

Certificate

Name: NACHIKET G. KALLAPUR

Class:

Roll No:

Exam No:

Institution _____

*This is certified to be the bonafide work of the student in the
_____ Laboratory during the academic
year 20 / 20 .*

No. of practicals certified _____ out of _____ in the
subject of _____

.....
Teacher In-charge

.....
Examiner's Signature

.....
Principal

Date:

Institution Rubber Stamp

(N.B: The candidate is expected to retain his/her journal till he/she passes in the subject.)

Output Program

```
File Edit Selection C client.c X E test.txt
C client.c > O main(int char *[])
31     inet_pton(AF_INET, argv[1], &address.sin_addr);
32
33     if (connect(create_socket, (struct sockaddr*)&address,
34                 sizeof(sockaddr)) == -1) {
35         perror("The connection was accepted with the server 127.0.0.1...");
36     }
37
38     printf("Enter The Filenmae to Request : ");
39     scanf("%s", filename);
40
41     if (connect(create_socket, (struct sockaddr*)&address,
42                 sizeof(sockaddr)) == -1) {
43         perror("The connection was accepted with the server 127.0.0.1...");
44     }
45
46     if (send(create_socket, filename, strlen(filename), 0) == -1) {
47         perror("Request Accepted by server... Waiting to receive contents of file...");
48     }
49
50     if (recv(create_socket, buffer, 1024, 0) == -1) {
51         perror("Result obtained, the contents of file are...");
52     }
53
54     printf("Hey! This is Nachiket \"nerdyhunter\" Kallapur");
55
56
57 nachiketkallapur@DESKTOP-1K4S1LQ:~/nps_lab$ ./client 127.0.0.1
58
59 1
60
61 The Socket was created
62 The connection was accepted with the server 127.0.0.1...
63 Enter The Filenmae to Request : test.txt
64 Request Accepted by server...Waiting to receive contents of file...
65
66 Result obtained, the contents of file are...
67
68 Hey! This is Nachiket "nerdyhunter" Kallapur
69 EOF
70
71 nachiketkallapur@DESKTOP-1K4S1LQ:~/nps_lab$
```

WSL Ubuntu-18.04 ① 0 △ 0 Spaces: 4 UTF-8 LF C Linux 12:58 AM 14/06/2020

1) Implement a client - server communication using socket programming

server

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet.h>
#include <sys/stat.h>
#include <unistd.h>
#include <stdlib.h>
#include <strof.h>
#include <fcntl.h>
#include <arpa/inet.h>
#include <string.h>

int main( int argc , char *argv ) {
    int lsocket , create_socket , new_socket , addresslen , fd ;
    int buffsize = 1024 ;
    char *buffer = malloc( buffsize ) ;
    char frame[ 256 ] ;
    struct sockaddr_in address ;

    if (( create_socket = socket( AF_INET , SOCK_STREAM , 0 ) ) > 0 )
        fprintf (" Socket was created successfully " ) ;
    else
        exit( 0 ) ;

    address . sin_family = AF_INET ;
    inet_pton( AF_INET , argv[ 1 ] , &address . sin_addr ) ;
    address . sin_port = htons( atoi( argv[ 2 ] ) ) ;
    if ( bind( create_socket , ( struct sockaddr * ) &address , sizeof( address ) ) == 0 )
        printf (" Binding socket to IP : %s \n " , inet_ntoa( address . sin_addr ) ) ;
```

```
else exit(0);
```

```
listen(create_socket, 2);
```

```
address.sin_family = AF_INET; (struct sockaddr_in);
```

```
new_socket = accept(create_socket, (struct sockaddr*)&address, &addrlen);
```

```
if (new_socket > 0) {
```

```
    fprintf("The client is connected ... \n", inet_ntoa(address.sin_addr));
```

```
else exit(0);
```

```
recv(new_socket, fname, 255, 0);
```

```
printf("A request for filename %s received ... \n", fname);
```

```
if ((fd = open(fname, O_RDONLY)) < 0) {
```

```
    perror("File open failed"); exit(0);
```

```
}
```

```
while (cont = read(fd, buffer, bufsize)) > 0) {
```

```
    printf("Reading file contents \n");
```

```
    printf("Sending file contents to client \n");
```

```
    send(new_socket, buffer, cont, 0);
```

```
y
```

```
printf("Request complete \n");
```

```
close(new_socket);
```

```
return close(create_socket);
```

```
y
```

Client

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/stat.h>
#include <unistd.h>
#include <fcntl.h>
#include <netinet/in.h>
```

```
int main(int argc, char *argv[7]) {
    int create_socket, buffsize = 1024, lont;
    char *buffer = malloc(buffsize);
    char fname[256];
    struct sockaddr_in address;
```

```
if ((create_socket = socket(AF_INET, SOCK_STREAM, 0)) > 0)
    printf("socket was created\n");
else exit(0);
```

```
address.sin_family = AF_INET;
address.sin_port = htons(15002);
inet_pton(AF_INET, argv[1], &address.sin_addr);
```

```
if (connect(create_socket, (struct sockaddr *) &address, sizeof(address)) == 0)
    printf("The connection was accepted with the server\n");
else exit(0);
```

```
prompt("Enter the filename to request : ");  
scanf("%s", fname);  
send(create_socket, fname, sizeof(fname), 0);  
prompt("Request accepted by the server... Waiting to receive the  
contents of the file ... (\n\r)");  
prompt("Results obtained, the contents of the file are ... (\n\r)");  
  
while ((cont = recv(create_socket, buffer, buffsize, 0)) > 0)  
    write(1, buffer, cont);  
  
prompt("\nEOF\n");  
return close(create_socket);
```

Output - Distance Vector

```
File Edit Selection View Go Run Terminal Help
distancevector.c:189:10 (WSL: Ubuntu-20.04) - Visual Studio Code
PROBLEMS OUTPUT TERMINAL CONSOLE
1 bash
OPEN EDITORS 1 UNSAVED
NPS_Lab (WSL: UBUNTU-20)
  > vscode
  > program1
  > program2
  > distancevector
  > distancevector.c
  > program4
  > program5
  > program6
  > program7
  > programs
  (1) README.md

From Router 4 to Router 2
Cost: 7 | Path: 2 <- 3 <- 4

From Router 4 to Router 1
Cost: 4 | Path: 3 <- 4

From Router 4 to Router 4
Cost: 0 | Path: 4

From Router 4 to Router 5
Cost: 2 | Path: 5 <- 4

From Router 5 to Router 1
Cost: 18 | Path: 1 <- 2 <- 5

From Router 5 to Router 2
Cost: 9 | Path: 2 <- 5

From Router 5 to Router 3
Cost: 6 | Path: 3 <- 4 <- 5

From Router 5 to Router 4
Cost: 2 | Path: 4 <- 5

From Router 5 to Router 5
Cost: 0 | Path: 5
nachaketkallapur@DESKTOP-1K4S1LQ:~/NPS_Lab/programs$
```

2) Write a program to implement distance vector routing protocol for a simple topology of routers

→ code

```
#include <stdio.h>
```

```
int A[10][10], n, d[10], p[10];
```

```
int bellman_ford()
```

```
int i, u, v;
```

```
for (int i = 0; i < n; i++) {
```

```
    for (int u = 0; u < n; u++) {
```

```
        for (v = 0; v < n; v++) {
```

```
            if (d[v] > d[u] + A[u][v]) {
```

```
                d[v] = d[u] + A[u][v];
```

```
                p[v] = u;
```

```
            }
```

```
        }
```

```
    }
```

```
    if (d[v] > d[u] + A[u][v]) {
```

```
        printf("Detected negative edge");
```

```
        return -1;
```

```
    }
```

```
    return 0;
```

```
}
```

```
int main() {
```

```
    printf("N: "); scanf("%d", &n);
```

```
    printf("Matrix:\n");
```

```
    int source = 0, destination = 0;
```

```
    for (int i = 0; i < n; i++) for (int j = 0; j < n; j++) scanf("%d", &A[i][j]);
```

```

for(source = 0; source < n; source++) {
    for(destination = 0; destination < n; destination++) {
        for(int i = 0; i < n; i++) { if(d[i][j] == 999, p[i][j] = -1); }
        d[source] = 0;
        int valid = ullman_ford();
        if(valid == 0) { printf("Negative edge detected\n"); return -1; }
        printf("BFS Route %d to Router %d\n", source + 1, destination + 1);
        foundf("Total : %d d [Path]", d[destination]);
        if(destination != source) {
            int j = destination;
            while(p[j] != -1) {
                printf("%d d < ", j + 1);
                j = p[j];
            }
        }
    }
}

```

3

printf("%d\n", source + 1);

3

Y

Output Checksum

```
File Edit View Insert Cell Help Terminal Help  
Computing checksum at number 1702  
Enter IP header information 36 10 01 00  
Field 1:  
1 Field 2:  
2 Field 3:  
3 Field 4:  
4 Field 5:  
5 Field 6:  
6 Field 7:  
7 Field 8:  
8 Field 9:  
9  
Computed checksum at index 1702:  
checksum=11Apqgk0m32QK5  
2024-08-29 15:55:40 +0000 UTC
```

3) write a program to implement error detection and correction using checksum and hamming code

→ checksum

```
#include <stdio.h>
unsigned fields[10];
unsigned short checksum() {
    int i;
    int sum=0;
    printf("Enter IP header information in 16-bit words\n");
    for(i=0; i<9; i++) {
        printf("Field %d\n", i+1);
        scanf("%x", &fields[i]);
        sum = sum + (unsigned short) fields[i];
    }
    while(sum >> 16)
        sum = (sum & 0xFFFF) + (sum >> 16);
    }
}
```

~~for~~

sum = ~sum;

return (signed short) sum;

}

int main() {

unsigned short result1, result2;

result1 = checksum();

printf("\nComputed checksum at sender %x\n", result1);

11 Receiver

```
result2 = checksum();
printf ("In computed checksum at receiver %x\n", result2);
if (result1 == result2)
    printf ("No errors");
else
    printf ("Error in data received");
y
```

output -Hamming code

The screenshot shows a Windows desktop environment with a Visual Studio Code window open. The title bar reads "HammingCode - Apps (4) - Microsoft Store". The left sidebar shows a file tree with a folder named "HammingCode" containing files like "HammingCode.c", "HammingCode.h", "HammingCode.o", and "HammingCode.exe". The main area is a terminal window titled "TERMINAL" showing the output of a C program. The program prompts the user to enter a 4-bit data word (1 2 3 4), then asks for a 7-bit received codeword (1 2 3 4 0 1 0). It calculates a syndrome (000) and concludes that the received word is error-free.

```
1 2 3 4
enter 4 bit data word:
1 2 3 4
the 7bit Hamming code word:
1 2 3 4 0 1 0
enter the 7bit received codeword: 1 2 3 4 0 1 0
syndrome: 000
RECEIVED WORD IS ERROR FREE
nachiketkallapur@DESKTOP-1K4S1LQ:~/NPM_Lab/programs$
```

Hamming code:

```

→ #include <stdio.h>
void main()
{
    int data[10];
    int dataatrec[10], c, c1, c2, c3, i;
    printf("Sender: ");
    scanf("%d %d %d %d", &data[0], &data[1], &data[2], &data[3]);
    data[4] = data[0] ^ data[2] ^ data[4];
    data[5] = data[0] ^ data[1] ^ data[4];
    data[6] = data[0] ^ data[1] ^ data[2];
    printf("Encoded data: ");
    for (i=0; i<7; i++) printf("%d", &data[i]);
    printf("Receiver: ");
    for (i=0; i<7; i++) scanf("%d", &dataatrec[i]);
    c1 = dataatrec[6] ^ dataatrec[4] ^ dataatrec[2] ^ dataatrec[0];
    c2 = dataatrec[5] ^ dataatrec[4] ^ dataatrec[1] ^ dataatrec[0];
    c3 = dataatrec[3] ^ dataatrec[2] ^ dataatrec[1] ^ dataatrec[0];
    printf("\n Syndrome bits: %d %d %d", c1, c2, c3);
    if (c==0) printf("No error in transmission of data");
    else {
        printf("\n Error position: %d ", c);
        printf("In Data sent: ");
        for (i=0; i<7; i++) printf("%d", data[i]);
        printf("\n In Data Received: ");
        for (i=0; i<7; i++) printf("%d", dataatrec[i]);
        printf("\n Incorrect Message is: ");
        if (dataatrec[7-c]==0) dataatrec[7-c]=1;
        else dataatrec[7-c]=0;
    }
}

