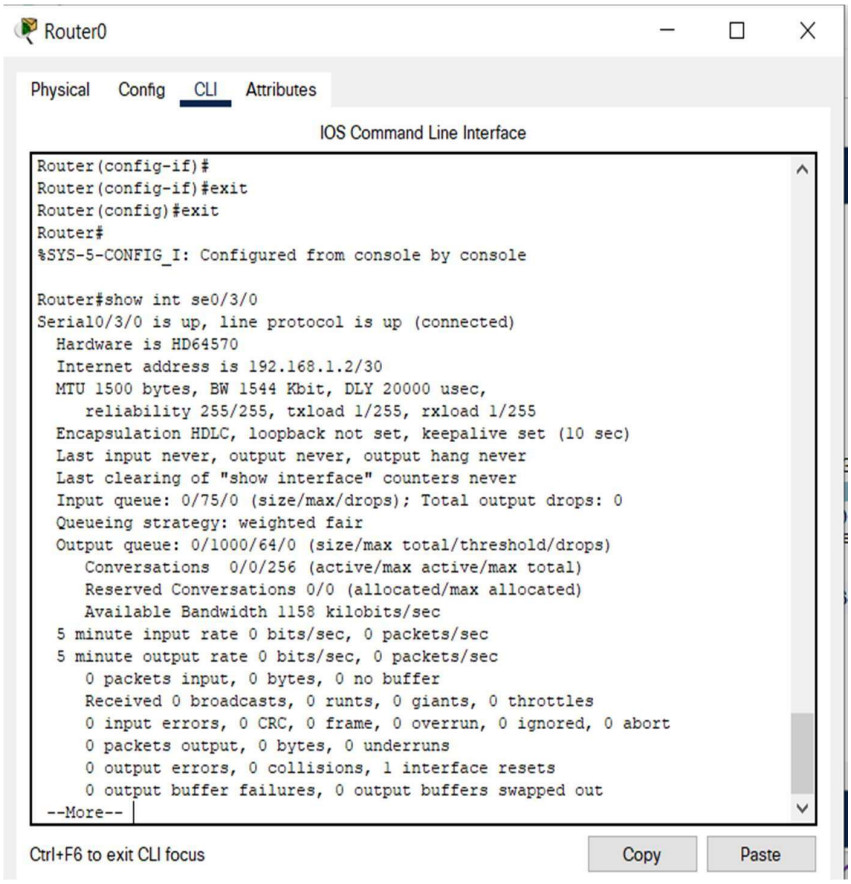
Device / Interface	IP Address	Connected with
PC0 / Fa0	10.0.0.2 /8	Router0 / Fa0/0
PC1 / Fa0	20.0.0.2 /8	Router1 / Fa0/0
Router0 / Se0/3/0	192.168.1.2 /30	Router1 / Se0/3/0
Router1 / Se0/3/0	192.168.1.3 /30	Router0 / Se0/3/0