```

Expt. No. _____

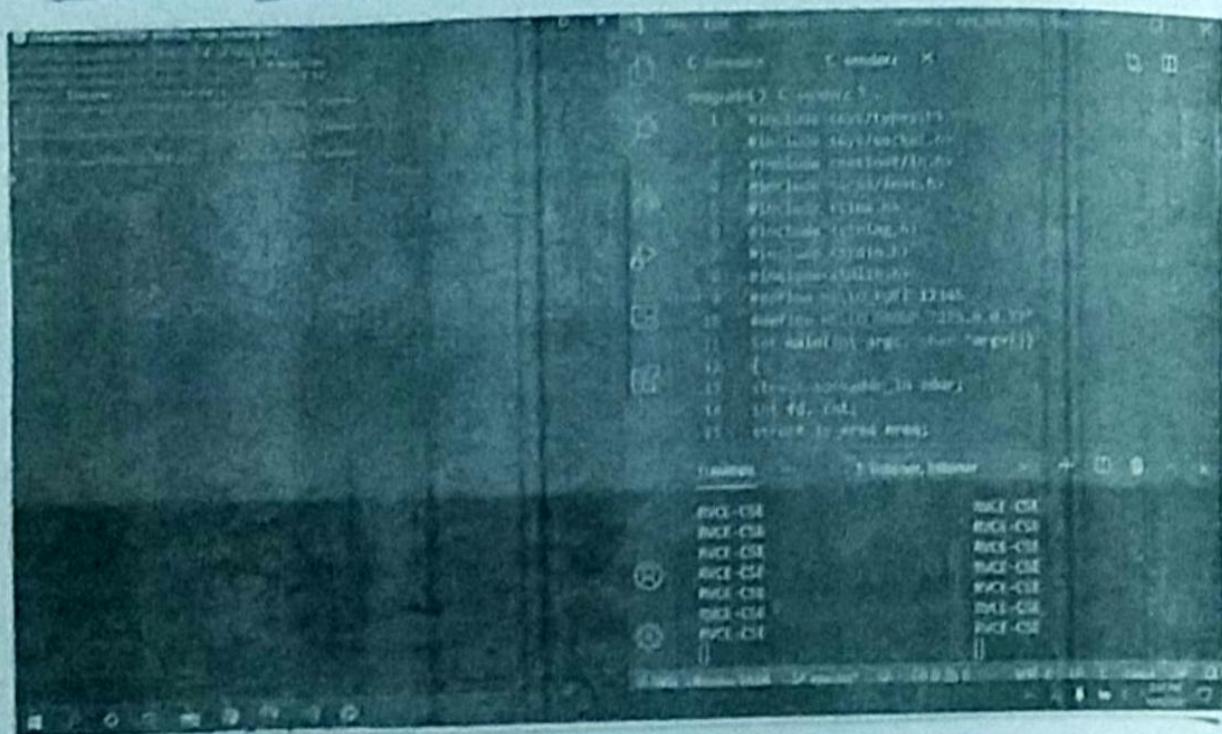
Date _____

Page No. _____

```
for (i=0; i<7; i++) {  
    printf ("%d", dataarray[i]);  
}  
y  
y  
y
```

Teacher's Signature _____

Output → Multicast program



*) Implement a simple multicast routing mechanism.

→ Listener

```
#include <sys/types.h> #include <sys/socket.h> #include <netinet/in.h>
#include <arpa/inet.h> #include <time.h> #include <ebang.h>
#include <strolio.h> #include <stplib.h>

#define HELLO_PORT 12345
#define HELLO_GROUP "225.0.0.37"
#define MSG_BUFSIZ 12E 1024000

int main(int argc, char* argv[])
{
    struct sockaddr_in addr;
    int fd; int nbytes, addrlen, m1;
    struct ip_mreq mreq;
    char msgbuf [MSG_BUFSIZ];
    m1 = 1;
    if (fd = socket(AF_INET, SOCK_DGRAM, 0) < 0) {
        perror(" Datagram socket error"); exit(1);
    } else {
        printf("Opening datagram socket... ok.\n");
    }

    if (getsockopt(fd, SOL_SOCKET, SO_REUSEADDR, &m1, sizeof(m1)) < 0) {
        perror("Reusing ADDR failed"); exit(1);
    } else {
        printf("Reusing ADDR... ok.\n");
    }

    bzero(&addr, sizeof(addr));
    addr.sin_family = AF_INET;
    addr.sin_addr.s_addr = htonl(INADDR_ANY);
    addr.sin_port = htons(HELLO_PORT);

    if (bind(fd, (const struct sockaddr*)&addr, sizeof(addr)) < 0) {
        perror("Bind error"); exit(1);
    } else {
        printf("Binding datagram socket... ok.\n");
    }
}
```

```

mreq.smr_multiaddr.s_addr = inet_addr("HELLO-GROUP");
mreq.smr_interface.s_addr = htonl(INADDR_ANY);
if (setsockopt(fd, IPPROTO_IP, IP_ADD_MEMBERSHIP, (char*)&mreq,
               sizeof(mreq)) < 0) {
    perror("Adding multicast error"); exit(1);
} else {
    printf("Adding multicast group.... ok.\n");
    ml = sizeof(mgbuf);
    while(1) {
        if ((nbytes = recvfrom(fd, mgbuf, sizeof(mgbuf), 0, NULL, NULL)) < 0)
            perror("Reading datagram message error"); exit(1);
        printf("The message from multicast server is : %s\n", mgbuf);
    }
}

```

Sender

```

→ #include <sys/types.h> #include <sys/socket.h> #include <netinet/in.h>
# include <arpa/inet.h> #include <time.h> #include <string.h>
#include <stropts.h> #include <stropts.h> #include <stropts.h>
#define HELLO_PORT 12345
#define HELLO_GROUP "225.0.0.37"
int main (int argc, char *argv[]) {
    struct sockaddr_in addr;
    int fd, cfd;
    char *message;
    struct mreq mreq;
    if(argc == 1) message = "PVC-E-CSF";
    else message = argv[1];
}

```

```

if ((fd = socket (AF_INET, SOCK_DGRAM, 0)) < 0) {
    perror ("opening datagram socket error"); exit(1);
} else
    printf ("Opening the datagram socket... ok.\n");
    memset (&addr, 0, sizeof(addr));
    addr.sin_family = AF_INET;
    addr.sin_addr.s_addr = inet_addr(HELLO_GROUP);
    addr.sin_port = htons(HELLO_PORT);

while (1) {
    if (sendto(fd, message, strlen(message), 0, (struct sockaddr *)&addr,
               sizeof(addr)) < 0) {
        perror ("sending datagram message error"); exit(1);
    } else
        printf ("Sending datagram message ... ok\n");
    sleep(2);
}

return 0;
}

```

socket - Programs

```
Client 1: C:\Users\HP\Desktop\Java\src>javac Client.java
Client 1: C:\Users\HP\Desktop\Java\src>java Client
Client 1: Client created 2 threads for port 8080
Client 1: Client created 2 threads for port 8080
Client 1: Client[clientCount] index 0 = acceptor_1
Client 1: Client[clientCount] index 1 = client1
Client 1: client1_create & thread(client1), index 0
Client 1: Client[clientCount] index 0 = acceptor_1
Client 1: Client[clientCount] index 1 = client1
Client 1: Connection established -----
Client 1: LIST
Client 1: client1 at socket 5
Client 1: Hello
Client 1: 1 on 2
Client 1: C:\Users\HP\Desktop\Java\src>java Client
Client 2: C:\Users\HP\Desktop\Java\src>javac Client.java
Client 2: C:\Users\HP\Desktop\Java\src>java Client
Client 2: Client created 2 threads for port 8080
Client 2: Client created 2 threads for port 8080
Client 2: Client[clientCount] index 0 = acceptor_1
Client 2: Client[clientCount] index 1 = client2
Client 2: client2_create & thread(client2), index 0
Client 2: Client[clientCount] index 0 = acceptor_1
Client 2: Client[clientCount] index 1 = client2
Client 2: Connection established -----
Client 2: LIST
Client 2: client2 at socket 5
Client 2: Hello
Client 2: 1 on 2
```

5) write a program to implement concurrent chat server that allows current logged in users to communicate one with other.

→ Server

```
#include <sys/types.h> #include <sys/socket.h> #include <netinet/in.h>
#include <sys/stat.h> #include <unistd.h> #include <stropts.h>
#include <stropts.h> #include <fcntl.h> #include <sys/types.h>
#include <stropts.h>

#define max 255

void sdr_echo(int connfd)
{
    int n;
    char *buffer = malloc(max);
    while ((n = read(connfd, buffer, max, 0)) > 0)
    {
        fputs("From client: ", stdout);
        fputs("\n", stdout);
        if (fgets(buffer, max, stdin) != NULL) send(connfd, buffer, strlen(buffer), 0);
        bzero(buffer, max);
    }
}

int main()
{
    int cont, listenfd, connfd, addrlen, addrlen2, fd, pid, addrlen3;
    struct sockaddr_in address, cli_address;
    if ((listenfd = socket(AF_INET, SOCK_STREAM, 0)) > 0)
        perror("socket was created");
    address.sin_family = AF_INET;
    address.sin_addr.s_addr = INADDR_ANY;
    address.sin_port = htons(16001);
    printf("Address before bind\ns.. \n", inet_ntoa(address.sin_addr));
    if (bind(listenfd, (struct sockaddr *) &address, sizeof(address)) == 0)
        perror("Binding socket\n");
    else perror("Bind error"); exit(0);
}
```

```

    printf("The address after bind %s...\n", inet_ntoa(address.sin_addr));
    listen(listenfd, 5);
    printf("Server is listening");
    gethostname(listenfd, (struct sockaddr*)&address, &addrlen);
    printf("Server's local address %s ... and port %d\n", inet_ntoa(address.sin_addr),
    htons(address.sin_port));
    for(i; i<5; i++) {
        addrlen = sizeof(struct sockaddr_in);
        connfd = accept(listenfd, (struct sockaddr*)&cli_address, &addrlen);
        addrlen2 = sizeof(struct sockaddr_in);
        int i = getpeername(connfd, (struct sockaddr*)&cli_address, &addrlen);
        printf("Client connected on port %d\n", ntohs(cli_address.sin_port));
    }
}

```

```

if((pid = fork()) == -1) {
    printf("Inode child");
    printf("\n");
    close(listenfd);
    strchx(connfd);
    exit(0);
}

```

y

```
close(connfd);
```

y

```
return 0;
```

y

Client

```

#include <sys/types.h> #include <sys/socket.h> #include <netinet/in.h>
#include <errno.h> #include <stropts.h> #include <stropts.h> #include <stropts.h>
#include <stropts.h>
#define max 255

```

```

void sendc(FILE *fp, int sockfd) {
    int cont; char *buffer = malloc(max); fprintf("To server:", stdout);
    while (fgets(buffer, max, fp) != NULL) {
        send(sockfd, buffer, strlen(buffer), 0);
        if ((cont = recv(sockfd, buffer, max, 0)) > 0) {
            fputs("From server:", stdout); fputs(buffer, stdout);
            if (strcmp(buffer, "exit", 4) == 0) {
                printf("client exit...\n"); break;
            }
            if (read(buffer, 10240) > 0)
                fputs("\nIt To server:", stdout);
        }
    }
    free(buffer);
}

int main(int argc, char *argv[]) {
    int create_socket; struct sockaddr_in address;
    if ((create_socket = socket(AF_INET, SOCK_STREAM, 0)) > 0)
        printf("socket created successfully");
    address.sin_family = AF_INET; address.sin_port = htons(16001);
    inet_nton(AF_INET, argv[1], &address.sin_addr);
    if (connect(create_socket, (struct sockaddr *) &address,
                sizeof(address)) == 0)
        printf("connection accepted with the server\n");
    else
        printf("Error in connect");
    close(create_socket);
}

```

Y

Output - TCP connection

The screenshot shows a Windows desktop with a terminal window open in WSL (Ubuntu 18.04). The terminal window title is "File Edit Selection /home/nachikethallapur/Programs/WSL/Ubuntu".