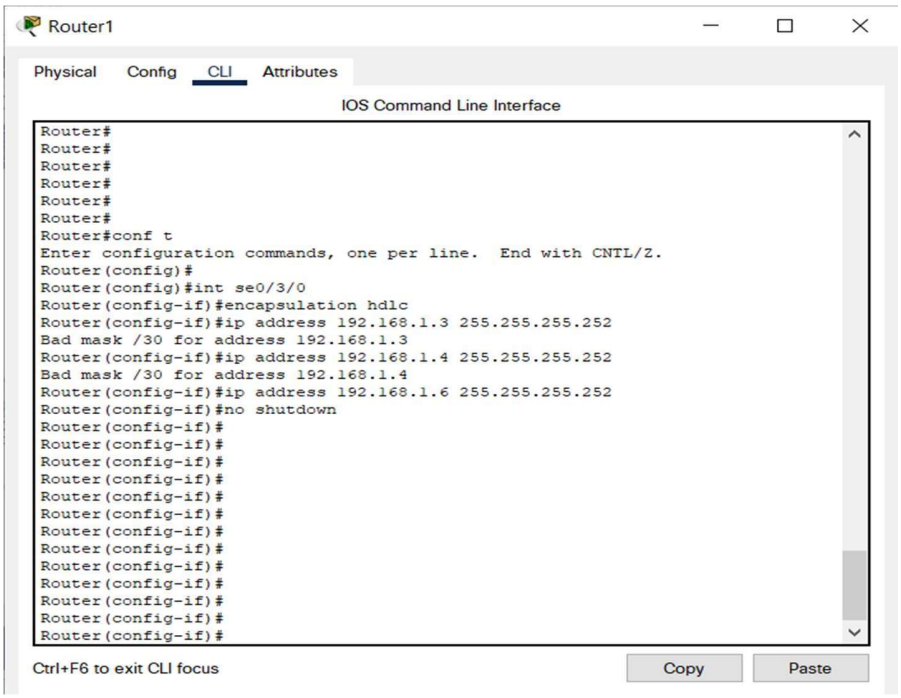

Router0
Physical
Config
CLI
Attributes
IOS Command Line Interface
Router>conf t
^
% Invalid input detected at '^' marker.
Router>enable
Router#show controllers se0/3/0
Interface Serial0/3/0
Hardware is PowerQUICC MPC860
DCE V.35, clock rate 2000000
idb at 0x81081AC4, driver data structure at 0x81084AC0
SCC Registers:
General [GSMR]=0x2:0x00000000, Protocol-specific [PSMR]=0x8
Events [SCCE]=0x0000, Mask [SCCM]=0x0000, Status [SCCS]=0x00
Transmit on Demand [TODR]=0x0, Data Sync [DSR]=0x7E7E
Interrupt Registers:
Config [CICR]=0x00367F80, Pending [CIPR]=0x0000C000
Mask [CIMR]=0x00200000, In-srv [CISR]=0x00000000
Command register [CR]=0x580
Port A [PADIR]=0x1030, [PAPAR]=0xFFFF
[PAODR]=0x0010, [PADAT]=0xCBFF
Port B [PBDIR]=0x09C0F, [PBPAR]=0x0800E
[PBODR]=0x00000, [PBDAT]=0x3FFFD
Port C [PCDIR]=0x00C, [PCPAR]=0x200
[PCSO]=0xC20, [PCDAT]=0xDF2, [PCINT]=0x00F
Receive Ring
rmd(68012830): status 9000 length 60C address 3B6DAC4
rmd(68012838): status B000 length 60C address 3B6D444
Transmit Ring
--More--
Ctrl+F6 to exit CLI focus
Copy
Paste


Router1
Physical
Config
CLI
Attributes
IOS Command Line Interface
Press RETURN to get started!
Router>enable
Router#show controllers se0/3/0
Interface Serial0/3/0
Hardware is PowerQUICC MPC860
DTE V.35 TX and RX clocks detected
idb at 0x81081AC4, driver data structure at 0x81084AC0
SCC Registers:
General [GSMR]=0x2:0x00000000, Protocol-specific [PSMR]=0x8
Events [SCCE]=0x0000, Mask [SCCM]=0x0000, Status [SCCS]=0x00
Transmit on Demand [TODR]=0x0, Data Sync [DSR]=0x7E7E
Interrupt Registers:
Config [CICR]=0x00367F80, Pending [CIPR]=0x0000C000
Mask [CIMR]=0x00200000, In-srv [CISR]=0x00000000
Command register [CR]=0x580
Port A [PADIR]=0x1030, [PAPAR]=0xFFFF
[PAODR]=0x0010, [PADAT]=0xCBFF
Port B [PBDIR]=0x09C0F, [PBPAR]=0x0800E
[PBODR]=0x00000, [PBDAT]=0x3FFFD
Port C [PCDIR]=0x00C, [PCPAR]=0x200
[PCSO]=0xC20, [PCDAT]=0xDF2, [PCINT]=0x00F
Receive Ring
rmd(68012830): status 9000 length 60C address 3B6DAC4
rmd(68012838): status B000 length 60C address 3B6D444
Transmit Ring
--More--
Ctrl+F6 to exit CLI focus
Copy
Paste


Router0
Physical
Config
CLI
Attributes
IOS Command Line Interface
Router#
Router#
Router#
Router#
Router#
Router#conf t
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#int se0/0/0
%Invalid interface type and number
Router(config)#int se0/3/0
Router(config-if)#clock rate 250000
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Ctrl+F6 to exit CLI focus
Copy
Paste


Router0
Physical
Config
CLI
Attributes
IOS Command Line Interface
Router#
Router#
Router#
Router#
Router#
Router#
Router#conf t
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#int se0/3/0
Router(config-if)#encapsulation hdlc
Router(config-if)#ip address 192.168.1.2 255.255.255.252
Router(config-if)#no shutdown
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Ctrl+F6 to exit CLI focus
Copy
Paste


Router0
Physical
Config
CLI
Attributes
IOS Command Line Interface
Router(config-if)#
Router(config-if)#exit
Router(config)#exit
Router#
%SYS-5-CONFIG_I: Configured from console by console
Router#show int se0/3/0
Serial0/3/0 is up, line protocol is up (connected)
Hardware is HD64570
Internet address is 192.168.1.2/30
MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
reliability 255/255, txload 1/255, rxload 1/255
Encapsulation HDLC, loopback not set, keepalive set (10 sec)
Last input never, output never, output hang never
Last clearing of "show interface" counters never
Input queue: 0/75/0 (size/max/drops); Total output drops: 0
Queueing strategy: weighted fair
Output queue: 0/1000/64/0 (size/max total/threshold/drops)
Conversations 0/0/256 (active/max active/max total)
Reserved Conversations 0/0 (allocated/max allocated)
Available Bandwidth 1158 kilobits/sec
5 minute input rate 0 bits/sec, 0 packets/sec
5 minute output rate 0 bits/sec, 0 packets/sec
0 packets input, 0 bytes, 0 no buffer
Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
0 packets output, 0 bytes, 0 underruns
0 output errors, 0 collisions, 1 interface resets
0 output buffer failures, 0 output buffers swapped out
--More--
Ctrl+F6 to exit CLI focus
Copy
Paste



8.

Router#ping 192.168.1.6

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 198.168.1.6, timeout is 2 seconds:

!!!!!

Success rate is 100 percent (5/5), round-trip min/avg/max = 26/28/33 ms

Now we go to Router1 and test the network by pinging the Router0 interface.

Router#ping 192.168.1.2

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 192.168.1.2, timeout is 2 seconds:

!!!!!

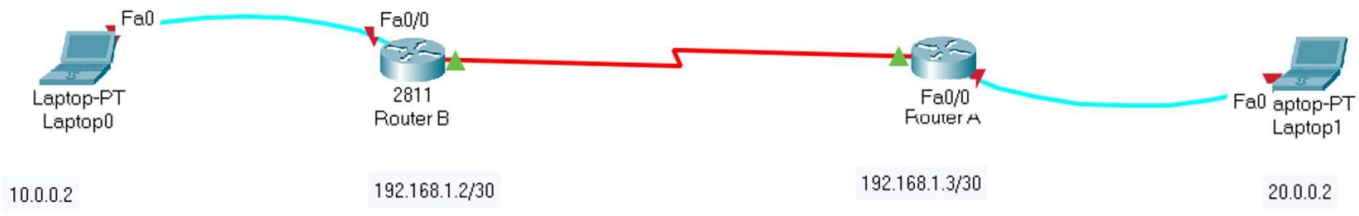
Success rate is 100 percent (5/5), round-trip min/avg/max = 25/28/32 ms

RESULT :

Ex.No:15	COMMUNICATION USING HDLC
Date:	

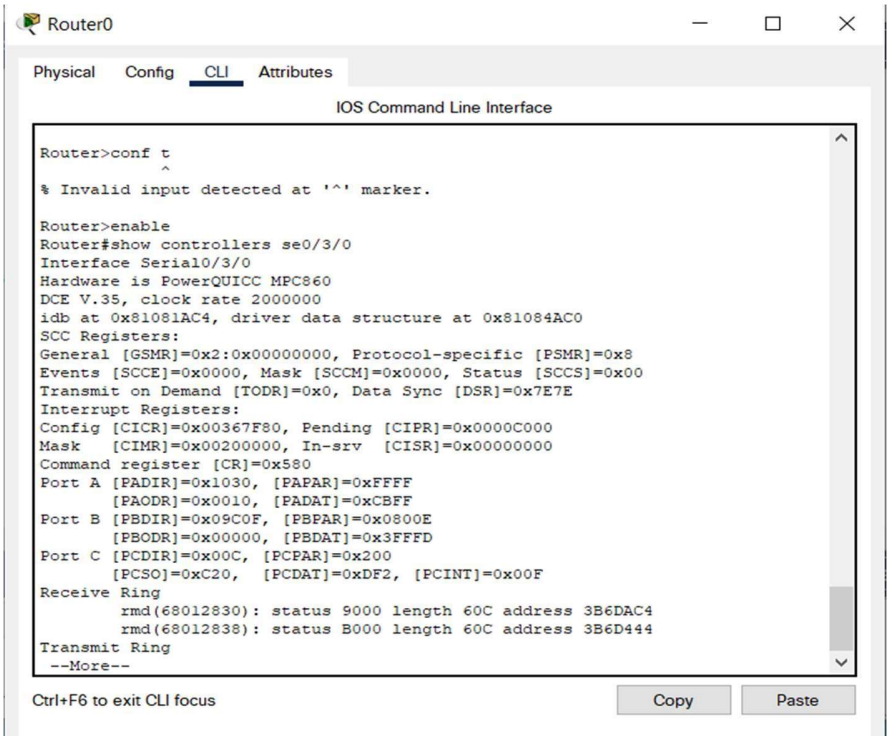
AIM:

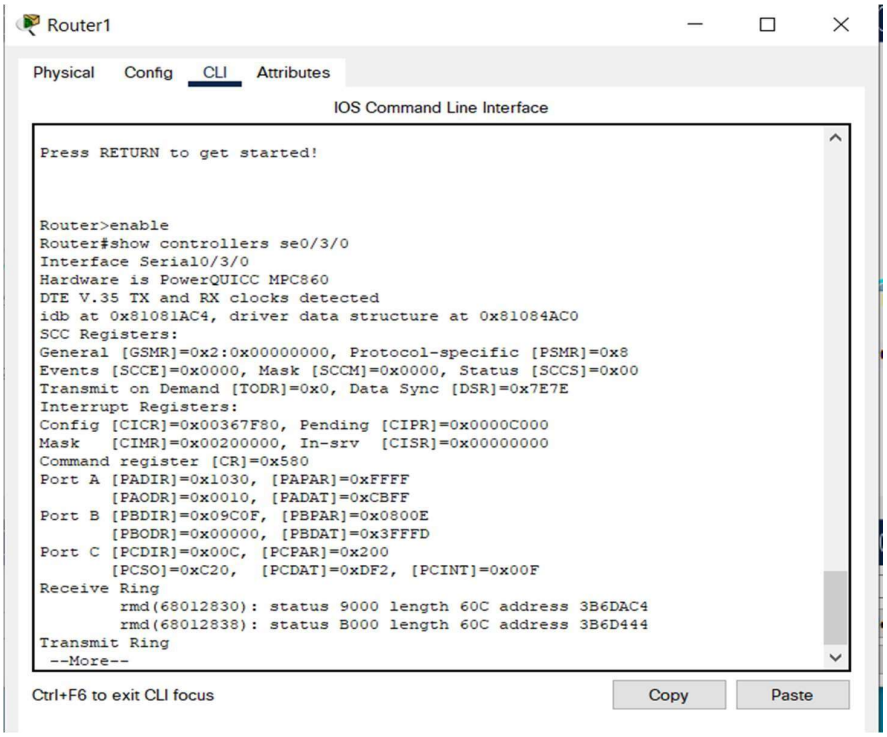
PROCEDURE:

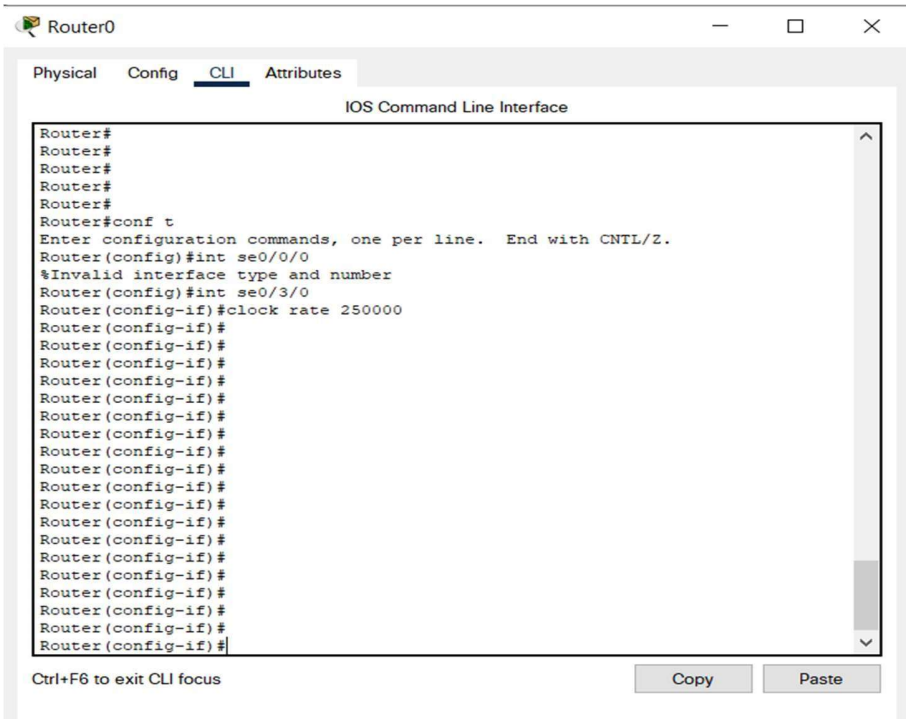


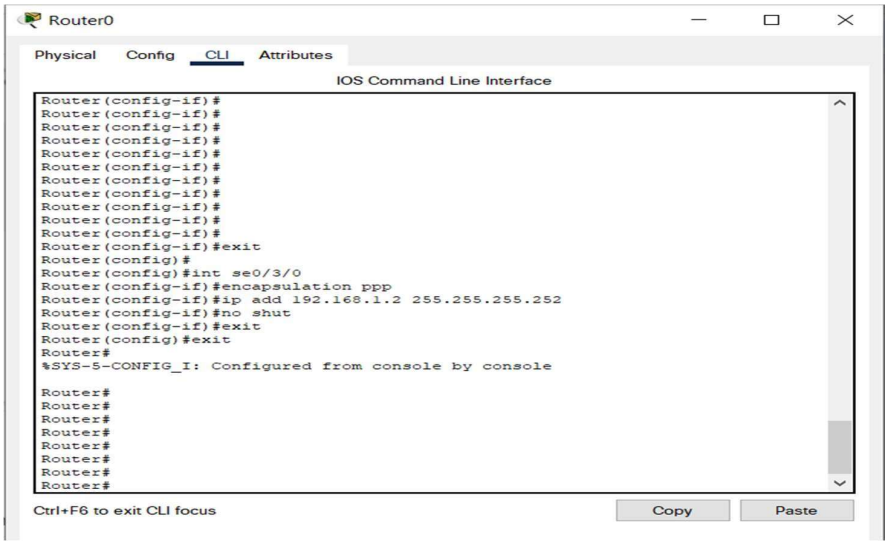
2 . Initial IP configuration.

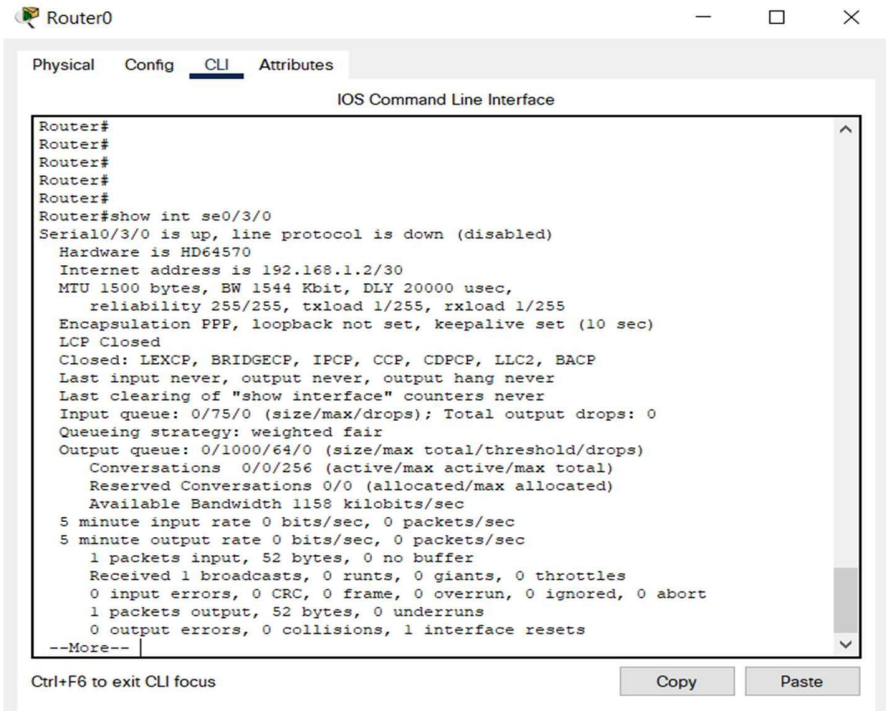
Device / Interface	IP Address	Connected with