The code in the terminal is:

```
1 C:\server> cat server.c
2
3 #include <sys/types.h>
4 #include <sys/socket.h>
5 #include <netdb.h>
6 #include <errno.h>
7 #include <stdio.h>
8
9 #define SERV_PORT 5000
10
11 int main()
12 {
13     int listenfd, connfd;
14     struct sockaddr_in servaddr;
15     struct sockaddr_in client;
16     socklen_t clientlen;
17     char clientname[INET_NTOA_SIZE];
18     char clientip[INET_NTOA_SIZE];
19     int n;
20
21     bzero(&servaddr, sizeof(servaddr));
22     servaddr.sin_family = AF_INET;
23     servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
24     servaddr.sin_port = htons(SERV_PORT);
25
26     bind(listenfd, (struct sockaddr *)&servaddr,
27          LISTENQ);
28
29     listen(listenfd, LISTENQ);
30
31     for(;;)
32     {
33         connfd = accept(listenfd, (struct sockaddr *)&client,
34                         &clientlen);
35
36         if(connfd < 0)
37             perror("accept");
38
39         else
40         {
41             bzero(clientname, INET_NTOA_SIZE);
42             bzero(clientip, INET_NTOA_SIZE);
43
44             if(inet_ntoa(client.sin_addr) != INADDR_ANY)
45                 strcpy(clientname, inet_ntoa(client.sin_addr));
46             else
47                 strcpy(clientname, "Unknown Client");
48
49             if(inet_ntoa(client.sin_addr) != INADDR_ANY)
50                 strcpy(clientip, inet_ntoa(client.sin_addr));
51             else
52                 strcpy(clientip, "Unknown IP");
53
54             printf("Client %s (%s) connected\n", clientname, clientip);
55
56             n = read(connfd, "Hello!", 6);
57             if(n < 0)
58                 perror("read");
59             else
60             {
61                 write(connfd, "Hello!", 6);
62                 close(connfd);
63             }
64         }
65     }
66 }
```

The terminal output shows the execution of the server and client programs:

```
nachikethallapur@DESKTOP-1K4S1LQ:~/nps_lab/programs$ cd tcp
nachikethallapur@DESKTOP-1K4S1LQ:~/nps_lab/programs/tcp$ ls
client client.c server server.c
nachikethallapur@DESKTOP-1K4S1LQ:~/nps_lab/programs/tcp$ ./server
^C
nachikethallapur@DESKTOP-1K4S1LQ:~/nps_lab/programs/tcp$ ./client 127.0.0.1
My name is Nachiket
My name is Nachiket
Hey! I'm executing program 6
Hey! I'm executing program 6
```

6) Implementation of concurrent and iterative echo server using both connectionless and connection oriented socket system calls.

7) TCP (nonconcurrent):

Client:

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

```
void *cli(FILE *fp, int sockfd) {
    int buffsize = 1024;
    char *buffer = malloc(buffsize);
    while (fgets(buffer, buffsize, fp) != NULL) {
        if (strcmp(buffer, "exit\n") == 0)
            printf("Client Exit\n");
        send(sockfd, buffer, sizeof(buffer), 0);
        if (recv(sockfd, buffer, buffsize, 0) > 0)
            fputs(buffer, stdout);
    }
    fputs("\nEOF\n");
    free(buffer);
}
```

```
int main(int argc, char *argv[]) {
    int create_socket;
    struct sockaddr_in address;
    if ((create_socket = socket(AF_INET, SOCK_STREAM, 0)) > 0)
        printf("Socket was created\n");
    address.sin_family = AF_INET;
    address.sin_port = htons(15000);
    int pton(AF_INET, argv[1], &address.sin_addr);
}
```

```

if (connect(create_socket, (struct sockaddr*) &address, sizeof(address)) == -1)
    printf("Connection accepted with the server\n");
else {
    printf("Error in connect\n");
    exit(0);
}

```

```

strclr((char* create_socket));
return close(create_socket);
}

```

Server

```

#include <sys/types.h> #include <sys/socket.h> #include <netinet/in.h>
#include <sys/stat.h> #include <unistd.h> #include <stdlib.h>
#include <stdio.h> #include <fcntl.h> #include <arpa/inet.h>

```

void main echo (int connfd) {

int n;

int buffsize = 1024;

char *buffer = malloc(buffsize);

again:

while ((n = read(connfd, buffer, buffsize)) > 0)

send (connfd, buffer, n, 0);

if (n < 0) goto again;

free (buffer);

}

};

int main () {

int listenfd, connfd, addresslen, pid, addrlen;

struct sockaddr_in address, cli_address;

if ((listenfd = socket(AF_INET, SOCK_STREAM, 0)) > 0)

printf ("Socket was created\n");

address.sin_family = AF_INET; address.sin_addr.s_addr = INADDR_ANY;

```
address.sin_port = htons(1500);  
printf("Address before bind (%s:%d), net sockaddr_in addrs  
If (bind(listenfd, (struct sockaddr*) &address, sizeof(address)) == 0)  
printf("Binding failed (%d)\n");  
printf("Address after bind (%s:%d) net sockaddr_in addrs;  
listen(listenfd, 3);  
printf("Server is listening\n");  
gethostname(listenfd, (struct sockaddr*) &address, &addrlen);  
printf("The server's local address %s and port %d\n",  
inet_ntoa(address.sin_addr), htons(address.sin_port));  
for(;;){  
    address = recvfrom(listenfd, &buf, addrlen);  
    connfd = accept(listenfd, (struct sockaddr*) &buf, &addrlen);  
    int p = getpeername(connfd, (sockaddr*) &buf, &addrlen);  
    if (connfd > 0){  
        printf("Client %s connected on port %d", inet_ntoa(buf.sin_addr),  
               htons(buf.sin_port));  
        write(connfd, "Hello world", 13);  
        if (pid = fork() == 0){  
            printf("Inside child %d; close(%d); execve(%d);\n",  
                  getpid(), connfd);  
            exit(0);  
        }  
    }  
}
```

Output - TCP structure

TCP-ServerServers

```
#include <sys/types.h> #include <sys/socket.h> #include <netinet/in.h>
#include <sys/stat.h> #include <unistd.h> #include <stroff.h>
#include <stdio.h> #include <fcntl.h> #include <arpa/inet.h>
```

```
void str_echo(int connfd){
```

```
    int n;
```

```
    int buffsize = 1024;
```

```
    char * buffer = malloc(buffsize);
```

```
again:
```

```
    while ((n = read(connfd, buffer, buffsize, 0)) > 0)
```

```
        send(connfd, buffer, n, 0);
```

```
    if (n < 0)
```

```
        goto again;
```

```
}
```

```
int main(){
```

```
    int cont, listenfd, connfd, address, addrlen, fd, uid, addrlen3;
    struct sockaddr_in address, cli_address;
```

```
    if ((listenfd = socket(AF_INET, SOCK_STREAM, 0)) > 0)
```

```
        printf("Socket was created\n");
```

```
    address.sin_family = AF_INET;
```

```
    address.sin_addr.s_addr = INADDR_ANY;
```

```
    address.sin_port = htons(15001);
```

```
    printf("Address before bind is %s...\n", inet_ntoa(address.sin_addr));
```

```
    if (bind(listenfd, (struct sockaddr *) &address, sizeof(address)) == -1)
```

```
        printf("Binding Socket\n");
```

```
    printf("Address after bind is %s...\n", inet_ntoa(address.sin_addr));
```

```

listen(listenfd, 5);
printf("Server is listening\n");
for(;;) {
    addrlen = sizeof (struct sockaddr_in);
    connfd = accept(listenfd, (struct sockaddr *)&cli_address, &addrlen);
    printf("Client %s is connected on port %d\n",
           inet_ntoa(cli_address.sin_addr).s_addr, ntohs(cli_address.sin_port));
    write(connfd, "Hello Client", 13);
    close(connfd);
}
return 0;
}

```

Client

```

#include <stdio.h> #include <stdlib.h> #include <sys/types.h>
#include <sys/socket.h> #include <netinet/in.h> #include <fcntl.h>
#include <unistd.h> #include <sys/stat.h> #include <arpa/inet.h>
#include <string.h>

void str_cli(FILE *fp, int sockfd) {
    int buffsize = 1024, cont;
    char *buffer = malloc(buffsize);
    while (fgets(buffer, buffsize, fp) != NULL) {
        if (strcmp(buffer, "exit\n") == 0) {
            fprintf("Client Exit ---\n");
            break;
        }
        send(sockfd, buffer, sizeof(buffer), 0);
        if ((cont = recv(sockfd, buffer, buffsize, 0)) > 0)
            fputs(buffer, stdout);
    }
}

```

```
printf("EOF\n"); }
```

```
int main(int argc, char *argv[5]) {
```

```
    int create_socket;
```

```
    struct sockaddr_in address;
```

```
    if ((create_socket = socket(AF_INET, SOCK_STREAM, 0)) > 0)
```

```
        printf("Socket created\n");
```

```
    address.sin_family = AF_INET;
```

```
    address.sin_port = htons(15000);
```

```
    inet_pton(AF_INET, argv[1], &address.sin_addr);
```

```
    if (connect(create_socket, (struct sockaddr *) &address,
```

```
                sizeof(address)) == 0)
```

```
        printf("The client is connected to the server at... \n", argv[1]);
```

```
    else {
```

```
        perror("Error in connect\n");
```

```
        exit(0);
```

```
}
```

```
}
```

```
str_cli(stdin, create_socket);
```

```
return close(create_socket);
```

Teacher's Signature _____

output - von der

1. **Output**
2. **Input**
3. **Control**
4. **Memory**
5. **Processor**
6. **Power**
7. **Interfacing**
8. **Software**
9. **Hardware**
10. **System**
11. **Design**
12. **Implementation**
13. **Testing**
14. **Deployment**
15. **Maintenance**
16. **Evolution**
17. **Performance**
18. **Reliability**
19. **Cost**
20. **Energy**
21. **Environment**
22. **Safety**
23. **Security**
24. **Privacy**
25. **Ethics**
26. **Regulation**
27. **Standards**
28. **Protocols**
29. **Protocols**
30. **Protocols**
31. **Protocols**
32. **Protocols**
33. **Protocols**
34. **Protocols**
35. **Protocols**
36. **Protocols**
37. **Protocols**
38. **Protocols**
39. **Protocols**
40. **Protocols**
41. **Protocols**
42. **Protocols**
43. **Protocols**
44. **Protocols**
45. **Protocols**
46. **Protocols**
47. **Protocols**
48. **Protocols**
49. **Protocols**
50. **Protocols**
51. **Protocols**
52. **Protocols**
53. **Protocols**
54. **Protocols**
55. **Protocols**
56. **Protocols**
57. **Protocols**
58. **Protocols**
59. **Protocols**
60. **Protocols**
61. **Protocols**
62. **Protocols**
63. **Protocols**
64. **Protocols**
65. **Protocols**
66. **Protocols**
67. **Protocols**
68. **Protocols**
69. **Protocols**
70. **Protocols**
71. **Protocols**
72. **Protocols**
73. **Protocols**
74. **Protocols**
75. **Protocols**
76. **Protocols**
77. **Protocols**
78. **Protocols**
79. **Protocols**
80. **Protocols**
81. **Protocols**
82. **Protocols**
83. **Protocols**
84. **Protocols**
85. **Protocols**
86. **Protocols**
87. **Protocols**
88. **Protocols**
89. **Protocols**
90. **Protocols**
91. **Protocols**
92. **Protocols**
93. **Protocols**
94. **Protocols**
95. **Protocols**
96. **Protocols**
97. **Protocols**
98. **Protocols**
99. **Protocols**
100. **Protocols**

UDP Iterative

Server

```
#include <sys/types.h> #include <sys/conf.h> #include <sys/socket.h>
#include <strobf.h> #include <sys/stat.h> #include <inet/netinet.h>
#include <unistd.h> #include <fcntl.h> #include <sys/param.h>
```

```
void str_echo(int sockfd, struct sockaddr *cli_address, int cli_len) {
    int n, buffsize = 1024; int addrlen;
    char *buffer = malloc(buffsize);
    for (i;) {
        addrlen = cli_len;
        n = recvfrom(sockfd, buffer, buffsize, 0, cli_address, &addrlen);
        sendto(sockfd, buffer, n, 0, cli_address, addrlen);
    }
}
```

y

But main() {

int sockfd;

struct sockaddr_in servr_address, cli_address;

if (sockfd = socket(AF_INET, SOCK_DGRAM, 0)) > 0

printf("Socket was created\n");

servr_address.sin_family = AF_INET;

servr_address.sin_addr.s_addr = INADDR_ANY;

servr_address.sin_port = htons(16001);

printf("The address before bind is %s\n", inet_ntoa(servr_address.sin_addr));

if (bind(sockfd, (struct sockaddr *) &servr_address,

sizeof(servr_address)) == 0)

printf("Binding Socket\n");

str_echo (sockfd, (struct sockaddr*) &cli_address, sizeof(cli_address));
return;

}

Client

```
#include <sys/socket.h> #include <sys/types.h> #include <netinet/in.h>
#include <netinet/in.h> #include <sys/types.h> #include <sys/socket.h>
#include <sys/types.h> #include <sys/socket.h> #include <sys/types.h>
#include <sys/socket.h> #include <sys/types.h> #include <sys/socket.h>
#define SERV_PORT 9002
#define MAXLINE 1024
```

void dg_cli(FILE *fp, int sockfd, const struct sockaddr_in servaddr,
socklen_t servlen) {

int n;

char sendline[MAXLINE], receive[MAXLINE];

while (fgets(sendline, MAXLINE, fp) != NULL) {

sendto(sockfd, sendline, strlen(sendline), 0, &servaddr, servlen);
if (strncmp(receive, "exit", 4) == 0) {

fprintf(stderr, "Client Exit\n");

break;

}

n = recvfrom(sockfd, receive, MAXLINE, 0, NULL, NULL);

receive[n] = '\0';

fp puts(receive, stdout);

y

y

```

int main( int argc, char *argv[] ) {
    int sockfd;
    struct sockaddr_in serv_addr;
    if ( (sockfd = socket( AF_INET, SOCK_DGRAM, 0 )) > 0 )
        printf( "socket was created\n" );
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons( 16001 );
    setsockopt( AF_INET, argv[1], &serv_addr, sin_addr );
    dg_dlx( std::cin, sockfd, (struct sockaddr*)&serv_addr,
            sizeof(serv_addr) );
    exit(0);
}
    
```

Output - Bug ?