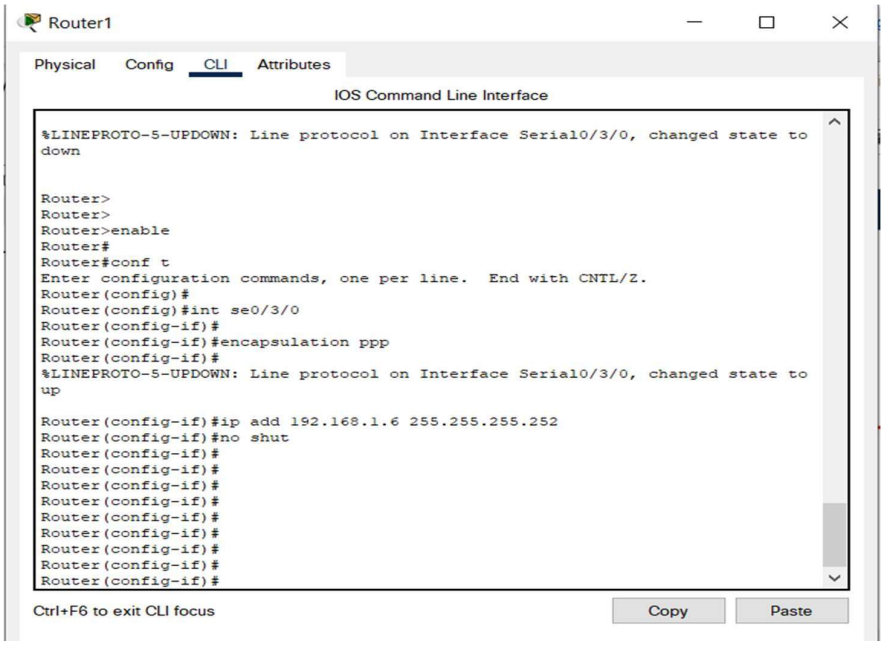

Router0
Physical
Config
CLI
Attributes
IOS Command Line Interface
Router>conf t
^
% Invalid input detected at '^' marker.
Router>enable
Router#show controllers se0/3/0
Interface Serial0/3/0
Hardware is PowerQUICC MPC860
DCE V.35, clock rate 2000000
idb at 0x81081AC4, driver data structure at 0x81084AC0
SCC Registers:
General [GSMR]=0x2:0x00000000, Protocol-specific [PSMR]=0x8
Events [SCCE]=0x0000, Mask [SCCM]=0x0000, Status [SCCS]=0x00
Transmit on Demand [TODR]=0x0, Data Sync [DSR]=0x7E7E
Interrupt Registers:
Config [CICR]=0x00367F80, Pending [CIPR]=0x0000C000
Mask [CIMR]=0x00200000, In-srv [CISR]=0x00000000
Command register [CR]=0x580
Port A [PADIR]=0x1030, [PAPAR]=0xFFFF
[PAODR]=0x0010, [PADAT]=0xCBFF
Port B [PBDIR]=0x09C0F, [PBPAR]=0x0800E
[PBODR]=0x00000, [PBDAT]=0x3FFFD
Port C [PCDIR]=0x00C, [PCPAR]=0x200
[PCSO]=0xC20, [PCDAT]=0xDF2, [PCINT]=0x00F
Receive Ring
rmd(68012830): status 9000 length 60C address 3B6DAC4
rmd(68012838): status B000 length 60C address 3B6D444
Transmit Ring
--More--
Ctrl+F6 to exit CLI focus
Copy
Paste