D S R

| Line Number | Code | Comment | Value |
|-------------|---------------------------|---------------------------|-------|
| 1 | Count = 0; | Count = 0 | 0 |
| 2 | for(i = 0; i < 10; i++) | for(i = 0; i < 10; i++) | |
| 3 | { | { | |
| 4 | cout << " "; | cout << " " | |
| 5 | } | } | |
| 6 | cout << endl; | cout << endl | |
| 7 | } | | |
| 8 | main() | main() | |
| 9 | { | { | |
| 10 | int a[10]; | a[10] | 0 |
| 11 | for(i = 0; i < 10; i++) | for(i = 0; i < 10; i++) | |
| 12 | a[i] = i + 1; | a[i] = i + 1 | 1 |
| 13 | | | |
| 14 | cout << a[0]; | cout << a[0] | 1 |
| 15 | cout << endl; | cout << endl | |
| 16 | cout << a[9]; | cout << a[9] | 10 |
| 17 | cout << endl; | cout << endl | |
| 18 | } | | |
| 19 | cout << endl; | cout << endl | |
| 20 | } | | |
| 21 | return 0; | return 0; | 0 |

② Implementation of remote command execution using socket system calls.

→ Server

```
#include <sys/types.h> #include <sys/socket.h> #include <string.h>
```

```
#include <netinet/in.h> #include <netdb.h> #include <errno.h>
```

```
#include <sys/types.h> #include <sys/socket.h> #include <errno.h>
```

```
int main()
```

```
int sd, accept, len, bytes, port;
```

```
char send[50], receive[50];
```

```
struct sockaddr_in servaddr;
```

```
if ((sd = socket(AF_INET, SOCK_STREAM, 0)) > 0) {
```

```
    printf("Error in socket\n");
```

```
    exit(0);
```

```
}
```

```
else
```

```
    printf("Socket created successfully...\n");
```

```
len = sizeof(serv);
```

```
serv.sin_family = AF_INET;
```

```
serv.sin_port = htons(8080);
```

```
serv.sin_addr.s_addr = htonl(INADDR_ANY);
```

```
if (bind(sd, (struct sockaddr*)&serv, sizeof(serv)) < 0)
```

```
    printf("Error in bind\n");
```

```
    exit(0);
```

```
}
```

```
else printf("Binding socket to IP : %s\n", inet_ntoa(serv.sin_addr));
```

```
listen(sd, 3);
```

```
if (Caupt == acht (sd, (struct sockaddr*)NULL, NULL)) <= 0) {
```

```
    perror ("Error in accept\n");
```

```
    exit(0);
```

Y

```
else {
    cout << "Client connected...\n";
```

```
    aclient(i);
```

```
    bytes = recv(acht, receive, 50, 0);
```

```
    receive[bytes] = '\0';
```

```
    if (strcmp (receive, "end\n") == 0) {
```

```
        close(acht);
```

```
        clx (sd);
```

```
        exit(0);
```

Y

```
else {
```

```
    if (strcmp (receive, "1.5") == 0) {
```

```
        system ("clear");
```

```
        perror ("\n");
```

Y

Y

Client

```
#include <stdio.h> #include <stdlib.h> #include <string.h>
```

```
#include <unistd.h> #include <netinet/in.h> #include <sys/types.h>
```

```
#include <sys/socket.h> #include <sys/sockio.h> #include <errno.h>
```

```
int main()
```

```
    int sd, acht, den, bytes, port;
```

```
    char send[50], receive[50];
```

```
    struct sockaddr_in servclis;
```

```

if ((sd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
    perror("Error in socket\n");
    exit(0);
}

else {
    printf("socket created successfully...\n");
    zero(&zero, sizeof(zero));
    zero.sin_family = AF_INET;
    zero.sin_port = htons(8080);
    zero.sin_addr.s_addr = htonl(INADDR_ANY);
    if (connect(sd, (struct sockaddr*)&zero, sizeof(zero)) < 0) {
        perror("Error in connection\n");
        exit(0);
    }

    else {
        printf("connection established with the server...\n");
        while (1) {
            printf("Enter the command: ");
            gets(cmd);
            if (strcmp(cmd, "end") != 0) send(sd, cmd, 50, 0);
            else {
                send(sd, sendr, 50, 0);
                close(sd);
                break;
            }
        }
    }
}

```

output - RSA

File Edit Selection View Run Terminal Help

rsa.cpp

Programs > C/C++/C/C++(Windows) long int gcd(long int a, long int b)

```
#include <iostream>
#include <cmath.h>
#include <math.h>
#include <string.h>
using namespace std;
long int gcd(long int a, long int b)
{
    if(a == 0)
        return b;
    if(b == 0)
        return a;
    return gcd(b, a%b);
}
```

PROBLEMS OUTPUT TERMINAL DEBUG CONTROL

nachiketkallurupuri@DESKTOP-1K4S1LQ:~/rsa_lab/programs\$ g++ -o rsa rsa.cpp
nachiketkallurupuri@DESKTOP-1K4S1LQ:~/rsa_lab/programs\$./rsa
Enter the text to be encrypted: Nachiket
Two prime numbers (p and q) are: 13 and 23
 $n(p \cdot q) = 13 \cdot 23 = 299$
 $(p - 1) \cdot (q - 1) = 264$
Public key (n, e): (299, 101)
Private key (n, d): (299, 223)
Encrypted message: 52111831324815975116
Decrypted message: Nachiket
nachiketkallurupuri@DESKTOP-1K4S1LQ:~/rsa_lab/programs\$

In 8 Col 11 Spaces 4 UTF-8 LF C++ max R D

8) write a program to encrypt and decrypt the data using RSA and exchange the key securely using Diffie-Hellman Key exchange protocol

→ RSA

```
#include <iostream> // include <stlib.h> // include <math.h>
#include <string.h>
using namespace std;
```

```
long long gcd(long int a, long int b) {
    if (a == 0) return b;
    if (b == 0) return a;
    return gcd(b, a % b);
```

}

```
long int isPrime(long int a) {
    int i;
    for (i = 2; i < a; i++) {
        if ((a % i) == 0)
            return 0;
    }
```

return 1;

}

```
long int encrypt(char ch, long int n, long int e) {
    int p;
    long int temp = ch;
    for (i = 1; p < e; i++)
        temp = (temp * ch) % n;
    return temp;
```

authn- DiffieHellman

The screenshot shows a terminal window titled "NFS LAB (WSL Ubuntu 20.04) - Virtual Studio Code". The terminal displays the following output:

```
print("The private key a for Alice is : 9\n");
print("The private key b for Bob is : 3\n");
Secret key for the Alice is : 9
Secret Key for the Bob is : 9
nachiketkallam@DESKTOP-3KA51LQ:~/NFS_Lab/programs$
```

The terminal also shows the command used to compile and run the program:

```
gcc -o dh.c -I . ./dh
```

The code itself is a C program named "dh.c" which performs the Diffie-Hellman key exchange. It includes a function "power" which is undefined, leading to a compilation error.

```
#include <math.h>
#include <stdio.h>

int power(int G, int b, int P)
{
    int y = 1;
    for (int i = 0; i < b; i++)
        y = (y * G) % P;
    return y;
}

int main()
{
    printf("The private key a for Alice is : 9\n");
    printf("The private key b for Bob is : 3\n");
    int ka = power(G, a, P); // secret key for Alice
    int kb = power(G, b, P); // secret key for Bob
    int s = (ka * kb) % P;
    printf("The value of P is : 23\n");
    printf("The value of G is : 9\n");
    printf("Secret key for the Alice is : %d\n", s);
    printf("Secret Key for the Bob is : %d\n", s);
}
```

char decript (long int ch, long int n, long int d) {
int i;

long int temp = ch;

for (i = 1; i < d; i++)

ch = (temp + ch) % n;

return ch;

}

int main () {

long int p, len;

long int p, q, n, phi, e, d, cipher[50];

char text[50];

cout << "Enter time the text to be encrypted: ";

cin > getline (text, sizeof (text));

len = strlen (text);

do {

p = rand() % 50;

} while (!expprime (p));

do {

q = rand() % 50;

} while (!expprime (q));

n = p * q;

phi = (p - 1) * (q - 1);

do {

e = rand() % phi;

} while (gcd(phi, e) != 1);

do {

d = rand() % phi;

} while (gcd(phi, d) != 1) ; while ((d * e) % phi != 1);

```

cout << "Two prime numbers (p and q) are : " << p << " and " << q << endl;
cout << " n(p+q) = " << p << " + " << q << " + " << p+q << endl;
cout << "(p-1)*(q-1)" = " << phi << endl;
cout << " Public Key (n, e) : (" << n << ", " << e << ")\n";
cout << " Private Key (n, d) : (" << n << ", " << d << ")\n";
for(i=0; i<len; i++)
    cipher[i] = encrypt(text[i], n, e);
cout << " Encrypted message : ";
for(i=0; i<len; i++)
    cout << cipher[i];
for(i=0; i<len; i++)
    text[i] = decrypt(cipher[i], n, d);
cout << endl;
return 0;
}

```

Digital Hellman

```

→ #include <iostream.h>
#include <math.h>
long long int power(long long int a, long long int b, long long int p){
    if(b==1) return a;
    else return ((long long int)power(a,b))%p;
}

```

list main()

```

long long int P, g1, x, a, y, b, ka, kb;
p = 23;

```

```

printf("The value of P: %d\n", P);

```

$g_1 = 9;$ `printf ("The value of g1 : %d\n", g1);` $a = 4;$ `printf ("The private key a for Alice : %d\n", a);``x = power(g1, a, P);` $b = 3;$ `printf ("The private key b for Bob : %d\n", b);``y = power(g1, b, P);``ka = power(y, a, P);``kb = power(x, b, P);``printf ("Secret key for the Alice is : %d\n", ka);``printf ("Secret key for the Bob is : %d\n", kb);``return 0;``y`