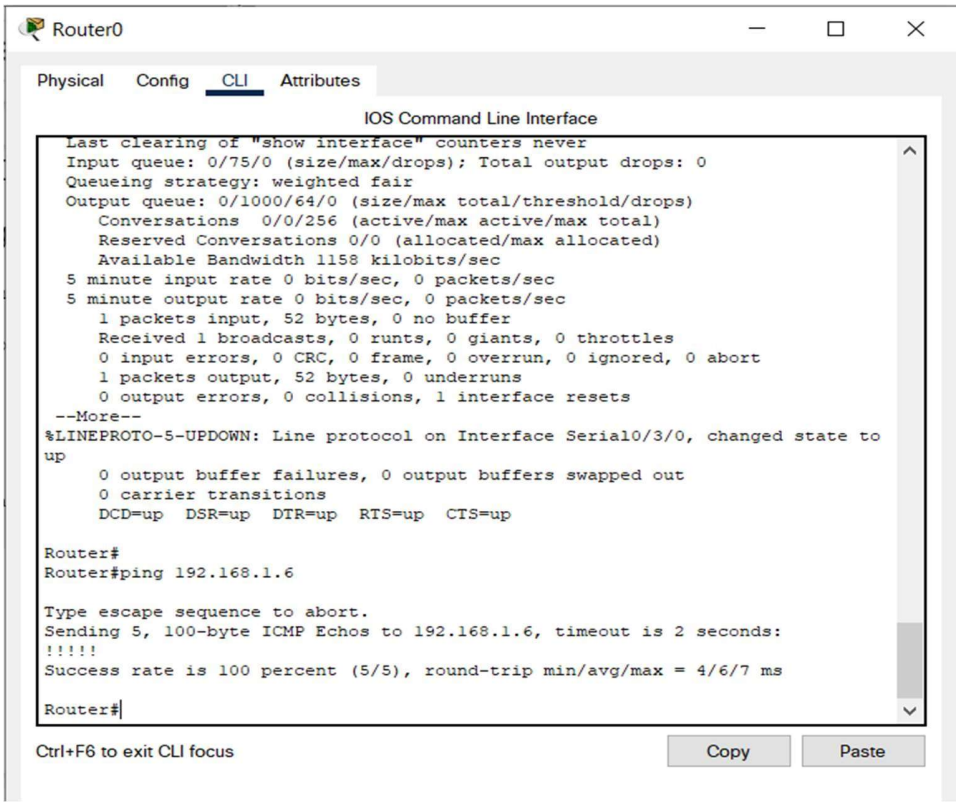

Router1
Physical
Config
CLI
Attributes
IOS Command Line Interface
Press RETURN to get started!
Router>enable
Router#show controllers se0/3/0
Interface Serial0/3/0
Hardware is PowerQUICC MPC860
DTE V.35 TX and RX clocks detected
idb at 0x81081AC4, driver data structure at 0x81084AC0
SCC Registers:
General [GSMR]=0x2:0x00000000, Protocol-specific [PSMR]=0x8
Events [SCCE]=0x0000, Mask [SCCM]=0x0000, Status [SCCS]=0x00
Transmit on Demand [TODR]=0x0, Data Sync [DSR]=0x7E7E
Interrupt Registers:
Config [CICR]=0x00367F80, Pending [CIPR]=0x0000C000
Mask [CIMR]=0x00200000, In-srv [CISR]=0x00000000
Command register [CR]=0x580
Port A [PADIR]=0x1030, [PAPAR]=0xFFFF
[PAODR]=0x0010, [PADAT]=0xCBFF
Port B [PBDIR]=0x09C0F, [PBPAR]=0x0800E
[PBODR]=0x00000, [PBDAT]=0x3FFFD
Port C [PCDIR]=0x00C, [PCPAR]=0x200
[PCSO]=0xC20, [PCDAT]=0xDF2, [PCINT]=0x00F
Receive Ring
rmd(68012830): status 9000 length 60C address 3B6DAC4
rmd(68012838): status B000 length 60C address 3B6D444
Transmit Ring
--More--
Ctrl+F6 to exit CLI focus
Copy
Paste


Router0
Physical
Config
CLI
Attributes
IOS Command Line Interface
Router#
Router#
Router#
Router#
Router#
Router#conf t
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#int se0/0/0
%Invalid interface type and number
Router(config)#int se0/3/0
Router(config-if)#clock rate 250000
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Ctrl+F6 to exit CLI focus
Copy
Paste