Part - 1a

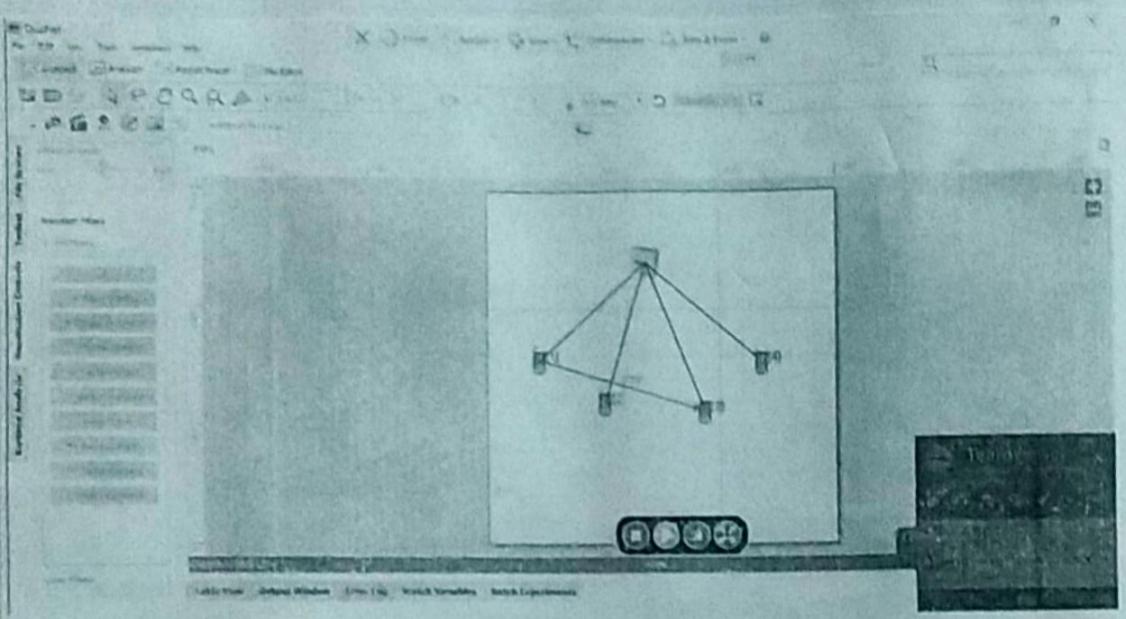


Fig 1

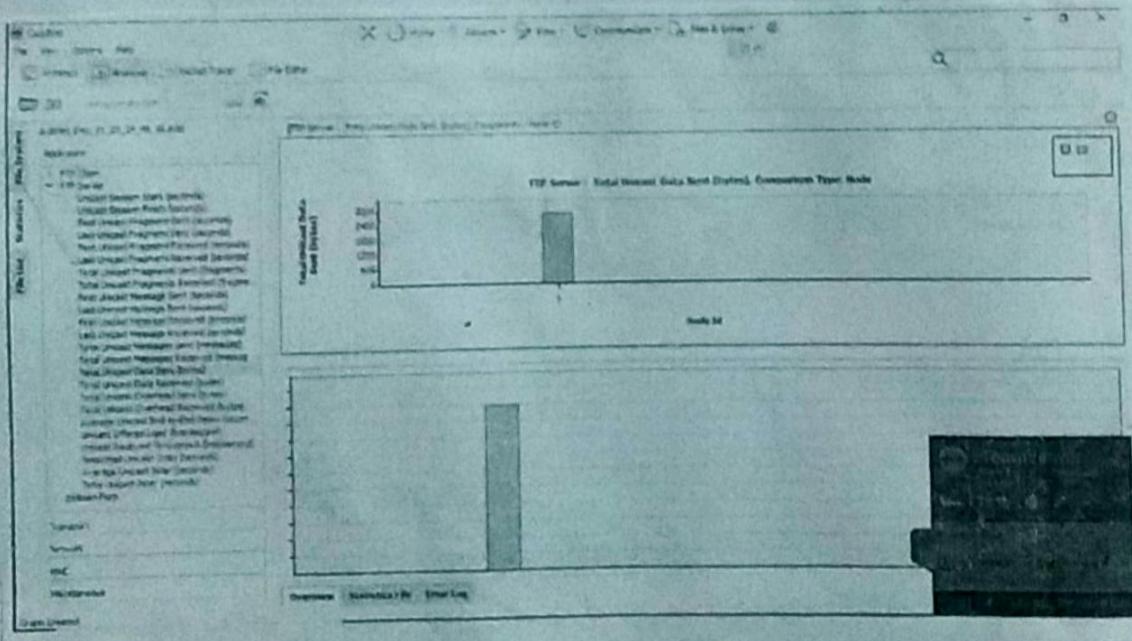


Fig 2

Part-B

- 1) Set up an IEEE 802.3 network with
 2) hub b) switch c) Hierarchy of switch. Apply the FTP, Telnet
 applications between nodes. Vary the number of nodes. Vary the
 bandwidth, queue size and observe the packet drop probability.

Fig 1: Network architecture with 4 devices and a switch.
 Device 1 and Device 3 exchange packets using FTP.

Fig 2: Total Unicast data sent (in bytes) by FTP server.

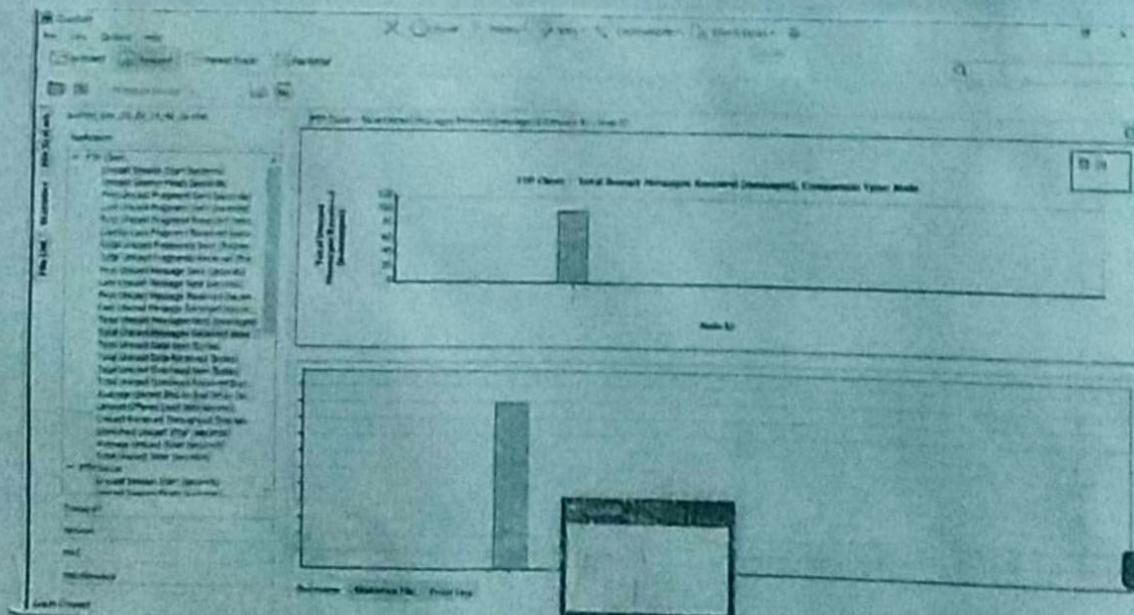
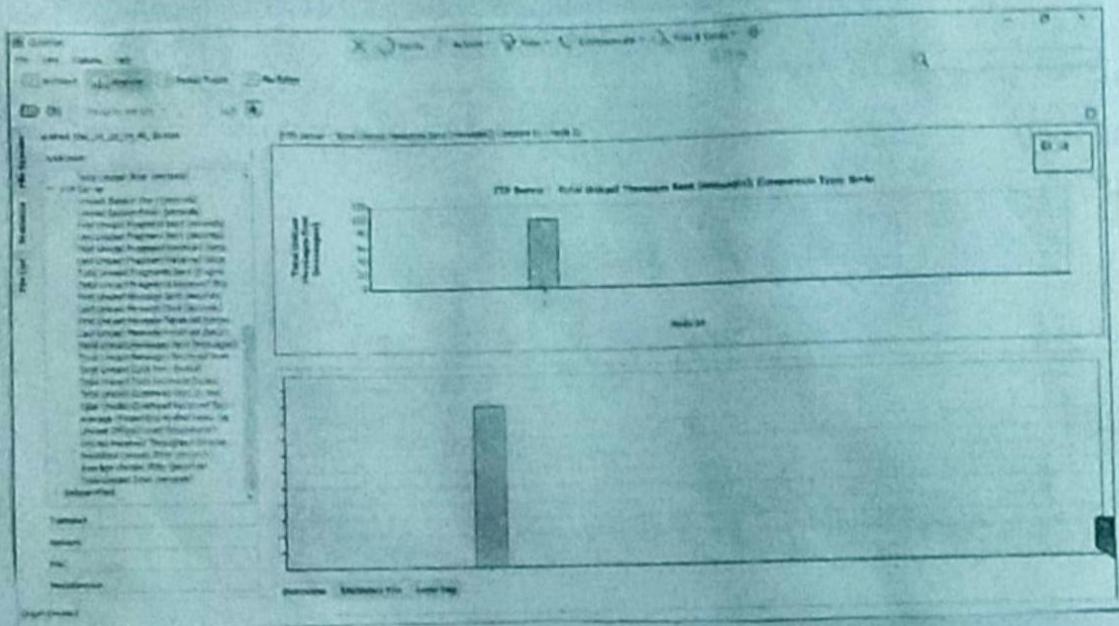


Fig3 : Total Unicast messages sent by FTP Server

Fig4 : Total Unicast messages received by TCP Client

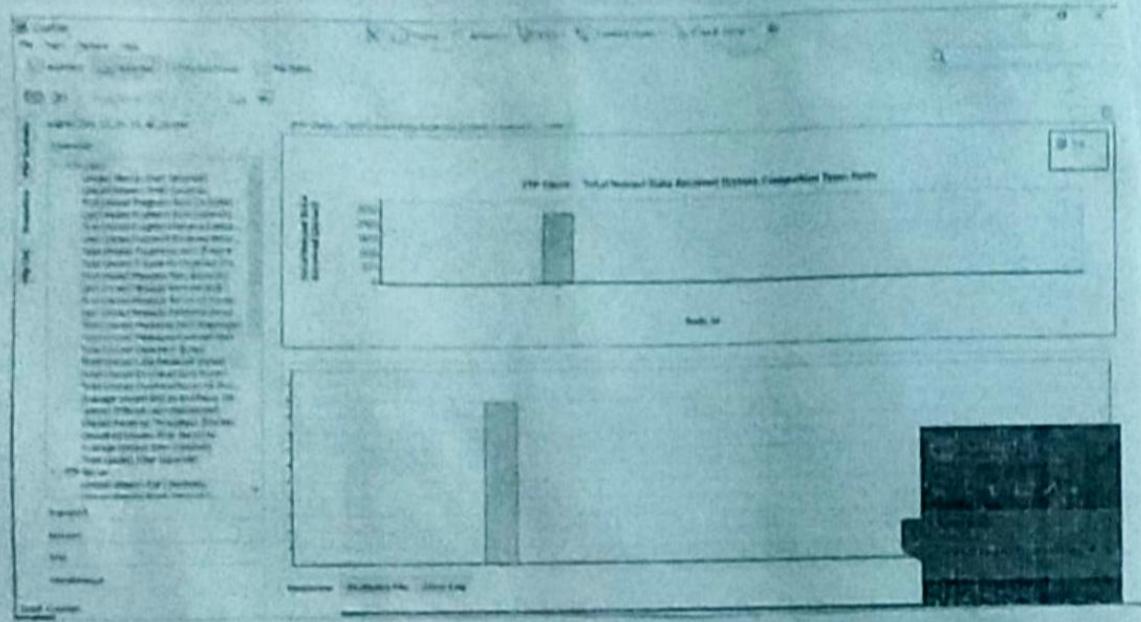


Fig 5: Total Unicast data Received (in bytes) by FTP Client.

Part B - 1b

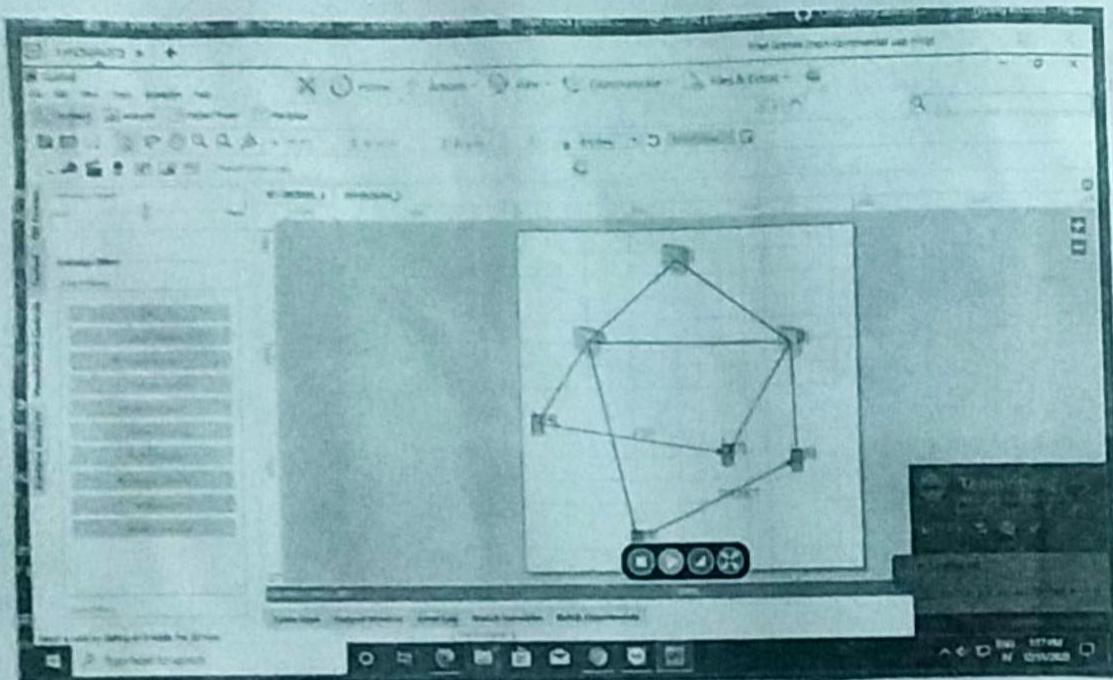


Fig1

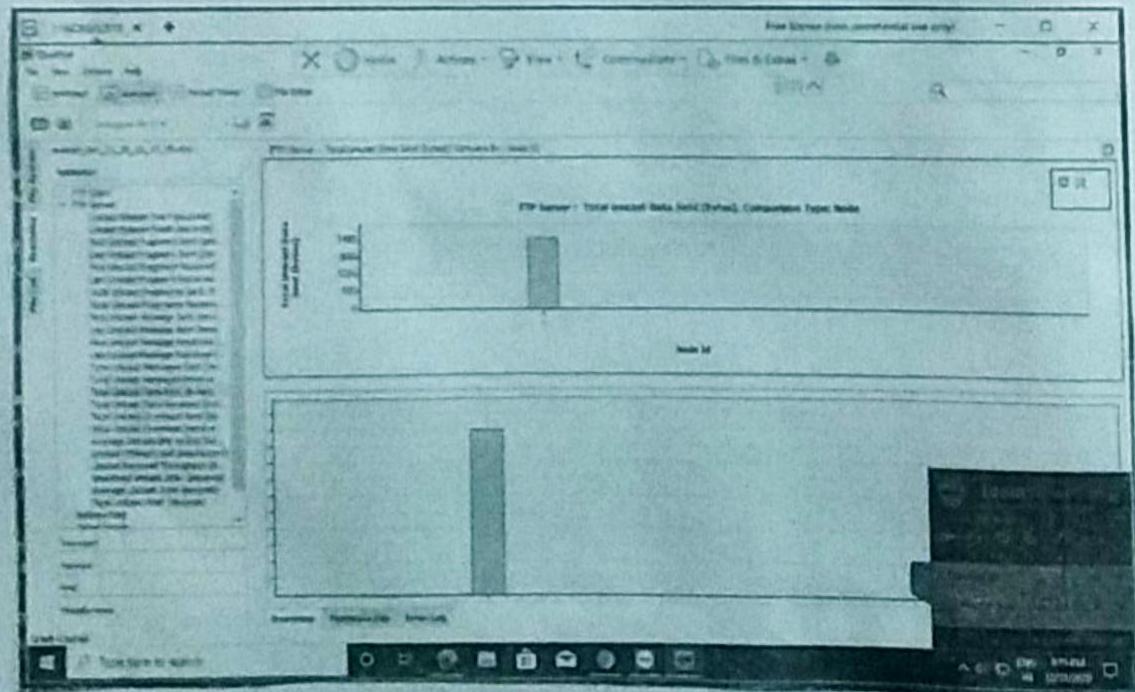


Fig2

(b)

Fig 1: Network hierarchy with

Fig 1: Network architecture with two hierarchy of switch and devices. There is FTP communication between devices 4 and 7 and TELNET communication between devices 5 and 6.

Fig 2: Total unique data (in bytes) sent by FTP server.

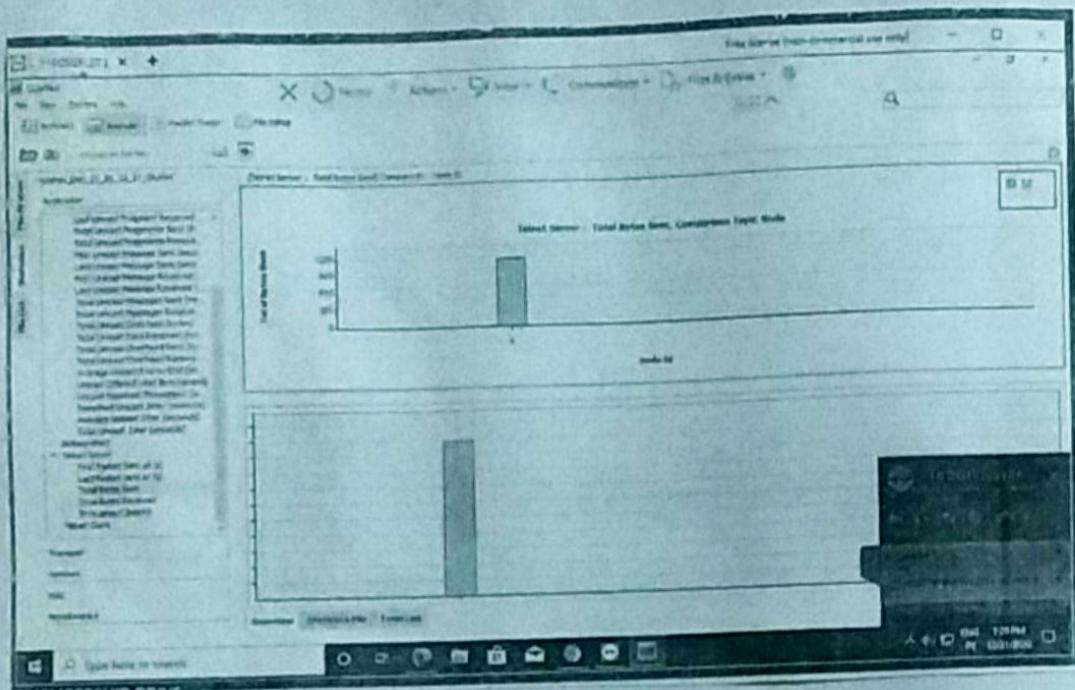


Fig3: Total bytes sent by Telnet server

PartB-2

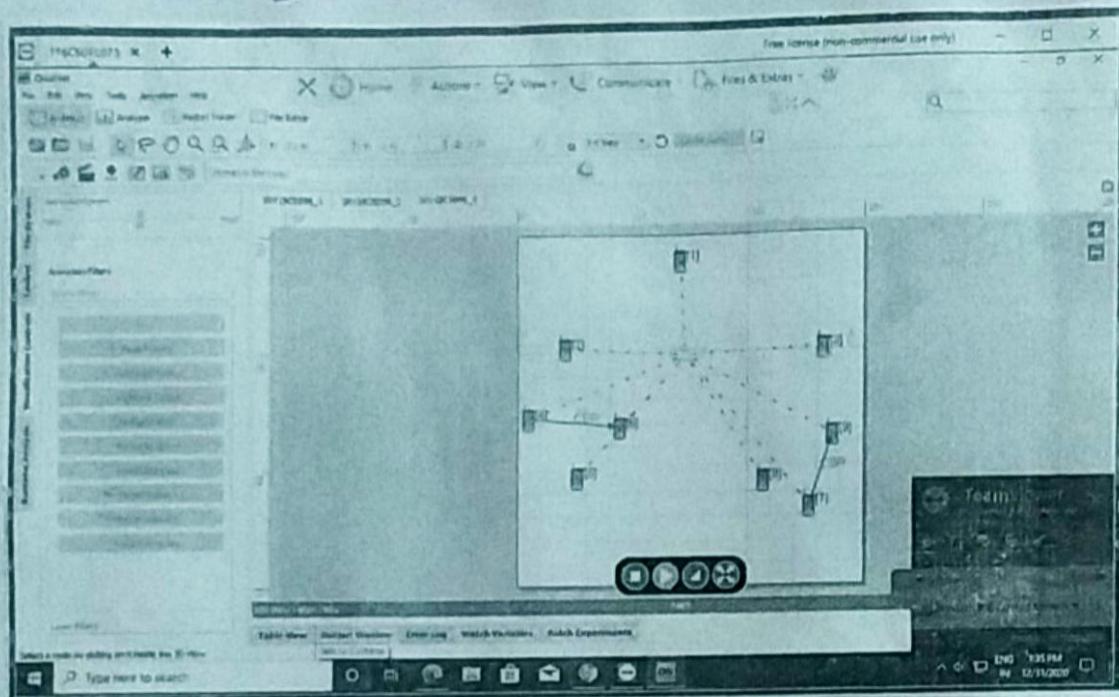


Fig1

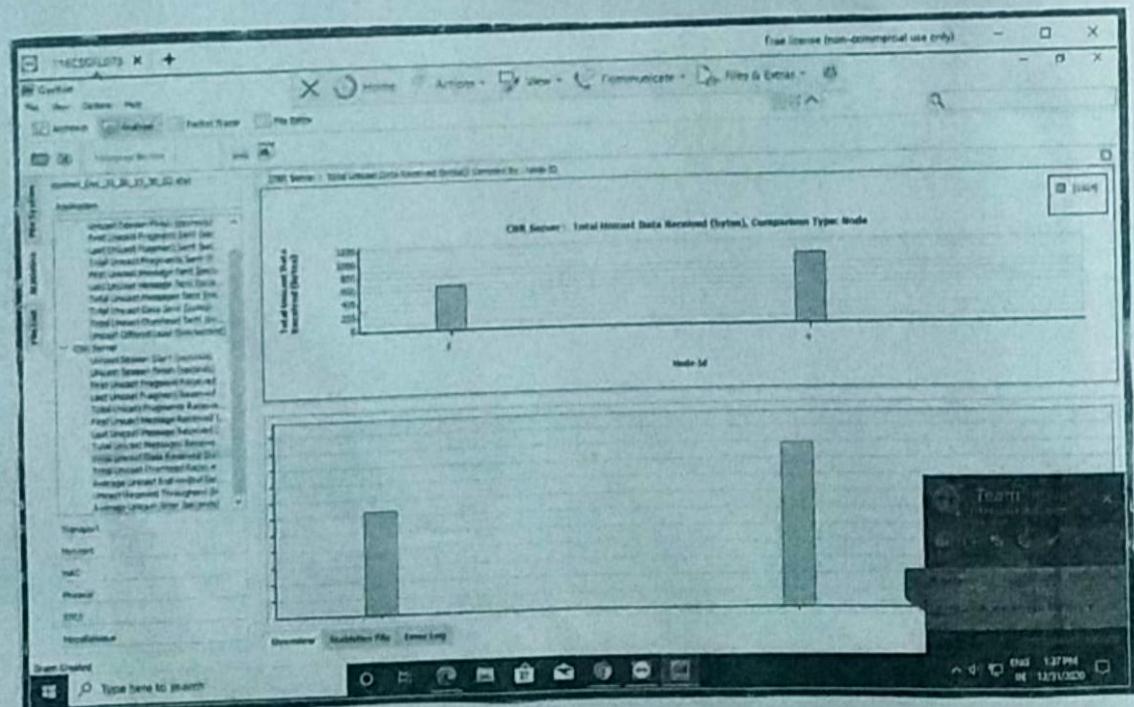


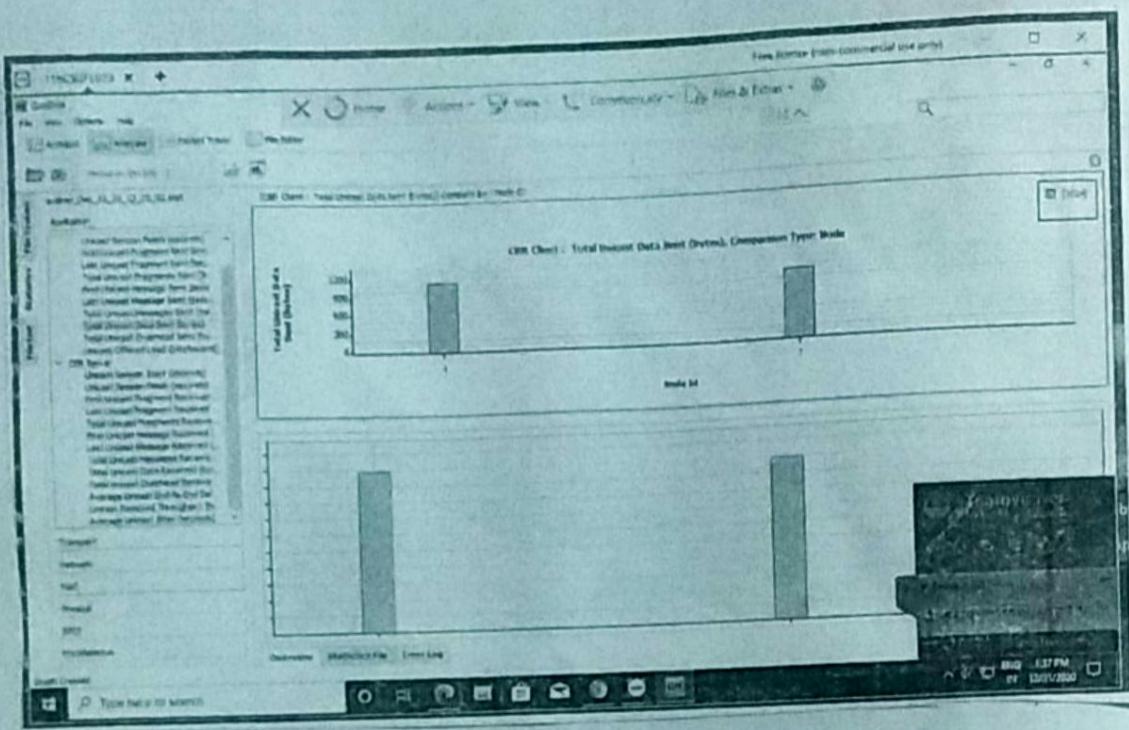
Fig2

2) Setup a wireless mesh network with atleast two device coordinators and nodes. Provide Constant Bit Rate(CBR), Variable Bit Rate(VBR) application between several nodes. Increase the number of coordinators and nodes in the same area and observe the performance at physical and MAC layers.

Fig1 : network architecture of wireless LAN. Here device 1 is the coordinator, devices 2 and 3 are PAN coordinator

Fig2 : Total Unicast data Received(Byts) by CBR server.

Fig 3:



Expt. No. _____

Date _____

Page No. _____

Fig 3: Total Unicast data sent (bytes) by CBR client

Teacher's Signature _____

Part B - 3

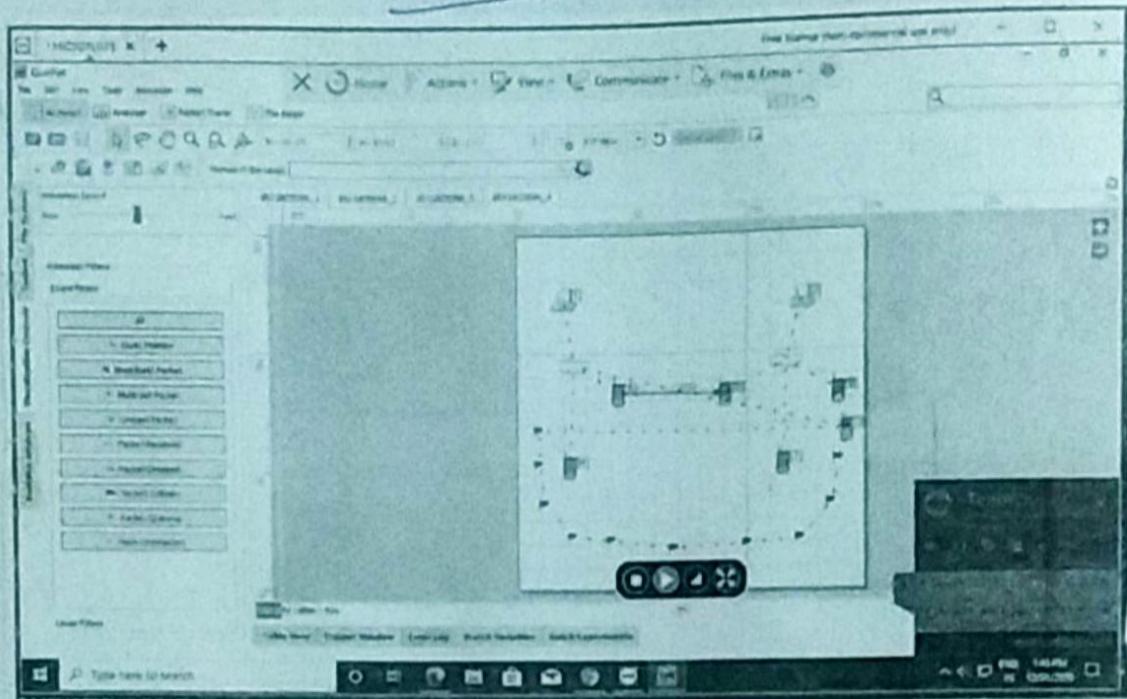


Fig 1

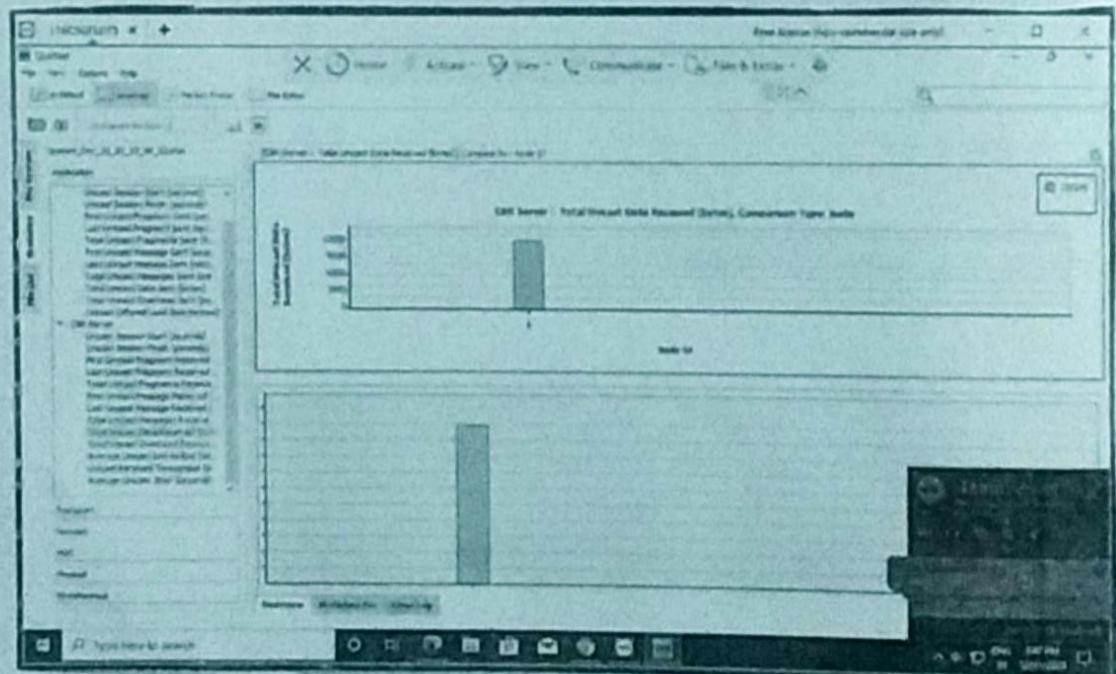


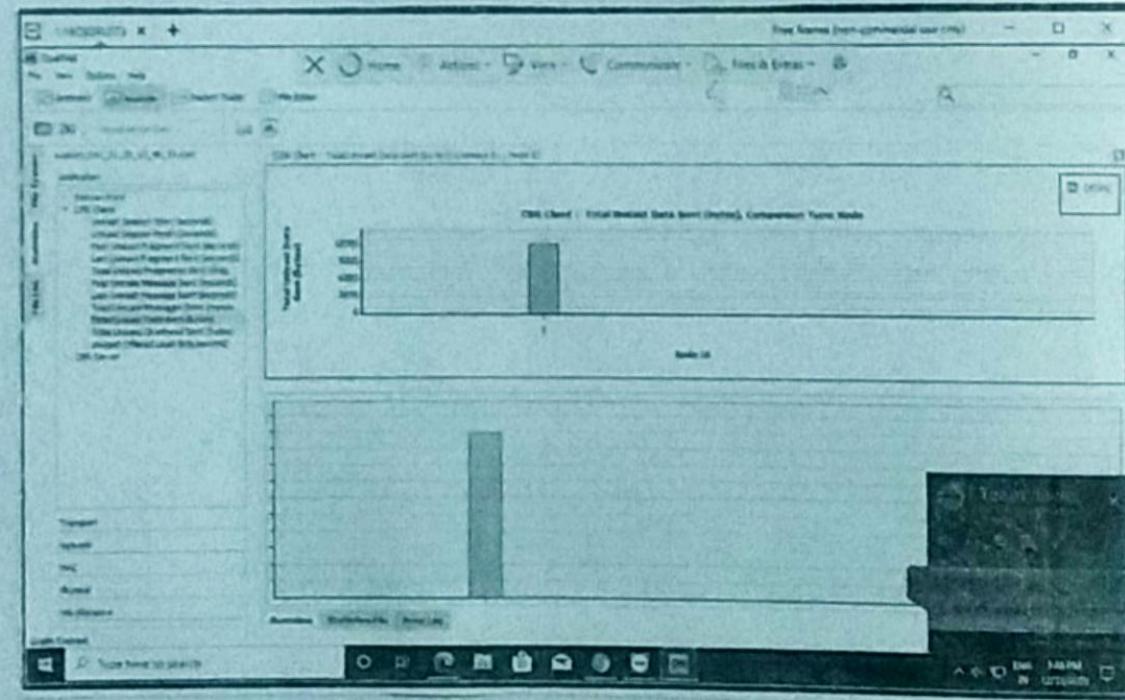
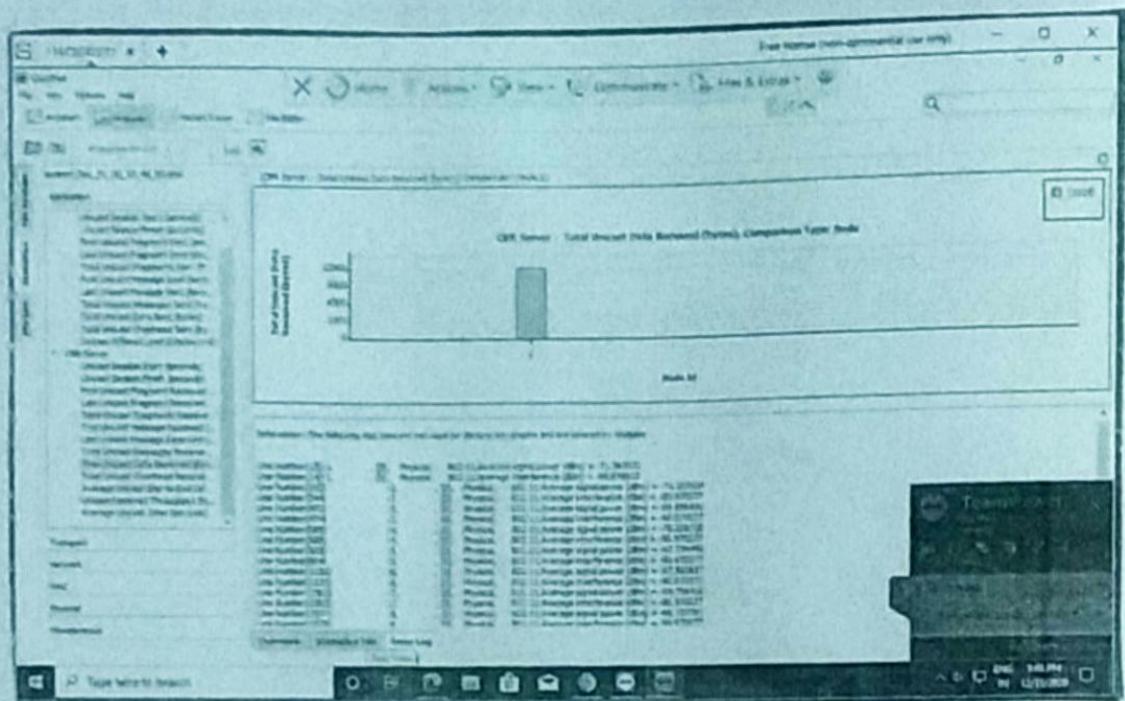
Fig 2

③ setup an IEEE 802.11 network with at least two access points.

Apply the CBR, VBR applications between devices belonging to same access points and different access points. Periodic scanning of any device, vary the numbers of access points and devices. Find out the delay in MAC layer, packet drop probability

Fig1: Network architecture of IEEE 802.11 wireless LAN. It has two access points

Fig2: Total unicast data received (in bytes) by CBR server.



Expt. No. _____

Date _____

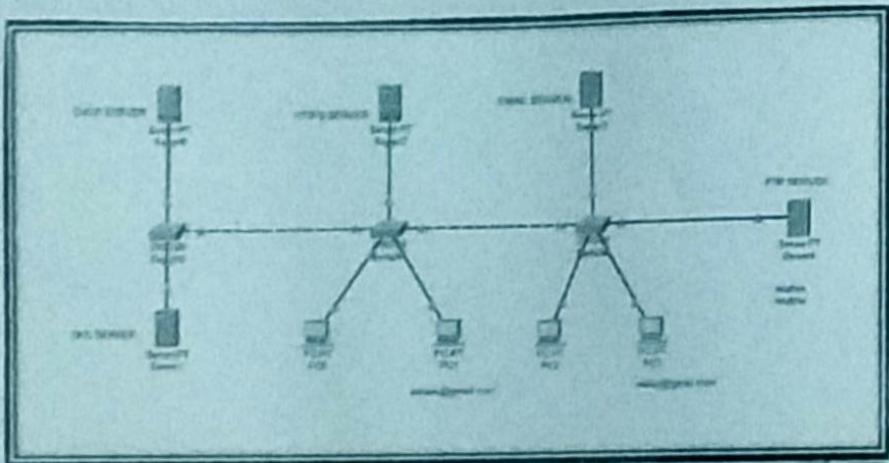
Page No. _____

Fig3: Total Client Data Received (bytes) by CBR server

Fig4: Total Client Data Sent (bytes) by CBR client

Teacher's Signature _____

Topology A:



commands used:

DHCP: subnetMask
offault gateway

DNS: Domain name
IP Address

Email: Domain username, password
Incoming and outgoing mail servers

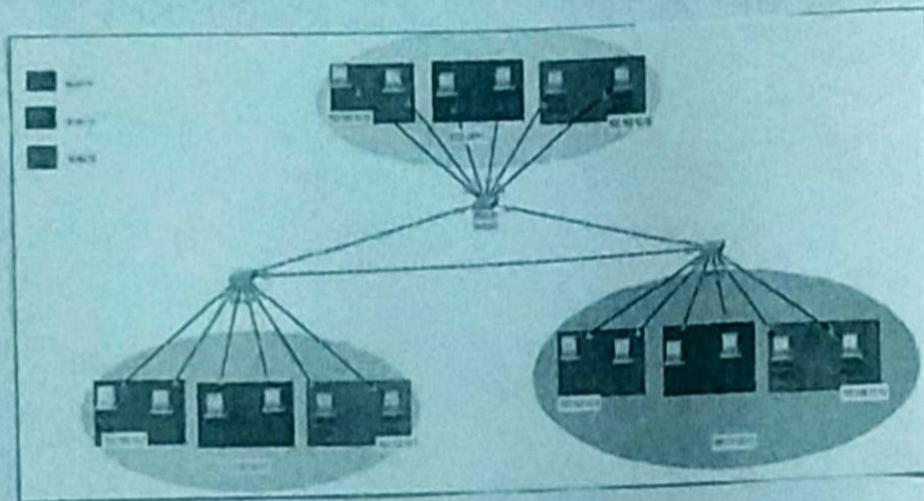
FTP: with ftp.com
ftp>do

4) We study on configuring switches using Cisco packet tracer

Inter-VLAN communication

VLAN is a subnetwork which can group together collections of devices on separate physical local area network (LANs). A LAN is a group of computers and devices and share a communication line or wireless link to a server within the same geographical area.

Topology B



commands used

switch>enable

switch# configure terminal

Enter configuration commands, one per line. End with control-Z

switch(config)# vlan 10

switch(config-vlan)# name V10

switch(config-vlan)# exit

switch(config)# vlan 11

switch(config-vlan)# name V11

switch(config)# vlan 12

switch(config-vlan)# name V12

switch(config)# int fast 0/1

switch(config-if)# switch m a

switch(config-if)# switch a vlan 10

switch(config)# int fa 0/2

switch(config-if)# switch a vlan 11

switch(config-if)# int fa 0/3

switch(config-if)# switch a vlan 12

switch(config-if)# int fast 0/7

switch(config-if)# int fast 0/7 mt

switch(config)# int fa 0/7

switch(config-if)# switch mt

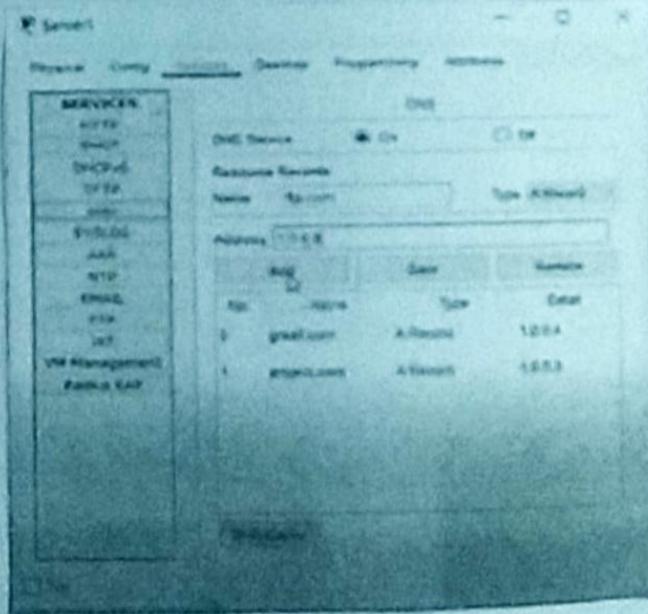
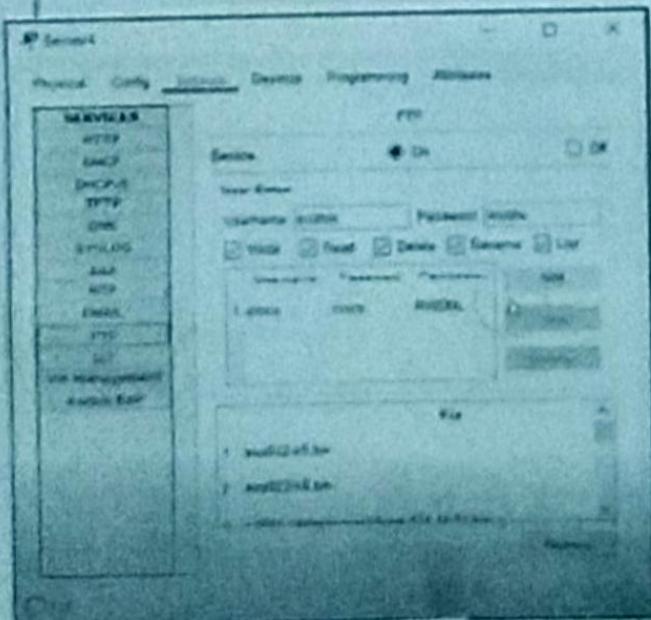
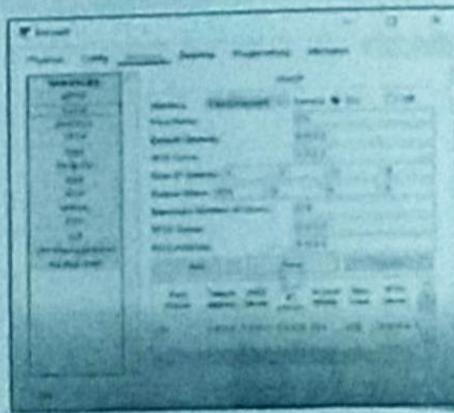
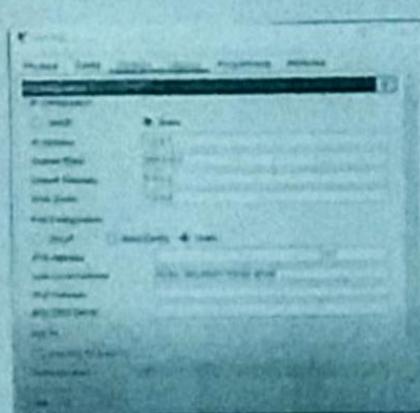
* LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/7, changed state to down

* LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/7, changed to state up

switch(config-if)# switch negotiate

screenshots of simulation

OHCP



ONS

FTP

HTTP

EMAIL

SonicWALL

Services

Services

- HTTP
- DHCP
- DNS/NS
- TFTP
- DAS
- SSL/SNMP
- AAA
- ADNS
- Radius
- LDAP
- iSCSI

DNS

DHCP Server **ICMP Server**

On Off On Off

Domain Name: **Search:**

User Status

User: **Comments:**

Account: **Address:**

| Profile | | Cart | | Orders | | Reporting | | Analytics | |
|-----------------------|---------------------|--------------|--|--------------|--|-----------|--|-----------|--|
| Customer Info: | | | | | | | | | |
| User Information: | | | | | | | | | |
| User Name: | admin | | | | | | | | |
| Email Address: | admin@mynewsite.com | | | | | | | | |
| Social Information: | | | | | | | | | |
| Facebook User Name: | greatman | | | | | | | | |
| Google+ User Name: | greatman | | | | | | | | |
| Super Admin Info: | | | | | | | | | |
| User Name: | admin | | | | | | | | |
| Password: | ***** | | | | | | | | |
| Save | | Clear | | Reset | | | | | |

卷之三

| File | Last Status | Source | Deployment | Type | Color | Timestamp | Periodic | Run | Exit |
|------|-------------|--------|------------|------|-------|----------------------|----------|-----|-------|
| 1 | Successful | PC1 | P01 | XP1 | Green | 2023-01-01T00:00:00Z | N | 0 | (000) |
| 2 | Successful | PC2 | P02 | XP2 | Green | 2023-01-01T00:00:00Z | N | 1 | (001) |
| 3 | Failed | PC3 | P03 | XP3 | Red | 2023-01-01T00:00:00Z | N | 2 | (002) |
| 4 | Successful | PC4 | P04 | XP4 | Green | 2023-01-01T00:00:00Z | N | 3 | (003) |
| 5 | Successful | PC5 | P05 | XP5 | Green | 2023-01-01T00:00:00Z | N | 4 | (004) |
| 6 | Failed | PC6 | P06 | XP6 | Red | 2023-01-01T00:00:00Z | N | 5 | (005) |
| 7 | Successful | PC7 | P07 | XP7 | Green | 2023-01-01T00:00:00Z | N | 6 | (006) |

output

| VLAN Name | Status | Ports |
|--|--------|---|
| 1 default | active | Fa0/9, Fa0/10, Fa0/11, Fa0/12 Fa0/13, Fa0/14, Fa0/15, Fa0/16 Fa0/17, Fa0/18, Fa0/19, Fa0/20 Fa0/21, Fa0/22, Fa0/23, Fa0/24 |
| 10 V10 | active | Fa0/1, Fa0/6 |
| 11 V11 | active | Fa0/2, Fa0/8 |
| 12 V12 | active | Fa0/3, Fa0/4 |
| 1002 fddi-default | active | |
| 1008 token-ring-default | active | |
| 1004 fddinet-default | active | |
| 1005 trnet-default | active | |
| VLAN Type SAID MTU Parent RingNo BridgeNo Isp BridgeMode Transl Trans2 | | |
| 1 enet 1000G1 1500 - - - - - - | | 0 0 |
| 10 enet 100G10 1500 - - - - - - | | 0 0 |
| 11 enet 100G11 1500 - - - - - - | | 0 0 |
| 12 enet 100G12 1500 - - - - - - | | 0 0 |
| 1002 fddi 101G01 1500 - - - - - - | | 0 0 |

Switch#

```
Windows Command Prompt

Ping statistics for 192.168.10.71
Source: Using interface Local Area Connection 1
Destination: 192.168.10.71
Bytes = 32 bytes of data.

Reply from 192.168.10.71: bytes=32 time=1ms TTL=128

Ping statistics for 192.168.10.71
Transmitting 400 bytes of data.
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 1ms, Average = 1ms

C:\> ping 192.168.10.10

Pinging 192.168.10.10 with 32 bytes of data

Reply from 192.168.10.10: bytes=32 time=1ms TTL=128

Ping statistics for 192.168.10.10
    Minimum = 1ms, Maximum = 1ms, Average = 1ms
```