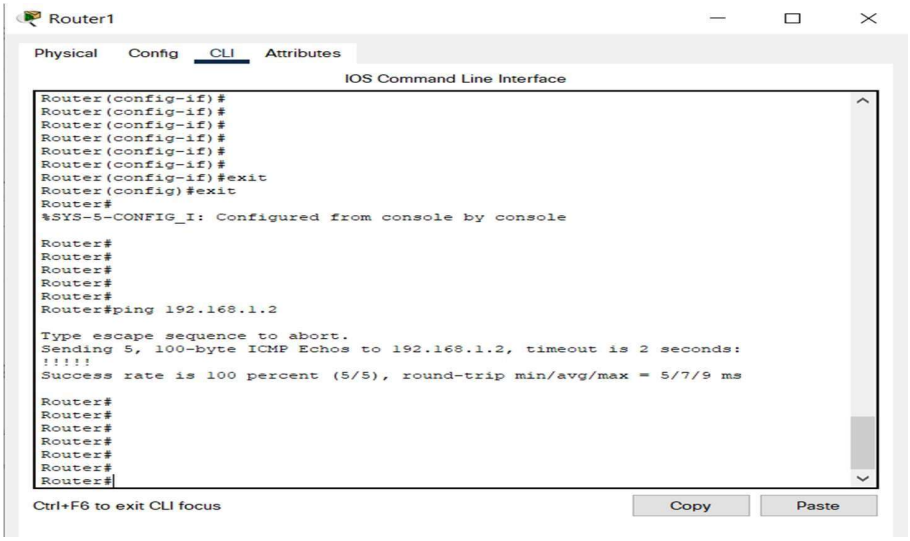

Router0
Physical
Config
CLI
Attributes
IOS Command Line Interface
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#exit
Router(config)#
Router(config)#int se0/3/0
Router(config-if)#encapsulation ppp
Router(config-if)#ip add 192.168.1.2 255.255.255.252
Router(config-if)#no shut
Router(config-if)#exit
Router(config)#exit
Router#
%SYS-5-CONFIG_I: Configured from console by console
Router#
Router#
Router#
Router#
Router#
Router#
Router#
Router#
Ctrl+F6 to exit CLI focus
Copy
Paste


Router0
Physical
Config
CLI
Attributes
IOS Command Line Interface
Router#
Router#
Router#
Router#
Router#
Router#show int se0/3/0
Serial0/3/0 is up, line protocol is down (disabled)
Hardware is HD64570
Internet address is 192.168.1.2/30
MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
reliability 255/255, txload 1/255, rxload 1/255
Encapsulation PPP, loopback not set, keepalive set (10 sec)
LCP Closed
Closed: LEXCP, BRIDGECP, IPCP, CCP, CDPCP, LLC2, BACP
Last input never, output never, output hang never
Last clearing of "show interface" counters never
Input queue: 0/75/0 (size/max/drops); Total output drops: 0
Queueing strategy: weighted fair
Output queue: 0/1000/64/0 (size/max total/threshold/drops)
Conversations 0/0/256 (active/max active/max total)
Reserved Conversations 0/0 (allocated/max allocated)
Available Bandwidth 1158 kilobits/sec
5 minute input rate 0 bits/sec, 0 packets/sec
5 minute output rate 0 bits/sec, 0 packets/sec
1 packets input, 52 bytes, 0 no buffer
Received 1 broadcasts, 0 runts, 0 giants, 0 throttles
0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
1 packets output, 52 bytes, 0 underruns
0 output errors, 0 collisions, 1 interface resets
--More--
Ctrl+F6 to exit CLI focus
Copy
Paste


Router1
Physical
Config
CLI
Attributes
IOS Command Line Interface
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0/3/0, changed state to down
Router>
Router>
Router>enable
Router#
Router#conf t
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#
Router(config)#int se0/3/0
Router(config-if)#
Router(config-if)#encapsulation ppp
Router(config-if)#
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0/3/0, changed state to up
Router(config-if)#ip add 192.168.1.6 255.255.255.252
Router(config-if)#no shut
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Ctrl+F6 to exit CLI focus
Copy
Paste



Now we go to Router1 and test the network by pinging the Router0 interface.



RESULT: