Date _____

Expt. No. _____ Page No. _____

SAKSHI KAMAL
1RV18CS143
NPS Lab Record

Teacher's Signature _____

Date _____

Expt. No. _____ Page No. _____

PART A

1. Implement a client and server communication using Socket programming.

client.c

```
#include <stdio.h>
#include <sys/socket.h>
#include <arpa/inet.h>
# include <unistd.h>
#include <string.h>
# define PORT 8080

int main( int argc , char const *argv[])
{
    int socket = 0, valread ;
    struct sockaddr_in serv_addr;
    char *hello = "Hello from client";
    char buffer [1024] = {0};
    if ((socket = socket (AF_INET, SOCK_STREAM, 0))<0)
    {
        printf ("\n Socket creation error \n");
        return -1;
    }
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(PORT);
    if (inet_pton (AF_INET , argv[1], &serv_addr.sin_addr)
                                        <=0 )
```

Teacher's Signature _____

Expt. No. _____ Date _____

Page No. _____

```
{
    printf ("\n Invalid Address");
    return -1;
}
if (connet (sock, (struct sockaddr *) &serv_addr,
            sizeof (serv_addr)) < 0)
{
    printf ("\n Connection failed\n"); return -1;
}
send (sock, hello, strlen(hello), 0);
printf ("Message sent to server\n");
valread = read (sock, buffer, 1024);
printf ("%s\n", buffer);
return 0;
}
```

Server.c

```
#include <unistd.h>
#include <stdio.h>
#include <sys/socket.h>
#include <stdlib.h>
#include <netinet.h>
#include <string.h>
#define PORT 8080
int main (int argc, char const *argv[])
{
    int server_fd, new_socket, valread;
    struct sockaddr_in address;
```

Teacher's Signature _____

Date _____

Expt. No. _____ Page No. _____

```
int opt=1; int addrlen = sizeof(address);
Char buffer[1024] = {0};
char *hello = "Hello from the server";
if ((served_fd = socket(AF_INET, SOCK_STREAM, 0))==0)
{
    perror("socket failed"); exit(EXIT_FAILURE);
}
if( setsockopt (server_fd, SOL_SOCKET, SO_REUSEADDR|
    SO_REUSEPORT, &opt, sizeof(opt)))
{
    perror("setsockopt"); exit(EXIT_FAILURE);
}
address.sin_family = AF_INET;
address.sin_addr.s_addr = INADDR_ANY;
address.sin_port = hton s(PORT);
if( bind(server_fd,(struct sockaddr *) &address, sizeof(address))
                                                    <0)
{
    perror("bind failed"); return -1;
}
if(listen(server_fd, 3)<0)
{   perror("listen failed"); exit(EXIT_FAILURE);
}
if(new_socket = accept(server_fd, (struct sockaddr *) &address,
            (socklen_t *) &addrlen)) <0)
{ perror("accept"); exit(EXIT_FAILURE); }
valread = read(new_socket, buffer, 1024);
printf("%s\n", buffer);
send(new_socket, hello, strlen(hello), 0);
printf("Hello message sent");
return 0;
}
```

Teacher's Signature _____

Output :

Terminal 1

```
$ cc server.c -o server
$ ./server
```

Hello from Client
Hello message sent

Terminal 2

```
$ cc client.c -o client
$ ./client 127.0.0.1
```

Message sent to Server
Hello from the Server.

Date _____

Expt. No. _____ Page No. _____

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

dist.c

```
#include<stdio.h>
int A[10][10], n, d[10], p[10];
void BellmanFord(int s)
{
    int i,u,v;
    for(i=1;i<n;i++){
        for(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;
                }
            }
        }
    }
    for(u=0;u<n;u++){
        for(v=0;v<n;v++){
            if(d[v]>d[u]+A[u][v]){
                printf("Negative edge");
            }
        }
    }
}
int main()
{
    printf("Enter the no of vertices:");
    Scanf("%d",&n);
    printf("Enter the adjacency matrix:\n");
```

Teacher's Signature _____

Date _____

Expt. No. _____ Page No. _____

```
int i, j;
for(i=0; i<n; i++) {
        for(j=0; j<n; j++) {
                scanf("%d", &A[i][j]); }}
int source;
for(source=0; source<n; source++){
        for(i=0; i<n; i++) {
                d[i] = 999;
                p[i] = -1;
        }
        d[source] = 0;
        BellmanFord(source);
        printf("Router %d \n", source);
        for(i=0; i<n; i++) {
                if(i != source){
                        j = i;
                        while(p[j] != -1) {
                                printf("%d <- ", j);
                                j = p[j];
                        }
                }
                printf("%d H Cost %d \n\n", sourc, d[i]);
        }
}
return 0;
}
```

Teacher's Signature _____

Output:

```
$ cc dist.c -o dist
$ ./dist
Enter the no. of vertices : 4
Enter the Adjacency Matrix :
0  2  999  1
2  0  3    7
999 3  0   11
1  7  11   0

Router 0
0        Cost 0
1  <-0   Cost 2
2  <-1<-0  Cost 5
3  <-0   Cost 1

Router 1
0 <- 1    Cost 2
1         Cost 0
2 <- 1    Cost 3
3 <- 0 <- 1   Cost 3

Router 2
0 <- 1 <- 2    Cost 5
1 <- 2         Cost 3
2              Cost 0
3 <- 0 <- 1 <- 2    Cost 6

Router 3
0 <- 3    Cost 1
1 <- 0 <- 3    Cost 3
2 <- 1 <- 0 <- 3    Cost 4
3         Cost 0
```

Date

Expt. No. Page No.

7 Implement a simple multicast routing mechanism

listener.c

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet.h>
#include <arpa/inet.h>
#include <time.h>
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#define HELLO_PORT 12345
#define HELLO_GROUP
#define MSGBUFSIZE 25

main(int argc, char *argv[])
{
    struct sockaddr_in addr;
    int fd, nbytes, addrlen;
    struct ip_mreq mreq;
    char msgbuf[MSGBUFSIZE];
    u_int yes = 1;
    if((fd = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
        perror("socket");
        exit(1);
    }
    memset(&addr, 0, sizeof(addr));
    addr.sin_family = AF_INET;
    addr.sin_addr.s_addr = htonl(INADDR_ANY);
    addr.sin_port = htons(HELLO_PORT);
```

Teacher's Signature

```
if(bind(fd,(struct sockaddr *)&addr, sizeof(addr))<0)
{
        perror("Bind");
        exit(1);
}
mreq.imr_multiaddr.s_addr = inet_addr(HELLO_GROUP);
mreq.imr_interface.s_addr = htonl(INADDR_ANY);
if(setsockopt(fd, IPPROTO_IP, IP_ADD_MEMBERSHIP,
        &mreq, sizeof(mreq))<0)
{
        perror("setsockopt");
        exit(1);
}
while(1){
        addrlen = sizeof(addr);
        if((nbytes = recvfrom(fd, msgbuf, MSGBUFSIZE, 0,
                (struct sockaddr *)&addr, &addrlen))<0)
        {
                perror("recvfrom");
                puts(msgbuf);
        }
    }
}
```

Sender.c

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

Date

Expt. No.

Page No.

```
#include <arpa/inet.h>
#include <unix.h>
#include <string.h>
#define HELLO_PORT 12345
#define HELLO_GROUP "225.0.0.37"
main(int argc, char *argv[])
{
    struct sockaddr_in addr;
    int fd, cnt;
    struct ip_mreq mreq;
    char *message = "RVCE-CSE";
    if ((fd = socket(AF_INET, SOCK_DGRAM, 0)) < 0){
        perror("socket");
        exit(1);
    }
    memset(&addr, 0, sizeof(addr));
    addr.sin_family = AF_INET;
    addr.sin_port = htons(HELLO_PORT);
    addr.sin_addr.s_addr = inet_addr(HELLO_GROUP);
    while(1){
        if(sendto(fd, message, sizeof(message), 0, (struct sockaddr *)
            &addr, sizeof(addr)) < 0)
        {
            perror("sendto");
            exit(1);
        }
        sleep(1);
    }
}
```

Teacher's Signature

Output

Terminal 1 - Sender

```
$ cc sender.c -o sender
$ ./sender    // starts sending messages
```

Terminal 2 - Listener 1

```
$ cc listener.c -o listener
$ ./listener
  RVCE-CSE
  RVCE-CSE
  RVCE-CSE
  RVCE-CSE
```

Terminal 3 - Listener 2

```
$ ./listener
  RVCE-CSE
  RVCE-CSE
  RVCE-CSE
  RVCE-CSE
```

Date _____

Expt. No. _____ Page No. _____

5. Write a program to implement concurrent chat server that allows logged in users to communicate with other.

server.c

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet.h>
#include <sys/stat.h>
#include <unistd.h>
#include <stdlib.h>
#include <fcntl.h>
#include <arpa/inet.h>
#include <string.h>
void str_echo(int connfd)
{
    int n;
    int buffsize = 1024
    char *buffer = malloc(bufsize);
    while((n = recv(connfd, buffer, bufsize, 0)) > 0)
    {   fputs("client:", stdout);
        fputs(buffer, stdout);
        fputs("Me:", stdout);
        if(fgets(buffer, bufsize, stdin) != NULL){
            send(connfd, buffer, sizeof(buffer), 0);
        }
    bzero(buffer, 1024);
    }
}
int main() {
    int cond, listenfd, connfd, addrlen, addrlen2, fd, pid,
                                                    addrlen3,
```

Teacher's Signature _____

Date

Expt. No.

Page No.

```
struct sockaddr_in address, cliaddress;
if (( listenfd = socket (AF_INET, SOCK_STREAM, 0)) >0)
    printf ("Socket was created");
address.sin_family = AF_INET;
address.sin_addr.s_addr = INADDR_ANY;
address.sin_port = htons (16001);
printf ("Address for binding %s... \n", inet_ntoa
    (address.sin_addr));
if ( bind ( listenfd, (struct sockaddr *) &address, sizeof
    (address)) == 0 )
        printf ("Binding socket");
printf ("address after binding is %s .. \n", inet_ntoa
    ( address.sin_addr));
listen ( listenfd, 3);
printf ("Server is listening \n");
getsockname ( listenfd, (struct sockaddr *) &address,
        & addrlen3);
printf ("The server's local address %s .... and port %d \n",
        inet_ntoa (address.sin_addr), htons( address.
                sin_port));

for (;;) {
    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 (" The Client is connected ... on port \n");
```

Teacher's Signature

Date _____

Expt. No. _____ Page No. _____

```
if((pid = fork()) == 0)
{
    printf("inside child \n");
    close(listenfd);
    str_echo(connfd);
    exit(0);
}
close(connfd);
return 0;
}
```

client.c

```
#include <sys/types.h>
#include <sys/socket.h>
#include<netinet.h>
#include <unistd.h>
#include <stdlib.h>
#include <arpa/inet.h>
#include <string.h>
void str_cli(FILE *fp, int sockfd)
{
    int buffsize = 1024, cont;
    char *buffer = malloc(bufsize);
    fputs("Me:", stdout);
    while(fgets(buffer, bufsize, fp) != NULL)
    {
        send(sockfd, buffer, sizeof(buffer), 0);
        if((cont = recv(sockfd, buffer, bufsize, 0)) > 0){
            fputs("Server:", stdout);
            fputs(buffer, stdout);
```

Teacher's Signature _____

Date _____

Expt. No. _____ Page No. _____

```
        }
        fputs ("Me:" stdout);
    }
    printf ("\nEOF\n");
}
int main( int argc, char *argv[])
{
    int sock;
    struct sockaddr_in address;
    if ( (sock = socket (AF_INET , SOCK_STREAM, 0 )) >0 )
        printf ("The socket was created");
    address.sin_family = AF_INET;
    address.sin_port = htons(16001);
    inet_pton (AF_INET, argv[1], &address.sin_addr) ;
    if ( connect ( sock, (struct sockaddr *) &address, sizeof
            (address)) == 0 )
        printf ("The connection was accepted by server %s
                                    ...\n", argv[1]);
    else
        printf ("error in connect");
    str_cli (stdin, sock);
    return close (sock);
}
```

Teacher's Signature _____

Output

T1 - server

$ cc server.c -o server

The socket was created
The address before bind 0.0.0.0 ...
The address after bind 0.0.0.0 ...
Server is listening
The server's local address 0.0.0.0 ... and port 16001
The client 127.0.0.1 is connected ... on port 40564
inside child
client : hi c1 Me: hi s
The client 127.0.0.1 is connected ... on port 40568
inside child
client : hi c2
Me: client : hello c1 Me: bye c2
bye c1

T2 - client 1

$ cc client.c -o client
$ ./client 127.0.0.1
The socket was created
Me: hi c1
Server : hi s
Me : hello c1
Server: bye c1

T3 - client 2

$ ./client 127.0.0.1

The socket was created

The connection was accepted with the server 127.0.0.1...

Me : hi C2

Server : bye C2

Date _____

Expt. No. _____ Page No. _____

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

connection - oriented - ~~iterative~~ concurrent

~~echo_server_concurrent.c~~ client.c (Tcp client)

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

void str_cli(FILE *fp, int sockfd)
{
    int bufsize = 1024, cont;
    char *buffer = malloc(bufsize);
    while (fgets(buffer, bufsize, fp) != NULL)
    {
        send(sockfd, buffer, sizeof(buffer), 0);
        if ((cont = recv(sockfd, buffer, bufsize, 0)) > 0)
        {
            fputs(buffer, stdout);
        }
    }
    printf("\nEOF\n");
}
int main(int argc, char *argv[])
{
    int sock;
```

Teacher's Signature _____

Date _____

Expt. No. _____ Page No. _____

```
struct sockaddr_in address;
if ((sock = socket (AF_INET, SOCK_STREAM, 0)) > 0)
        printf ("The Socket was created \n");
address.sin_family = AF_INET;
address.sin_port = htons(15001);
inet_pton (AF_INET, argv[1], &address.sin_addr);
if (connect (sock, (struct sockaddr *) &address, sizeof
    (address)) == 0)
        printf ("The connection was accepted with server
                %s .... \n", argv[1]);
else
        printf ("error in connects \n");

str_cli (stdin, sock);
return close (sock);
}
```

echo_server_concurrent.c (TCP concurrent)

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet.h>
#include <arpa/inet.h>
#include <stdlib.h>
#include <stdio.h>
#include <fcntl.h>
void str_echo (int connfd)
{
    int n;
    int bufsize = 1024;
    char *buffer = malloc(bufsize);
```

Teacher's Signature _____

Date _____

Expt. No. _____ Page No. _____

```
again : while ((n = recv( connfd, buffer, bufsize, 0))>0)
            send (connfd, buffer, n, 0);
            if (n<0)
                goto again;
}
int main()
{
    int const, listenfd, connfd, addrlen, addrlen2, fd, pid,
    addrlen3;
    struct sockaddr_in address, cli_address;
    if ((listenfd = socket (AF_INET, SOCK_STREAM, 0)) > 0)
        printf ("The Socket was created");
    address.sin_family = AF_INET;
    address.sin_addr.s_addr = INADDR_ANY;
    printf ("address before bind %s ... \n", inet_ntoa
            (address.sin_addr));
    if( bind (listenfd, (struct sockaddr *) &address, sizeof
            (address)) == 0)
            printf ("Binding Socket");
    printf (" The address after binding %s ... \n", inet_ntoa
            (address.sin_addr));
    listen (listenfd, 3);
    printf ("server is listening \n");
    getsockname (listenfd, (struct sockaddr *) &address, &addrlen3);
    printf (" The server's local address %s and port %d \n",
      inet_ntoa(address.sin_addr), htons (address.sin_port));
    for (;;)
    {
        addrlen = sizeof (struct sockaddr_in);
```

Teacher's Signature _____

Date _____

Expt. No. _____ Page No. _____

```
connfd = accept (listenfd, (struct sockaddr *) &cliaddress,
                 &addrlen);
addrlen2 = sizeof (struct sockaddr_in);
int i = getpeername (connfd, (struct sockaddr *) &cliaddress,
                     &addrlen);
printf ("The Client %s is connected ... on port %d\n",
    inet_ntoa (cli_address.sin_addr), htons (cliaddress.
        sin_port));
if ((pid = fork ()) == 0)
{
        printf ("inside child");
        close (listenfd);
        str_echo (connfd);
        exit (0);
}
close (connfd);
return 0;
}
```

echo_server_iterative.c (TCP iterative)

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <arpa/inet.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet.h>
#include <errno.h>
#define SERV_PORT 9002
```

Teacher's Signature _____

Output 1 (TCP concurrent)

T1 - TCP server

$ ./echo_server

The Socket was created

The address before bind 0.0.0.0 ...

Binding socket

The address after bind 0.0.0.0 ...

Server is listening

The server's local address 0.0.0.0 ... and port 15001

The client 127.0.0.1 is connected ... on port 35226

inside child

The client 127.0.0.2 is connected ... on port 50536

inside child

T2 - TCP Client 1

$ ./echo_client 127.0.0.1

The socket was created

The connection was accepted with server 127.0.0.1 ...

hello

hello

fine

fine

T3 - TCP Client 2

./echo-client 127.0.0.1

The Socket was created

The connection was accepted with server 127.0.0.1

byee

byee

okay

okay

Date

Expt. No. _____ Page No. _____

```
void str_cli (FILE *fp, int sockfd)
{
    char sendline[MAXLINE], recvline[MAXLINE];
    while (fgets (sendline, MAXLINE, fp))
    {
        write (fgets (sendline, MAXLINE, fp))
        if (read (sockfd, recvline, MAXLINE) == 0)
        {
            perror ("str_cli: server terminated prematurely");
            exit (EXIT_FAILURE)
        }
        fputs (recvline, stdout);
    }
}
int main (int argc, char *argv)
{
    int sockfd;
    struct sock_addr_in servaddr;
    if (argc != 2)
    {
        perror ("usage : tcp< IPaddress");
        exit (EXIT_FAILURE);
    }
    sockfd = socket (AF_INET, SOCK_STREAM, 0);
    bzero (&servaddr, sizeof (servaddr));
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s
    servaddr.sin_port = htons (SERV_PORT);
    inet_pton (AF_INET, argv[1], &servaddr.sin_addr);
```

Teacher's Signature _____

Output (TCP iterative)

T1 - server

$ ./server_it_tcp

T2 - client_tcp

$ ./client_tcp 127.0.0.1

hello
hello
bye
bye

```
    connect ( sockfd , (struct sockaddr *) , &servaddr ,
            sizeof (servaddr));
    str_cli ( stdin , sockfd);
    exit(0);
}
```

server_udp.c (UDP server - iterative)

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

void str_echo( int sockfd, struct sockaddr * cli_addrlen,
        int clilen)
{
    int n, bufsize = 1024;
    char *buffer = malloc (bufsize);
    int addrlen;
    for( ; ; )
    {
        addrlen = clilen;
        n = recvfrom( sockfd, buffer, bufsize, 0, cliaddrlen,
            &addrlen);
        sendto ( sockfd, buffer, n, 0, cli_addrlen, addrlen);
    }
}
```

Date _____

Expt. No. _____ Page No. _____

```
int main()
{
    int sockfd;
    struct sockaddr_in serv_address, cli_address;
    if((sockfd = socket(AF_INET, SOCK_DGRAM, 0))>0)
        printf("The Socket was created \n");
    serv_address.sin_family = AF_INET;
    serv_address.sin_addr.s_addr = INADDR_ANY;
    serv_address.sin_port = htons(16001);
    printf("The address before bind %s ... \n",
        inet_ntoa(serv_address.sin_addr));
    if(bind(sockfd, (struct sockaddr *) & serv_address,
        sizeof(serv_address)) == 0)
        printf("Binding Socket");
    str_echo(sockfd, (struct sockaddr *) & cli_address,
        sizeof(cli_address)) == 0)
    return 0;
}
```

client_udp.c (UDP - client)

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

void str_cli(FILE *fp, int sockfd, struct sockaddr *
    serv_address, int servlen);
{
```

Teacher's Signature _____

Date _____

Expt. No. _____ Page No. _____

```
int bufsize = 1024, cont;
char *buffer = malloc (bufsize);
int addrlen = sizeof (struct sockaddr_in);
int len = sizeof (struct sockaddr_in);
struct sockaddr_in *preply_addr;
while (fgets (buffer, bufsize, fp) != NULL) {
    sendto (sockfd, buffer, sizeof (buffers), 0,
        serv_address, servlen);
    if ((cond = recvfrom (sockfd, buffer, bufsize,
        0, (struct sockaddr *) preply_addr,
            &len) >0))
    {
        printf ("The address is %s \n", inet_
            ntoa (preply_addr -> sin_addr));
        fputs (buffer, stdout);
    }
}}
int main (int argc, char *argv[])
{
    int sockfd;
    struct sock_addr_in serv_address;
    if ((sockfd = socket (AF_INET, SOCK_DGRAM, 0))
        printf ("Socket created");
                                        >0)
    serv_address.sin_family = AF_INET;
    serv_address.sin_port = htons (16001);
    inet_pton (AF_INET, argv[1], &serv_address.sin_
                                        addr);

    str_cli (stdin, sockfd, (struct sockaddr *) &serv_address,
        sizeof (serv_address));
    exit(0);
}
```

Teacher's Signature _____

Output (UDP)

T1 - UDP server → $ ./server_udp

The socket was created

The address before bind 0.0.0.0 ...

Binding socket ...

T2 - UDP client

$ client-udp 127.0.0.1

The socket was created

hello

The address is 127.0.0.1 ...

hello

nice

The address is 127.0.0.1 ...

nice

Date _____

Expt. No. _____ Page No. _____

7. Implementation of Remote Command Execution using Socket system calls.

server.c

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

int main()
{
    int cond, create_socket, new_socket, addrlen, fd;
    int bufsize = 1024;
    char *buffer = malloc(bufsize);
    Char recieve[256];
    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_addr.s_addr = INADDR_ANY;
    address.sin_port = htons(15000);
    if (bind(create_socket, (struct sockaddr *)&address,
        sizeof(address)) == 0)
```

Teacher's Signature _____

Date _____

Expt. No. _____ Page No. _____

```
        printf ("Binding Socket \n");
listen (create_socket,3);
addrlen = sizeof (struct sockaddr_in);
new_socket = accept (create_socket, (struct
    sockaddr *) &address, &addrlen);
if (new_socked >0)
        printf ("The Client %s is connected ... \n",
                inet_ntoa (address.sin_addr));
int bytes;
while ((bytes = recv (new_socket, recier, 255, 0)) >0)
{
        recier [bytes] = '\0';
        if (strcmp (recier, "end") == 0)
        {
            close (new_socket);
            close (create_socket);
            exit (0);
        }
        else
        {
            printf ("Request received for command: %s\n",
                    recier);
            system (recier);
        }
    }
    close (new_socket);
    return close (create_socket);
}
```

Teacher's Signature

Date _____

Expt. No. _____ Page No. _____

client.c

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

int main (int argc, char *argv[]){
{
    int create_socket, bufsize = 1024;
    char *bufsize = 1024;
    char com[256];
    struct sockaddr_in address;
    if ((create_socket = socket(AF_INET, SOCK_STREAM,
        0)) > 0)
        printf ("The Socket was created");
    address.sin_family = AF_INET;
    address.sin_port = htons(15000);
    inet_pton(AF_INET, argv[1], &address.sin_address);
    if (connect (create_socket, (struct sockaddr *) &address,
        sizeof (address)) == 0)
        printf ("Connection accepted with server \n");
    while(1) {
        printf ("Enter command \n"); scanf ("%s", com);
        send (create_socket, com, sizeof (com), 0);
    }
    printf ["\nEOF\n");
    return close (create_socket);
}
```

Teacher's Signature _____

Output:

T1 - Server

$ ./server

Socket was created

Binding Socket

The client 127.0.0.1 is Connected ...

Request received for command : ls

client client.c server server.c

Request received for command : cal

'Snaps of Output'

January 2021

Su	Mo	Tu	We	Th	Fr	Sa
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

T2 - Client

$ ./client 127.0.0.1

The Socket was created

The connection was accepted with the server 127.0.0.1 ...

Enter command:

ls

Enter command:

cal

Date _____

Expt. No. _____ Page No. _____

8. Write a program to encrypt and decrypt the data using RSA and Exchange the key securely using Diffie-Hellman key Exchange protocol.

diffie.c

```
#include <stdio.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) pow(a,b)) % P);
}
int main()
{
    long long int P, G, x, a, y, b, Ka, Kb;
    P = 23;
    printf("The value of P: %lld\n", P);
    G = 9;
    printf("The value of G: %lld\n\n", G);

    a = 4;
    printf("The private key a for Alice : %lld\n", a);
    x = power(G, a, P);
    b = 3
    printf("The private key b for Bob : %lld\n", b);
    y = power(b, b, P);
```

Teacher's Signature _____

Date _____

Expt. No. _____ Page No. _____

```
//Generating secret key after exchange
ka = power( y, a, p);
kb = power( x, b, p);
printf("Secret key for Alice is: %lld \n", ka);
printf("Secret key for Bob is: %lld \n", kb);
return 0;
}
```

rsa.c

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

long int 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(int i=2; i<a; i++)
    {
        if(a%i==0)
            return 0;
    }
    return 1;
}
```

Teacher's Signature _____

Date _____

Expt. No. _____ Page No. _____

```
int main()
{
    long int i;
    int len;
    long int p, q, n, phi, e, d, cipher[50];
    char text[50];
    printf("Enter the text to be encrypted: ");
    scanf("%s", text);
    len = strlen(text);
    do {
        p = rand() % 30;
    } while(!isprime(p));
    do {
        q = rand()%30;
    } while(!isprime(q));
    n = p*q;
    printf("Two prime numbers (p and q) are:
        %ld and %ld\n", p, q);
    printf("n (p*q) = %ld * %ld = %ld", p, q, p*q);
    printf("(p-1)*(q-1) = %ld\n", phi);
    printf("Public key (n,e): (%ld, %ld)\n", n, e);
    printf("Private key (n,d): (%ld,%ld)\n", n, d);
    for(i=0; i<len; i++)
        cipher[i] = encrypt(text[i], n, e);
    printf("Encrypted message: ");
    for(i=0; i<len; i++)
        printf("%ld
        text[i] = decrypt(cipher[i], n, d);
    printf("Decrypted message: ");
```

Teacher's Signature _____

Date _____

Expt. No. _____ Page No. _____

```
    for(i=0; i<len; i++)
        printf("%c ", text[i]);
    return 0;
}
long int encrypt(long int ch, long int n, long int e)
{
    long int i, temp = ch;
    for(i=0; i<e; i++)
        temp = (temp * ch) % n;
    return temp;
}
long int decrypt(long int ch, long int n, long int d)
{
    long int i, temp = ch;
    for(i=1; i<d; i++)
        temp = (temp * ch) % n;
    return temp;
}
```

Teacher's Signature _____

Output

```
$ ./diffie
The value of P is : 23
The value of G : 9

The private key a for Alice : 4
The private key b for Bob : 3

Secret key for Alice is : 9
Secret key for Bob is : 9
```

Output

$ ./rsa

Enter the text to be encrypted: hi this is Sabeeh

Two prime numbers (p and q) are : 13 and 23

n(p*q) = 13*23 = 299

~~Public key (n, e) = 299,~~

(p-1)*(q-1) = 264

Public key (n, e) : (299, 103)

Private key (n, d) : (299, 223)

Encrypted message : 132489811613248184982481849886111

59184132 48

Decrypted message : hi this is Sabeeh

Date _____

Expt. No. _____ Page No. _____

3. Write a program to implement error detection and correction concept using checksum and hamming code.

checksum.c

```
#include <stdio.h>
Unsigned fields[10];
unsigned short checksum()
{
    int i, 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]);
        while(sum>16)
            sum = (sum & 0xFFFF) + (sum>>16);
    }
    sum = ~sum;
    return (unsigned short) sum;
}
int main()
{
    unsigned short result1, result2;
    result1 = checksum(); //sender
    printf("\n Computed checksum at sender %x\n", result1);
    result2 = checksum(); // reciever
    printf("\n Computed checksum at recieved %x\n", result2);
    if(result1 == result2)
        printf("No error");
    else
```

Teacher's Signature _____

Date _____

Expt. No. _____ Page No. _____

```
        printf ("Error in data received");
    }

hamming.c
#include<stdlib.h>
#include<stdio.h>
int main()
{
    int a[4], b[4], r[3], s[3], i, v[3], c[7];
    printf ("\nEnter 4 bit data word: \n");
    for (i=3; i>=0; i--)
      { scanf ("%d", &a[i]); }
    r[0] = (a[3] + a[1] + a[0])%2;
    r[1] = (a[0] + a[2] + a[3])%2;
    r[2] = (a[1] + a[2] + a[3])%2;
    printf ("\n\n The 7 bit hamming code word: \n");
    for (i=3; i>=0; i--)
        printf ("%d \t", a[i]);
    for (i=2; i>=0; i--)
        printf ("%d \t", r[i]);
    printf ("\n");
    printf ("Enter the 7bit received codeword: ");
    for (i=7; i>0; i--)
        scanf ("%d", &c[i]);
    b[3] = c[7]; b[2] = c[6]; b[1] = c[5]; b[0] = c[4];
    r[2] = c[3]; r[1] = c[2]; r[0] = c[1];
    s[0] = (b[0] + b[1] + b[3] + r[0])%2;
    s[1] = (b[0] + b[2] + b[3] + r[1])%2;
    s[2] = (b[1] + b[2] + b[3] + r[2])%2;
```

Teacher's Signature _____

Date _____

Expt. No. _____ Page No. _____

```
printf("\nSyndrome w : \n");
for(i=2; i>=0; i--)
    printf("%d", s[i]);
if((s[2]==0) && (s[1]==0) && (s[0]==0))
    printf("\n Recieved word is error free \n");
if((s[2]==1) && (s[1]==1) && (s[0]==1))
{
    printf("\n Error in recieved codeword, position
            - 7th bit from right \n");
    if(c[7]==0)
        c[7]=1;
    else
        c[7]=1;
    for(i=7; i>0; i--)
        printf("%d \t", c[i]);
}
if((s[2]==1) && (s[1]==1) && (s[0]==0))
{
    printf("\n Error in recieved Codeword, Position-
            6th bit from right \n");
    if(c[6]==0)
        c[6]=1;
    else
        c[6]=0;
    for(i=7; i>0; i--)
        printf("%d \t", c[i]);
}
if((s[2]==1) && (s[1]==0) && (s[0]==1))
{
```

Teacher's Signature _____

```
printf("\n Error in received codeword, position -
        5th bit from right \n");
c[5] = c[5] == 0 ? 1 : 0;
printf("Corrected codeword is \n");
for(i=7; i>=0; i--){
    printf("%d \t", c[i]);
}
if((s[2]==1)&&(s[1]==0)&&(s[0]==0))
{ printf("Error in received codeword, Position -
        4th bit from right \n");
  c[4] = c[4]==0 ? 1 : 0;
  printf("Corrected codeword is \n");
  for(i=7; i>=0; i--){
      printf("%d \t", c[i]);
}
if((s[2]==0)&&(s[1]==1)&&(s[0]==1))
{
  printf("ERROR in recieved codeword, Position-
      3rd bit from right \n");
  c[3] = c[3]==0 ? 1:0;
  printf("Corrected codewords is \n");
  for(i=7; i>=0; i--)
      printf("%d \t", c[i]);
}
if((s[2]==0)&&(s[1]==1)&&(s[0]==0))
{
  printf("Error in received codeword, Position
      2nd bit from right \n");
  c[2] = c[2]==0 ? 1:0;
```

Date

Expt. No. _____ Page No.

```
        printf ("Corrected codeword is \n");
        for (i=7; i>0; i--)
            printf ("%d", c[i]);
    }
    return (1);
}
```

Teacher's Signature

Output

```
$ ./checksum
Enter IP header information in 16 bit words
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 sender ffd2
Enter IP header information in 16 bit words.
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 receiver ffd2
No error

Output

./hamming
Enter a 4 bit data word :
1 2 3 4
the 7bit hamming code word : 1 2 3 4 1 0
1 0

Enter the 7 bit recieved codeword : 1 2 3 4 0 1 0

Syndrome is :
000
RECIEVED WORD IS ERROR FREE

Date _____

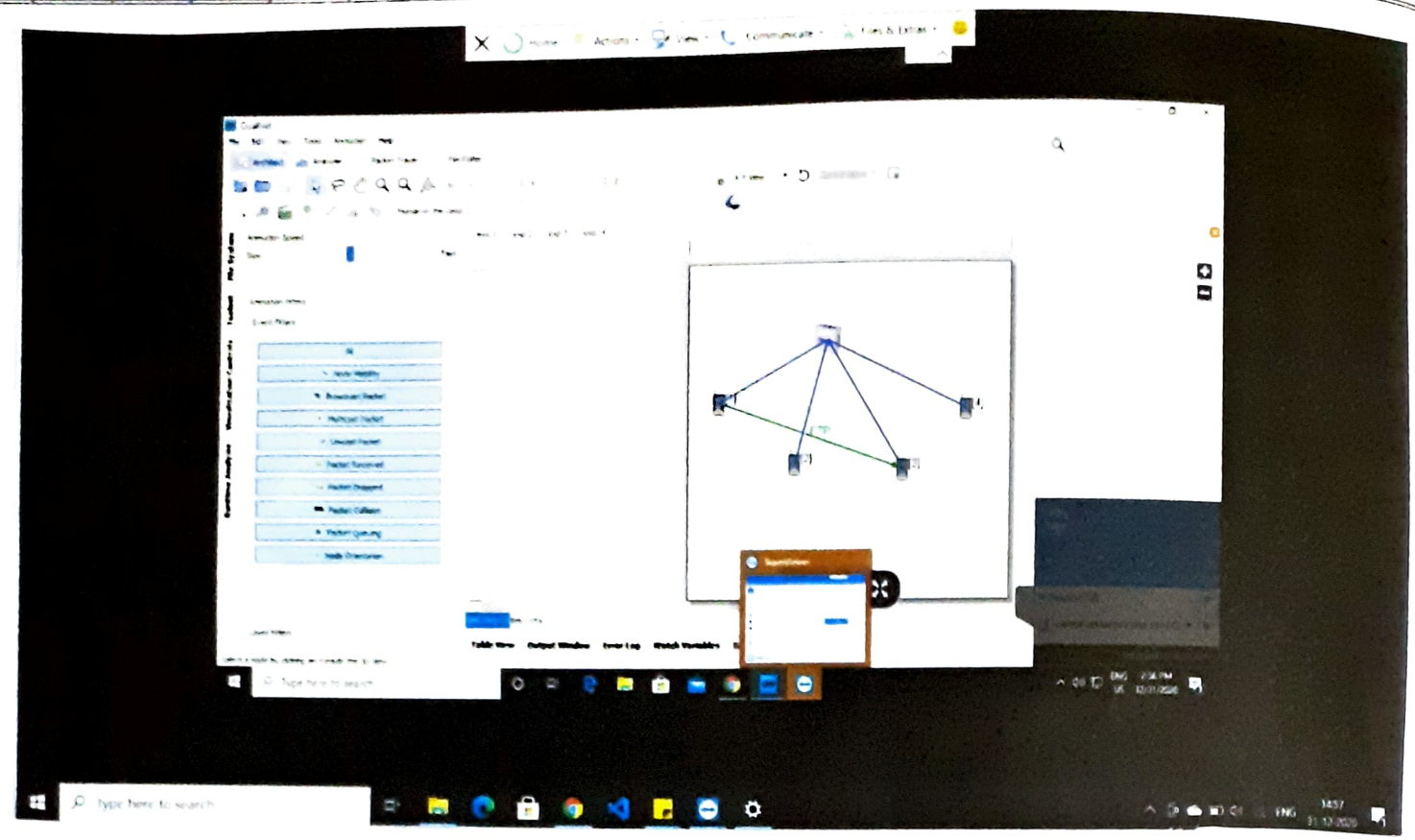
Expt. No. _____ Page No. _____

PART - B

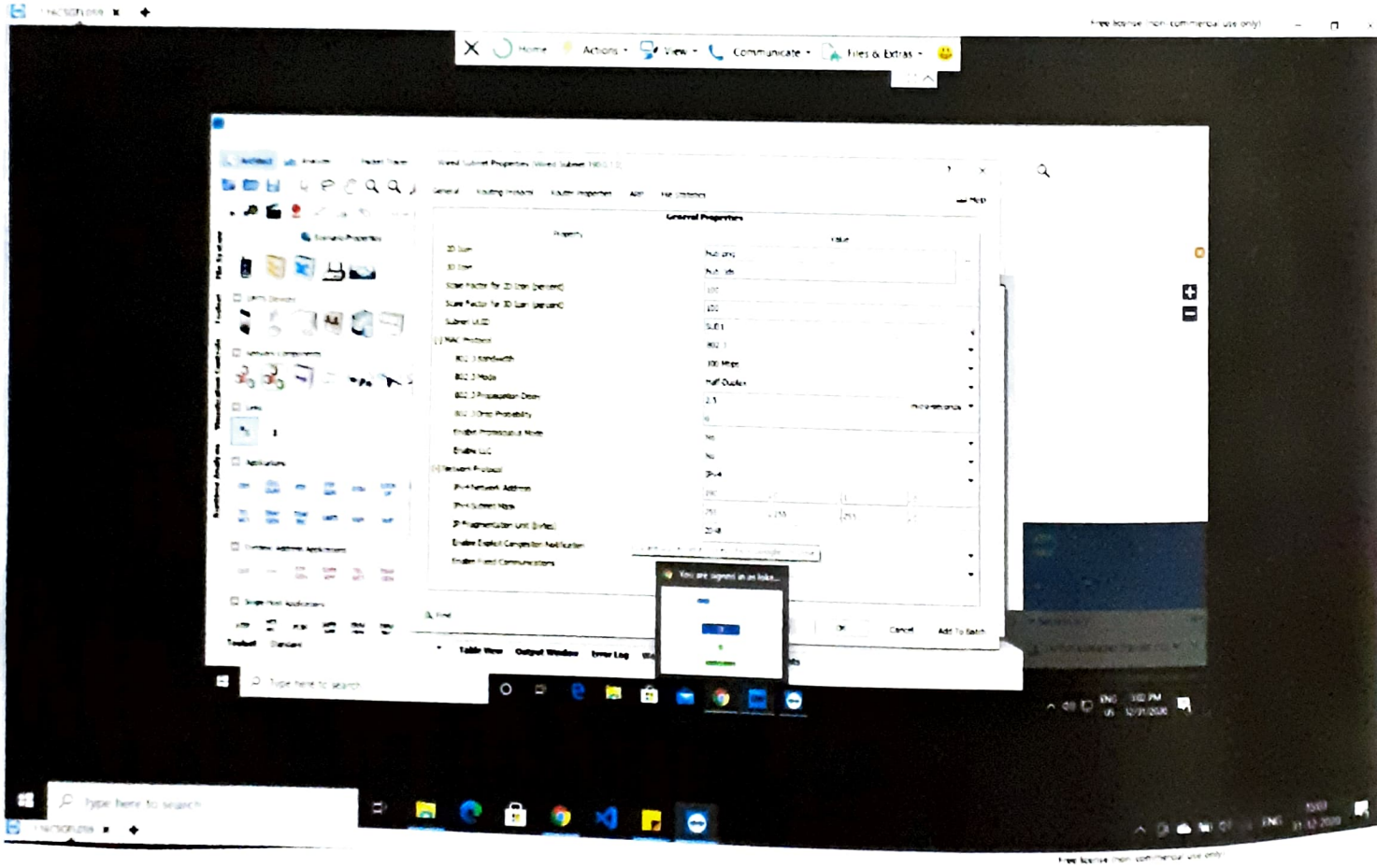
1. Setup an IEEE 802.3 network with a) hub b) switch c) ~~switching~~ Heirarchy of switch
Apply the TCP, Telnet applications between nodes. Vary the number of nodes, vary the bandwidth, queue size and observe the packet drop probability.

a) hub

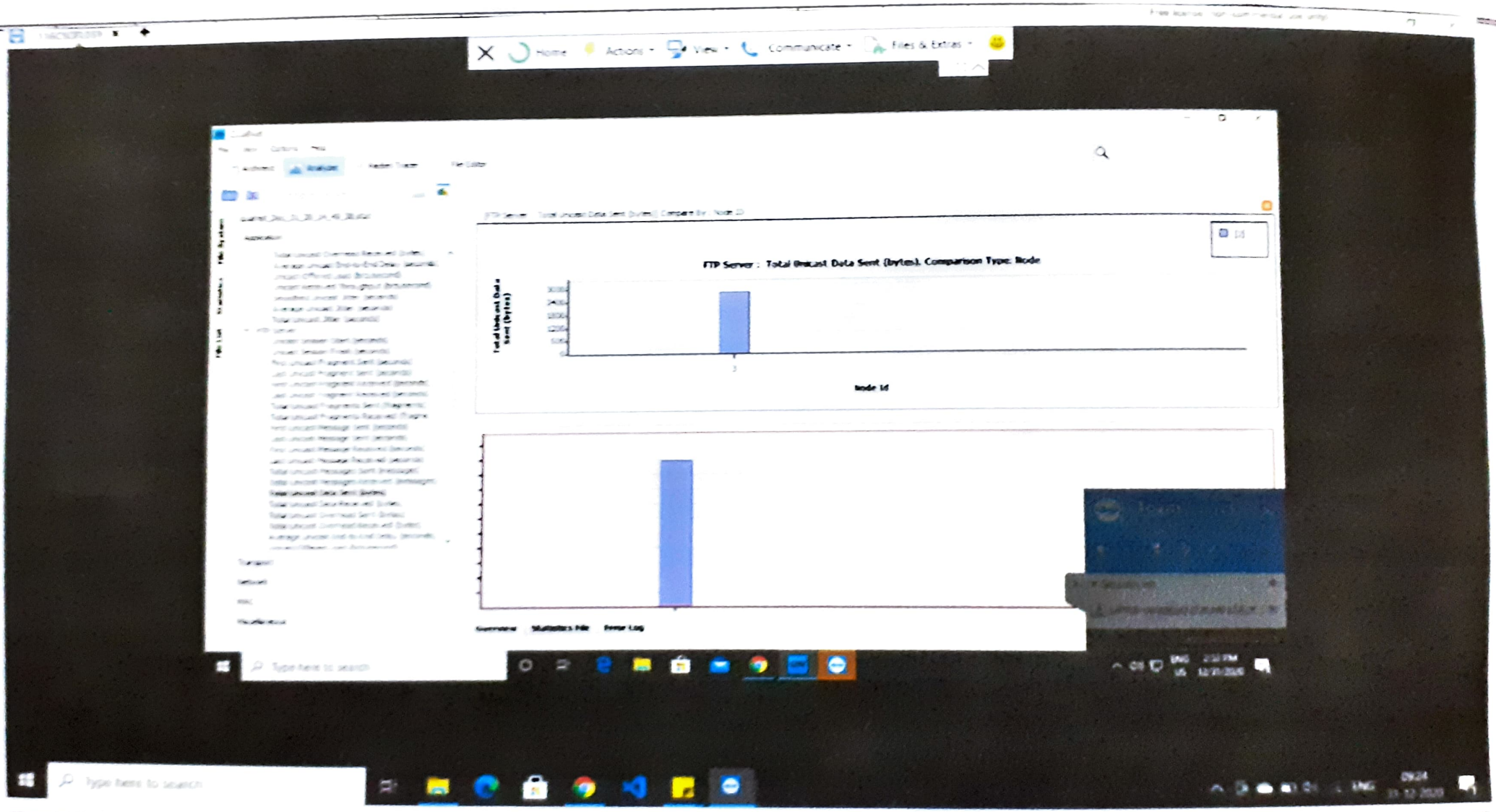
Teacher's Signature _____



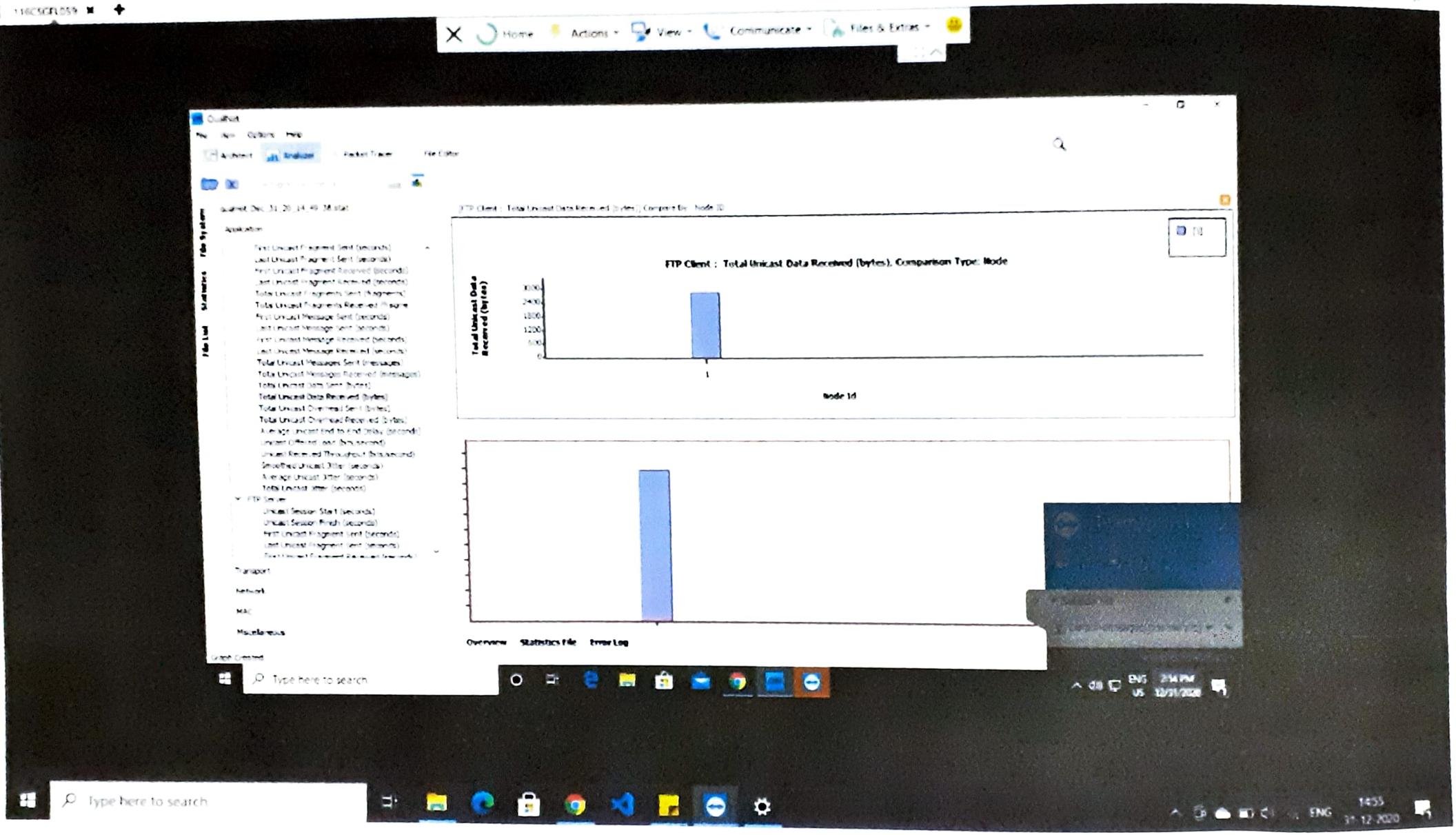
Network Topology: 1 hub connected to 4 mobile devices. FTP between device device 1 and device 3.



Wired ~~Subnet~~ Subnet Properties



Statistics : FTP Server : Total Unicast Data Sent (bytes)



Statistics – FTP Client : Total Unicast Data received (Bytes)

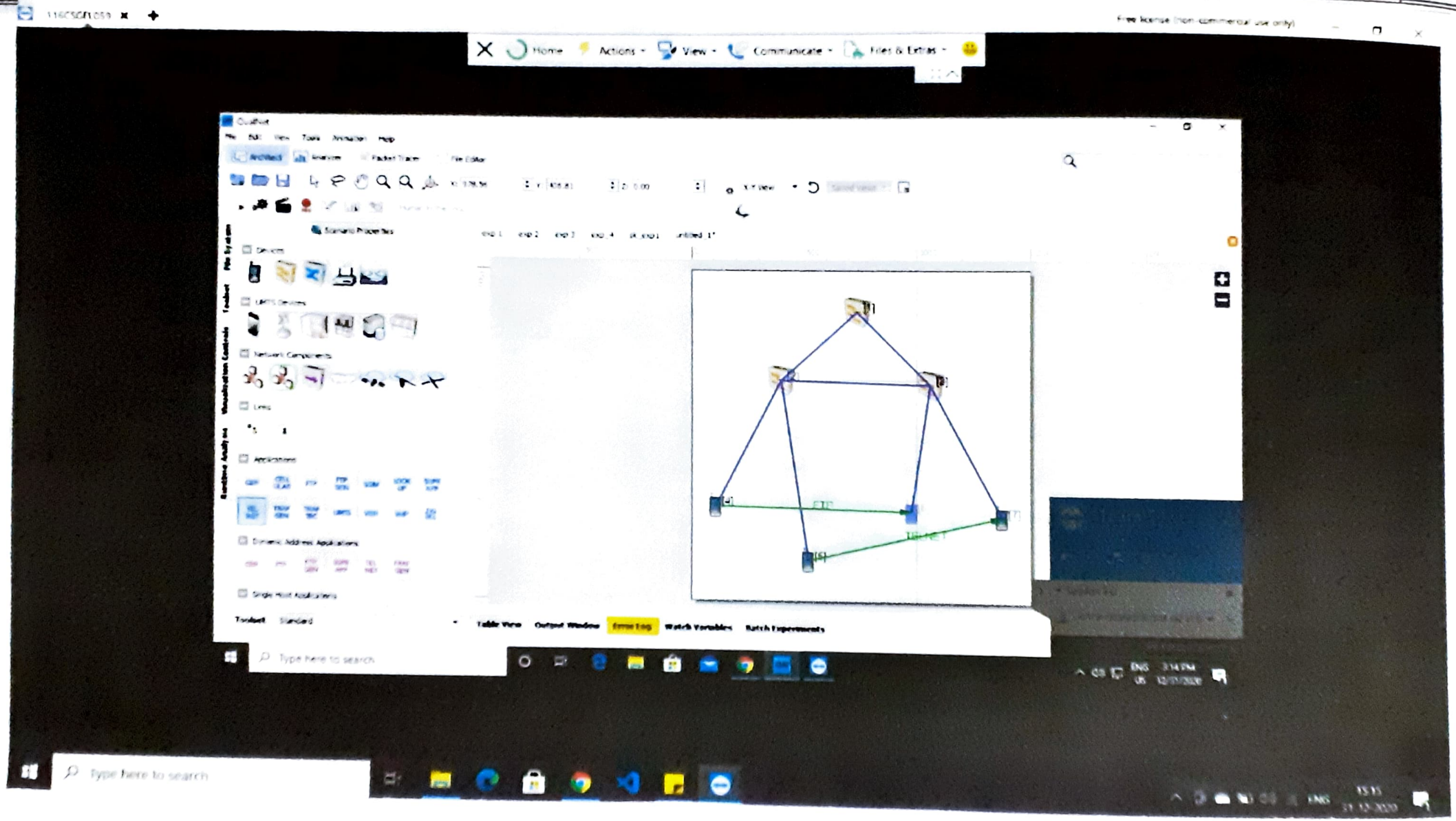
Date _____

Expt. No. _____ Page No. _____

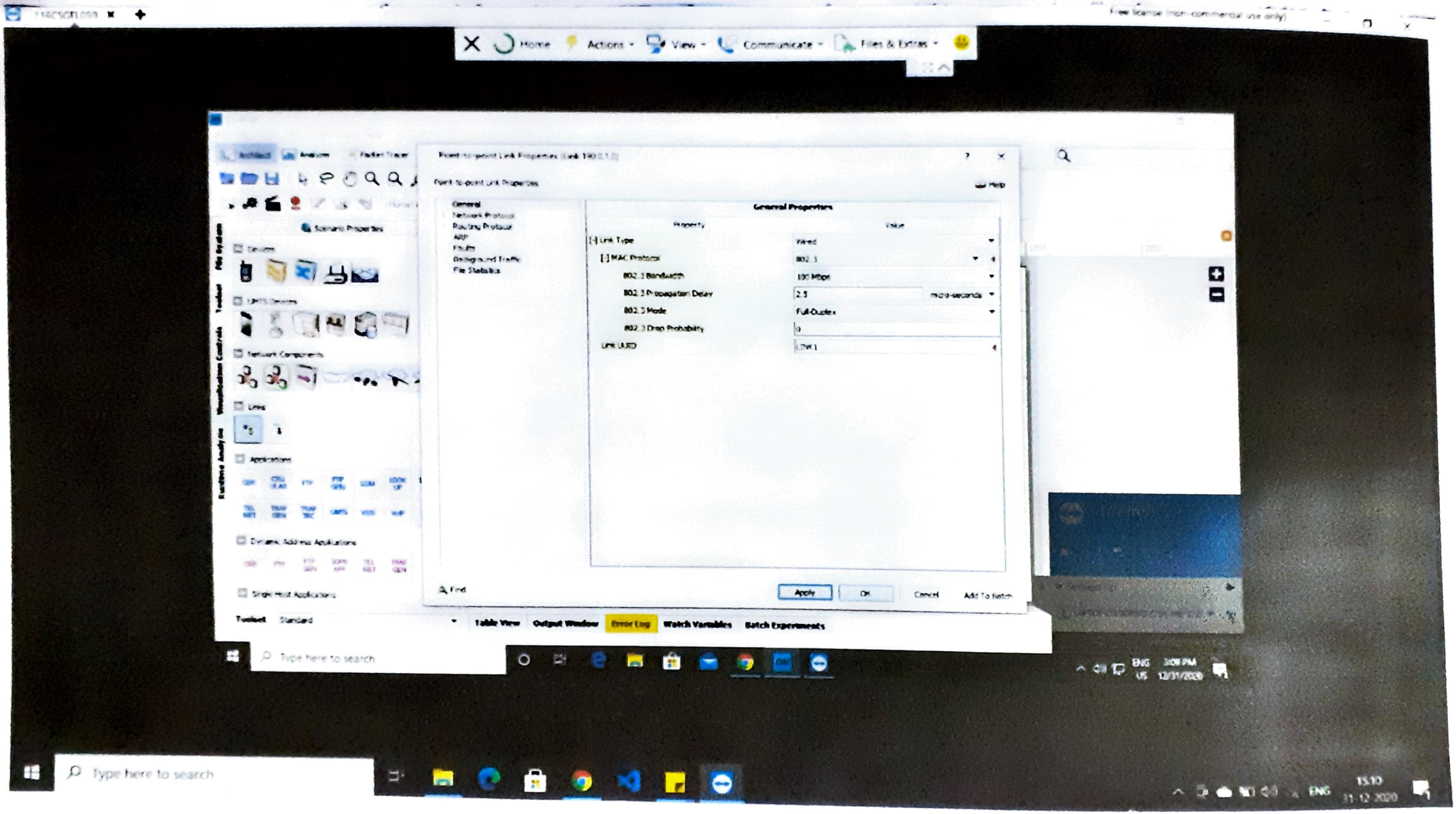
b) switch and Hierarchy of switch

& c)

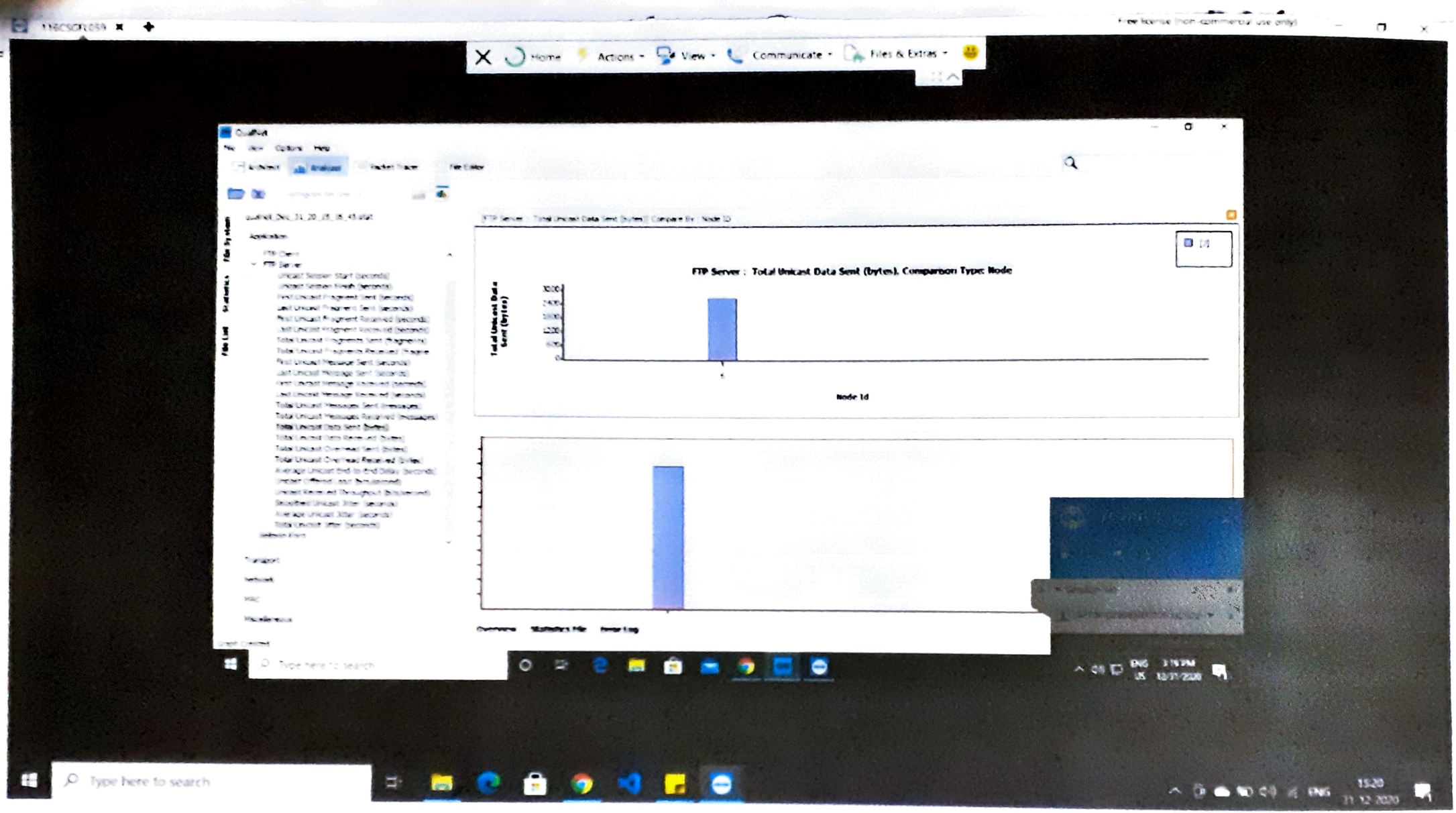
Teacher's Signature _____



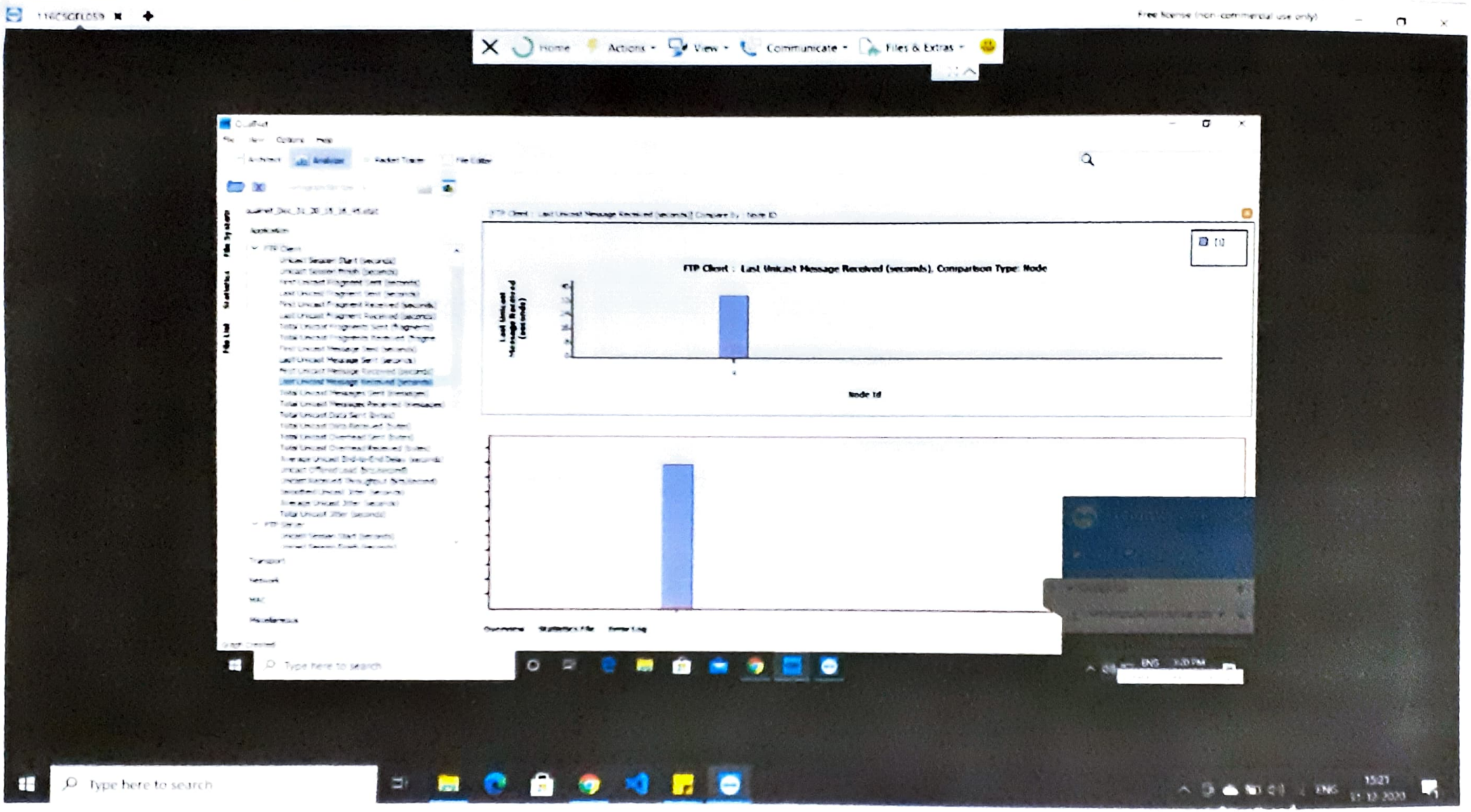
Network Topology - 1 switch at first level, 2 switches at second level
FTP between device 4 and 6, Telnet between device 5 and 7



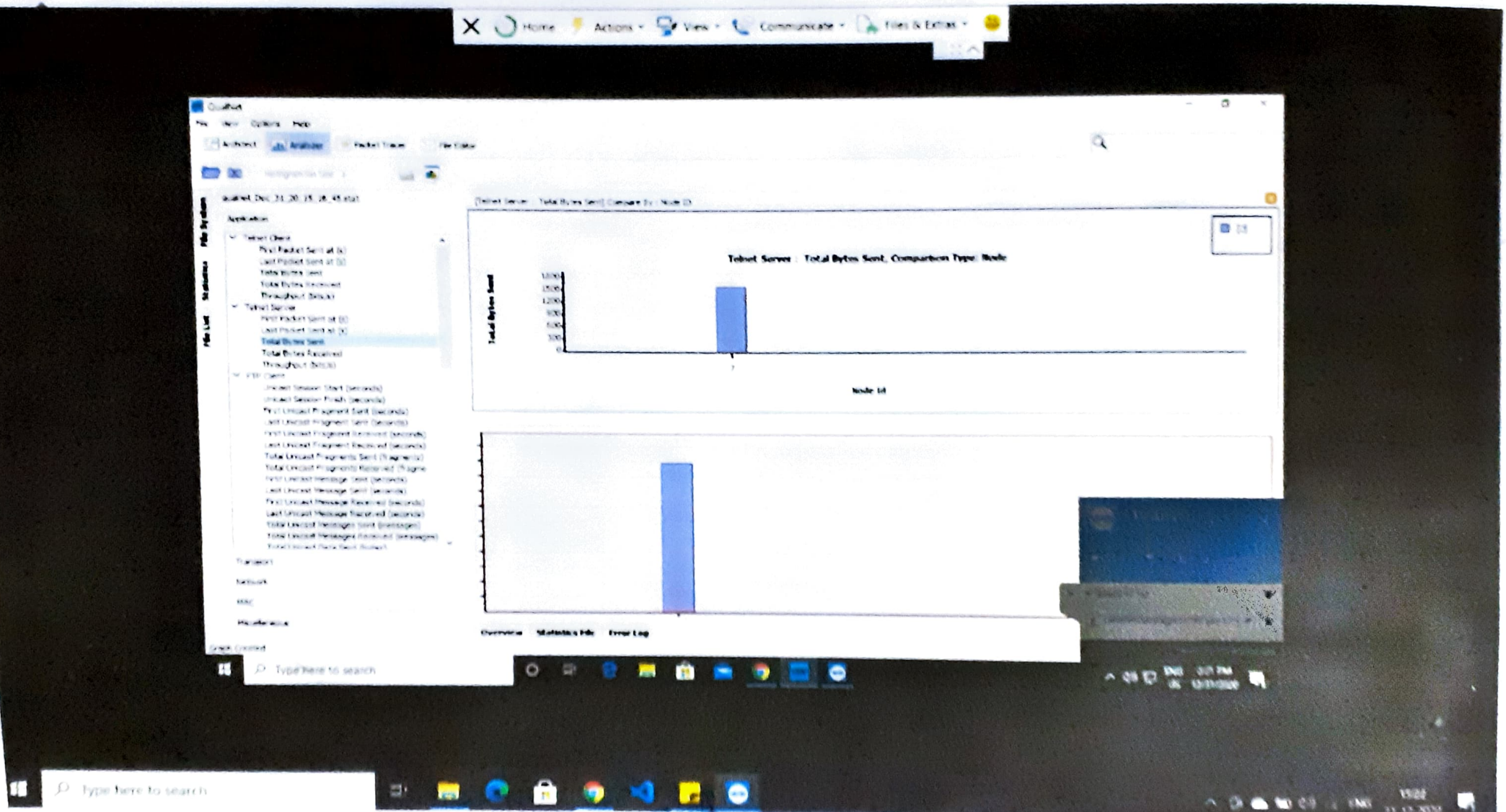
Point-to-Point Link Properties - MAC protocol set to 802.3



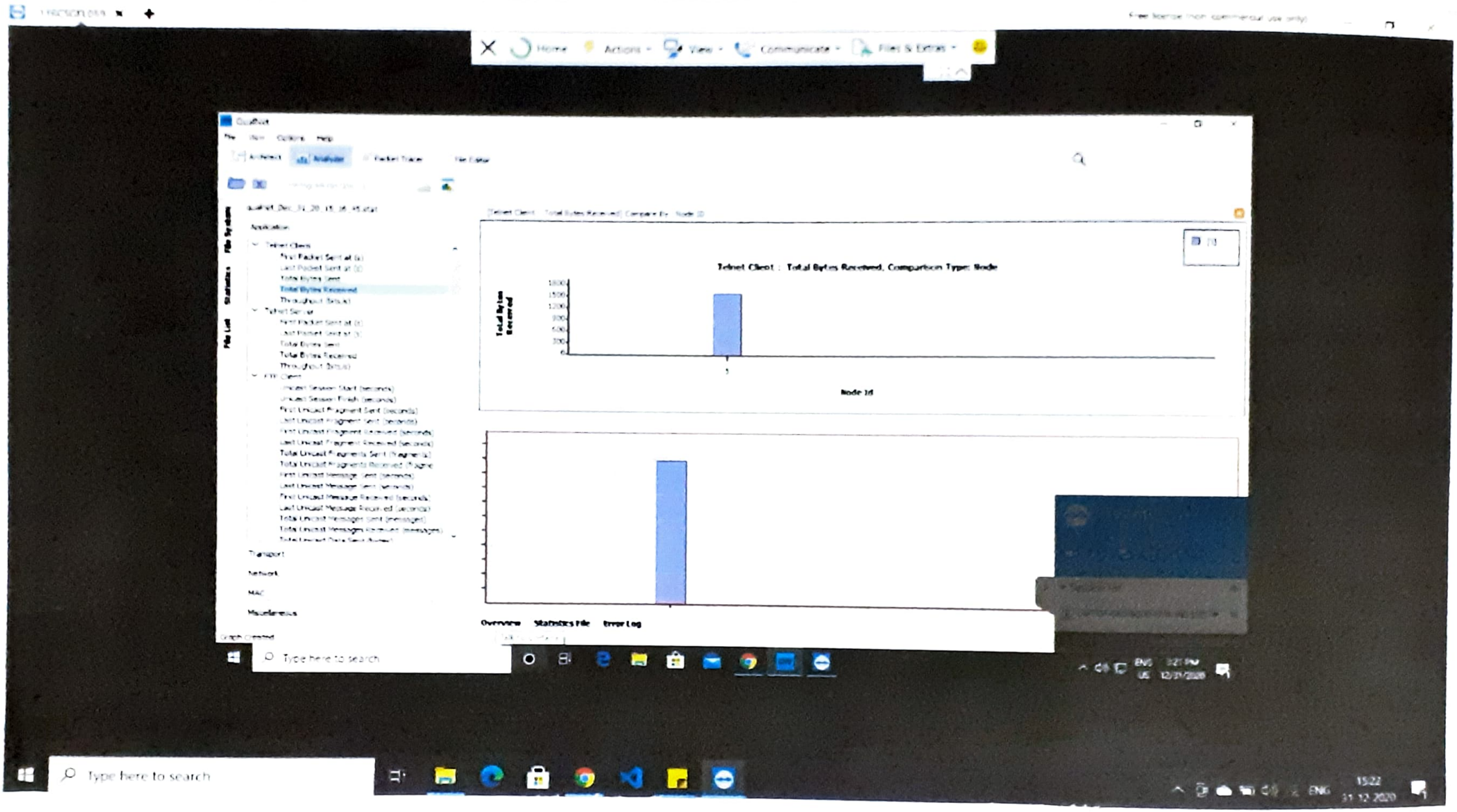
Statistics - FTP server: Total Unicast Data Sent (bytes)



Statistics - FTP client : Last Unicast message received (bytes)



Statistics: Telnet Server: Total Bytes sent



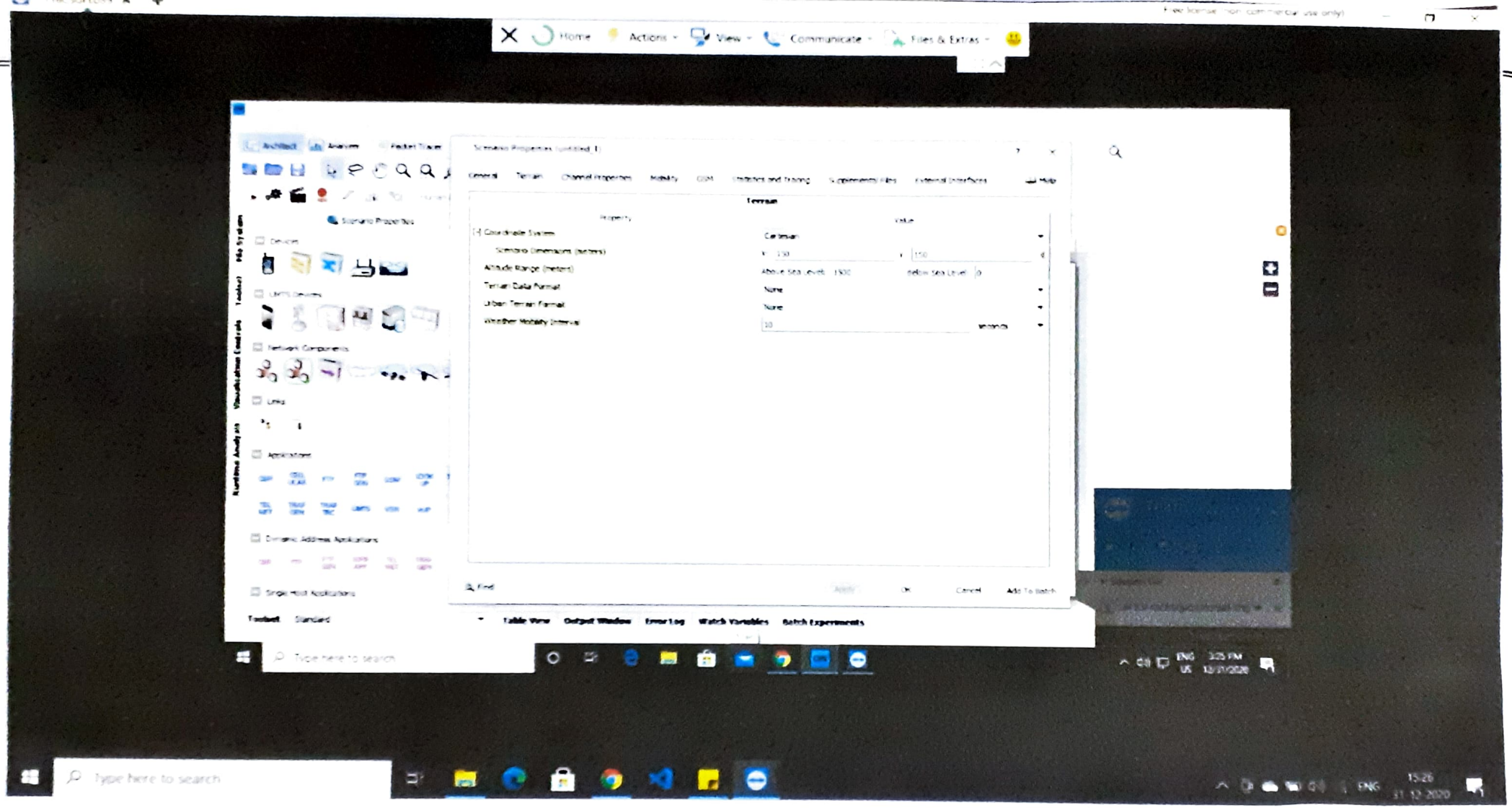
Statistics - Telnet Client : Total Bytes recieved, Comparison type : Node

Date _____

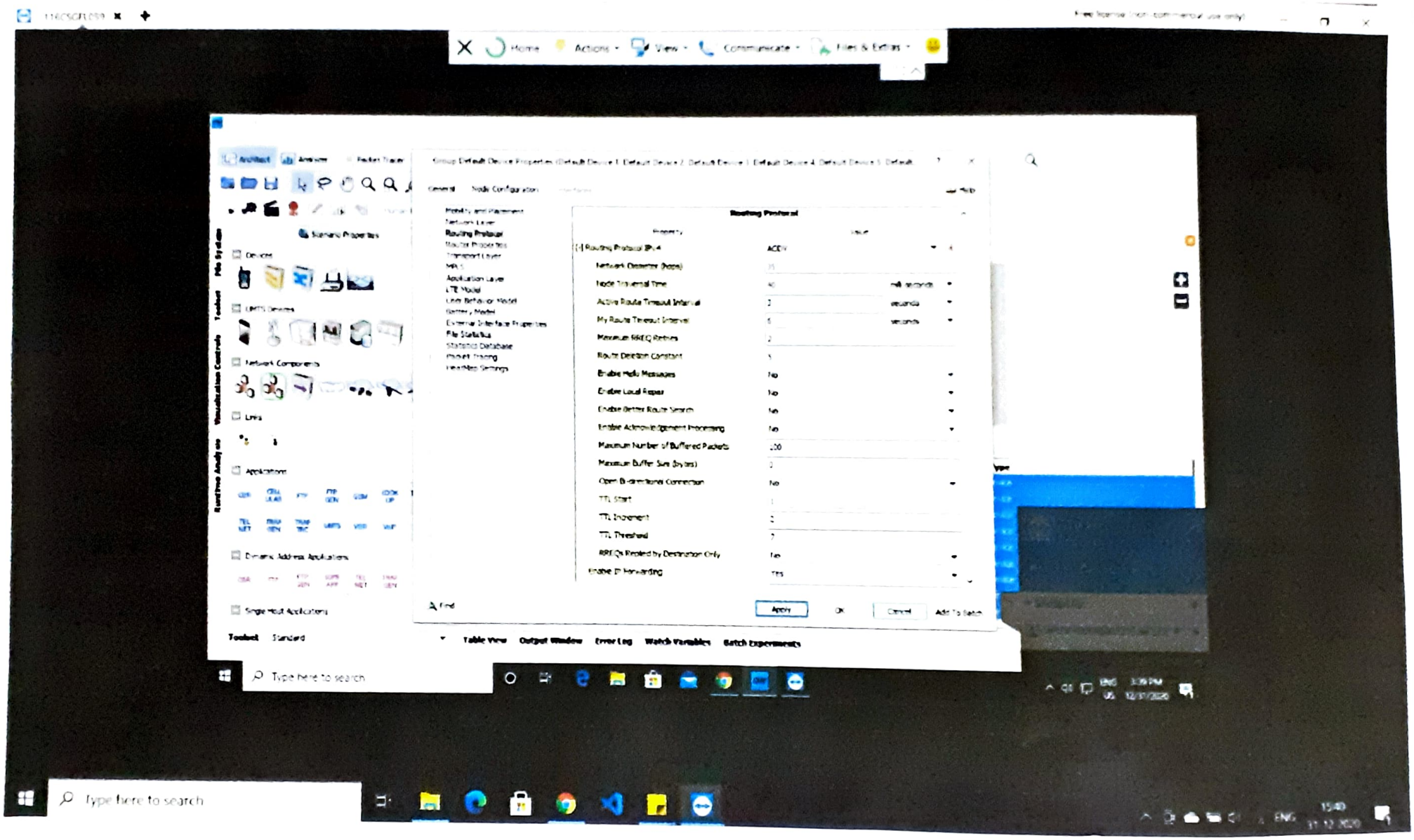
Expt. No. _____ Page No. _____

2. Setup a wireless sensor network with atleast 2 devices co-ordinators and nodes. Provide Constant Bit rate (CBR) and variable Bit Rate (VBR) application between nodes.

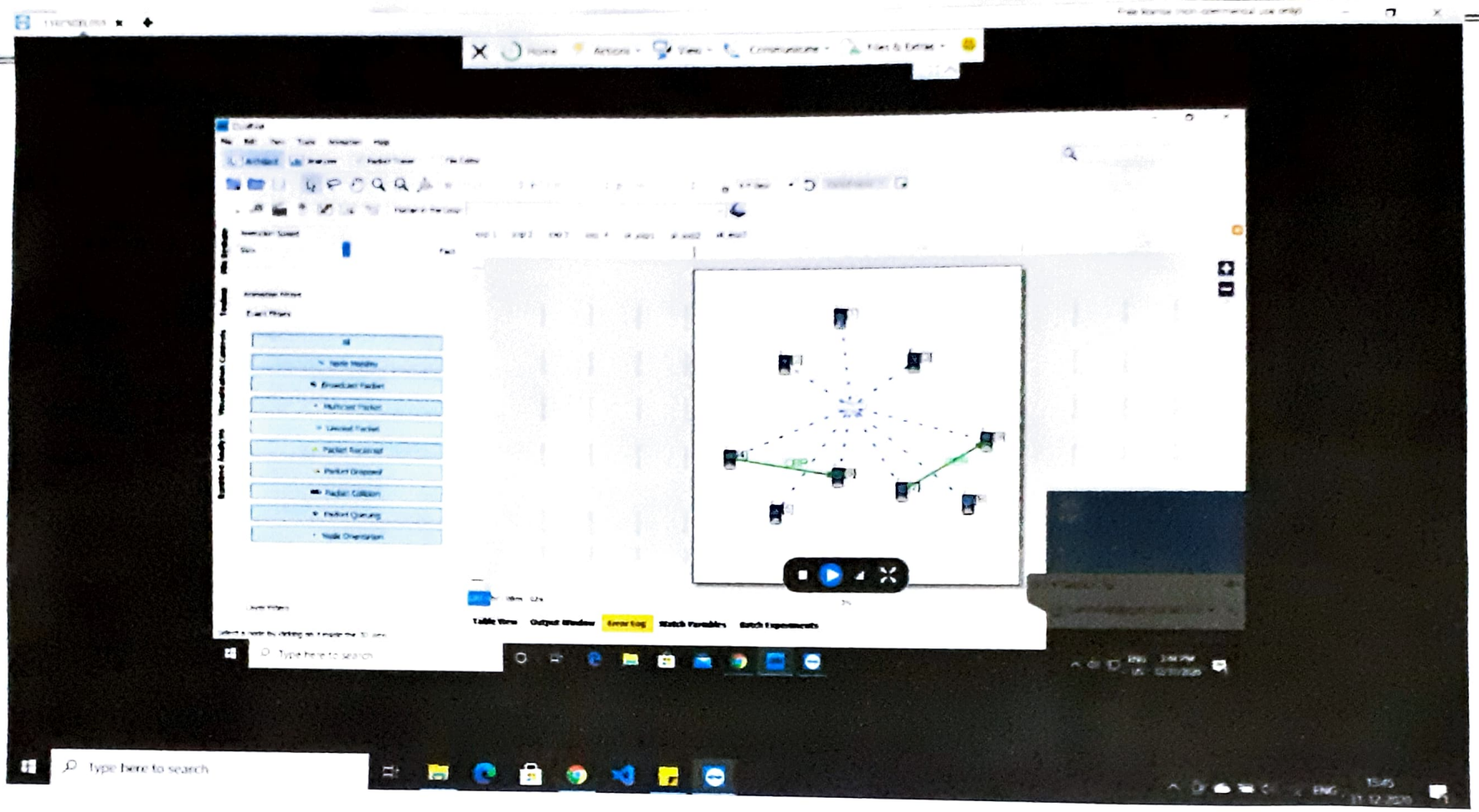
Teacher's Signature _____



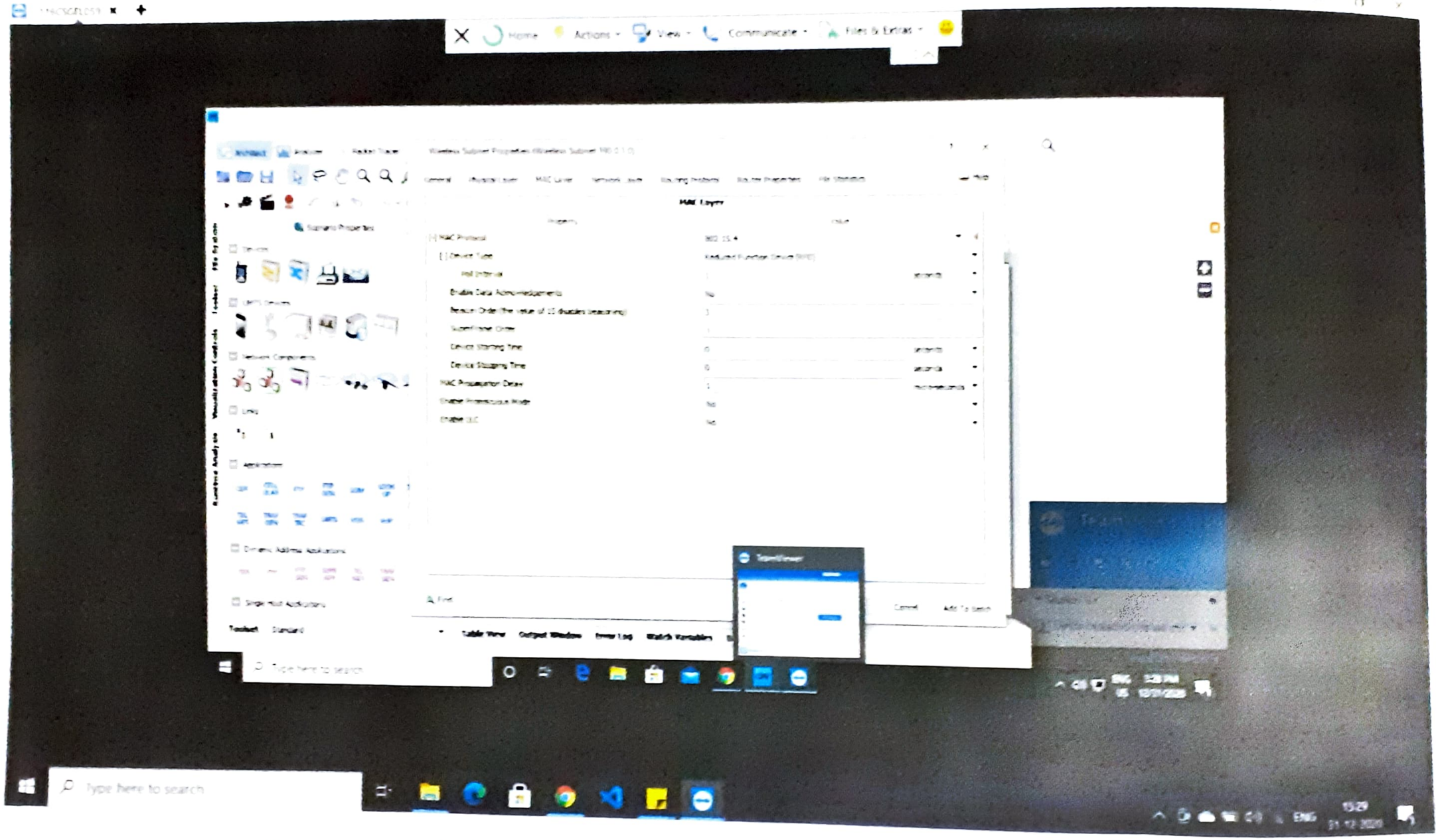
Scenario properties - Change the dimensions to 150 x 150 since it is a wireless network with lesser range.



Group Default device properties - Routing protocol for all devices to be set to AODV : Ad Hoc distance Vector Routing protocol

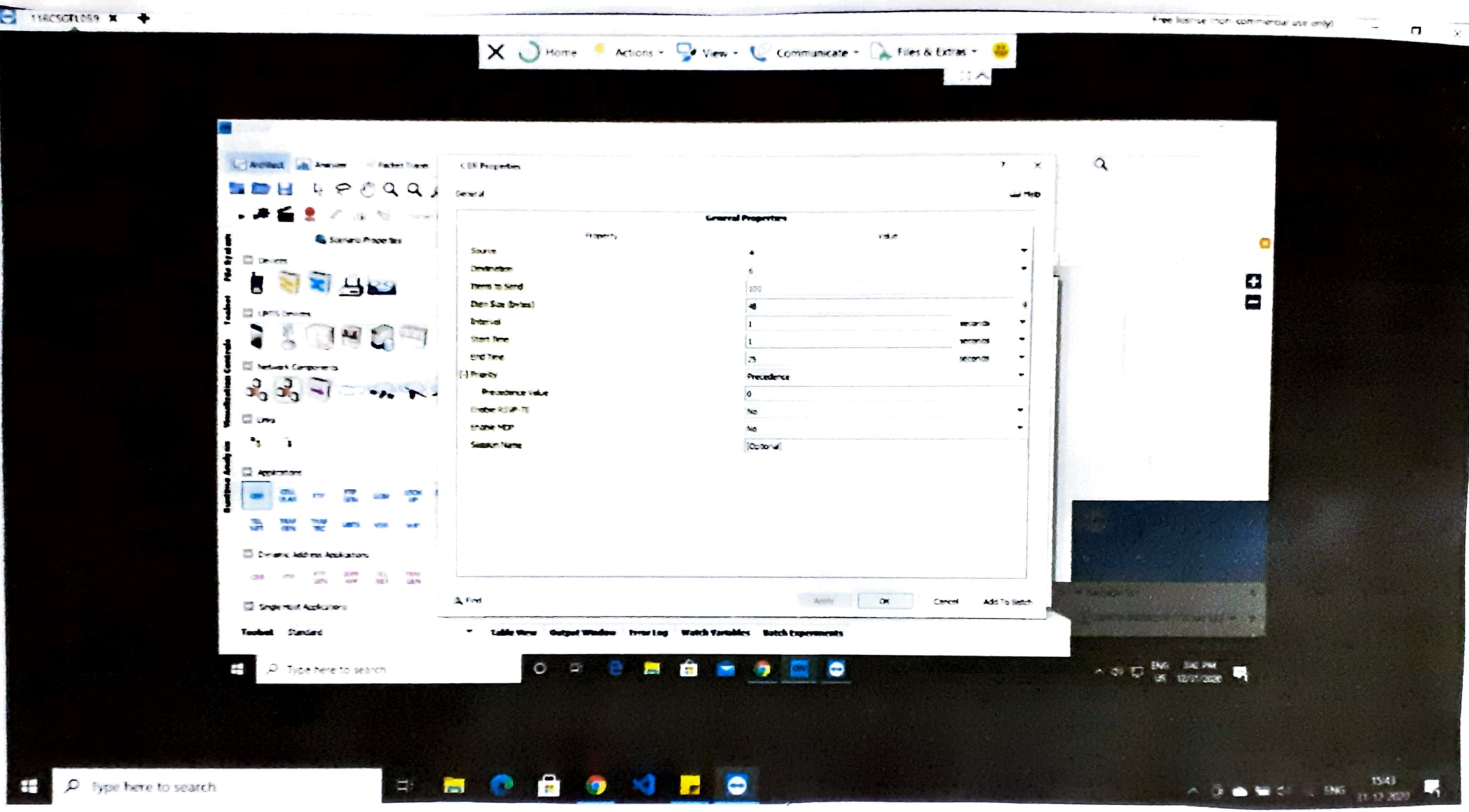


Network topology : 9 devices connected - Device 1 : Main coordinator (Full Function Device), Device 2 and 3 - Pan coordinator (FFD), Rest of the devices are RFDs (Reduced Function Device). CBR between Device 4 and 6 and Device 7 and 9 respectively.

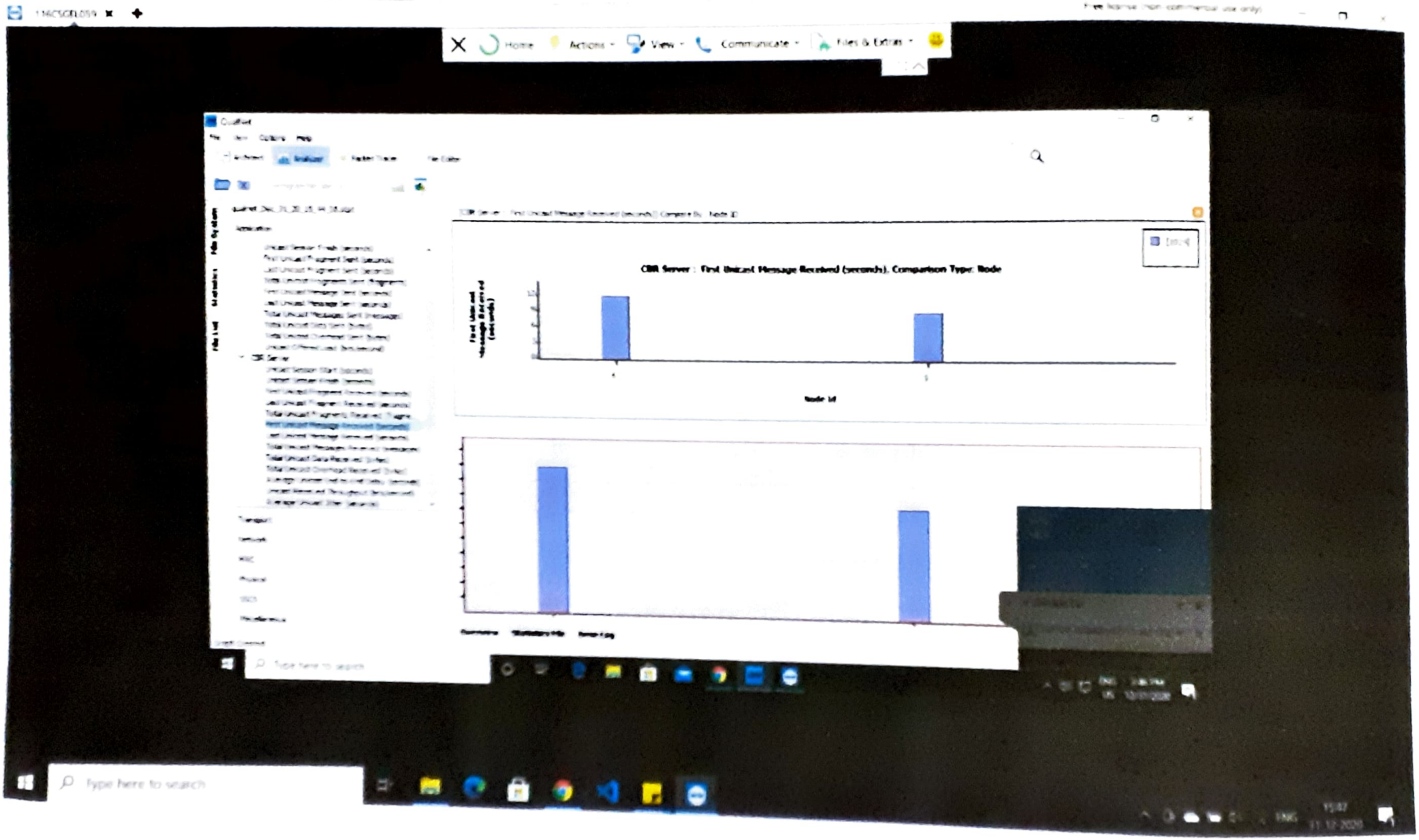


Wireless Subnet properties - MAC protocol set to 802.15.4

Device type - Reduced Function Device (RFD)



CBR properties : Item size (bytes) changed to 48 as the default 512 is too large for a wireless network.



Statistics : CBR server : First unicast Message Received (seconds).

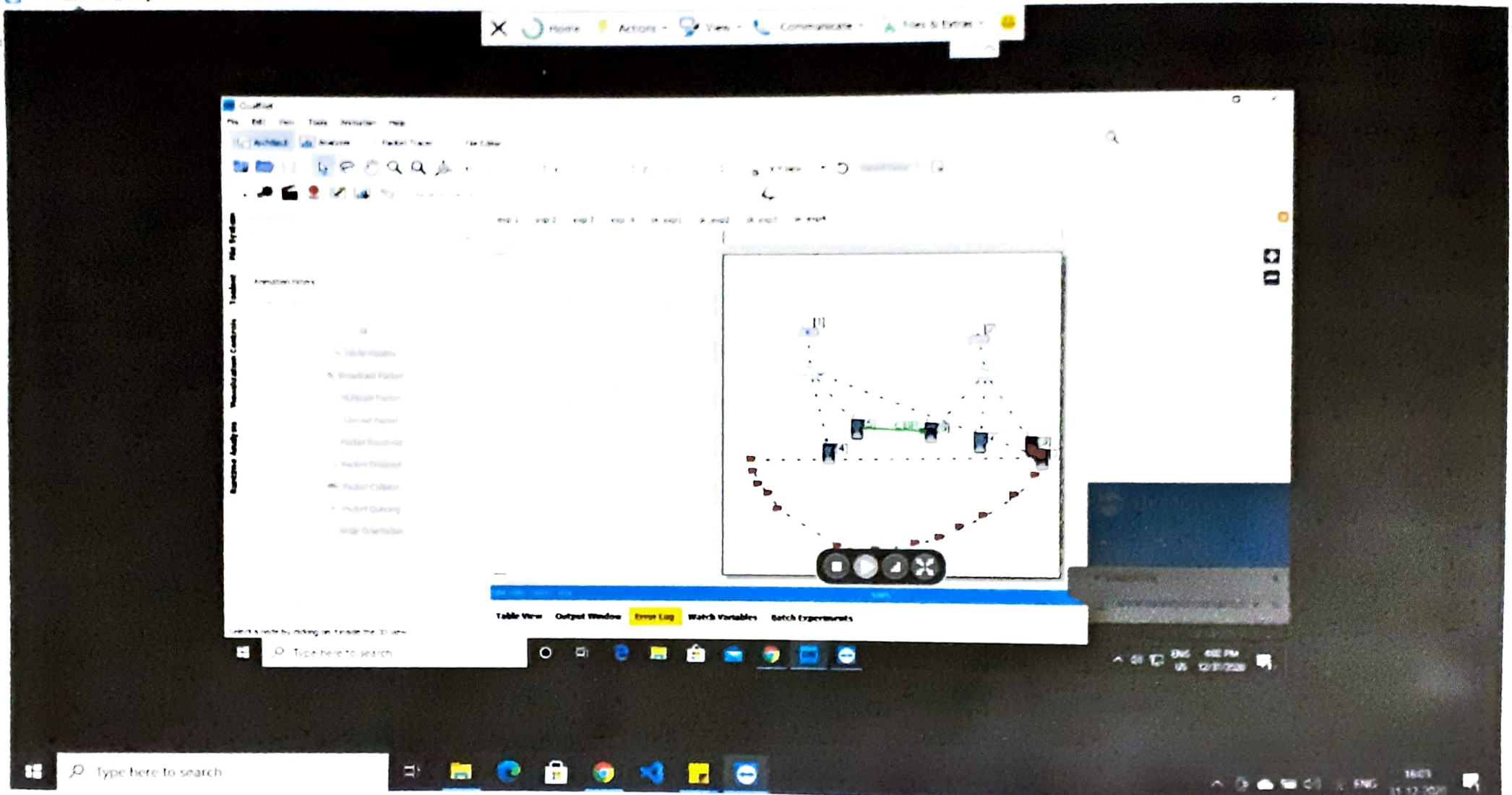
Few data packets lost as it is wireless network.

Date _____

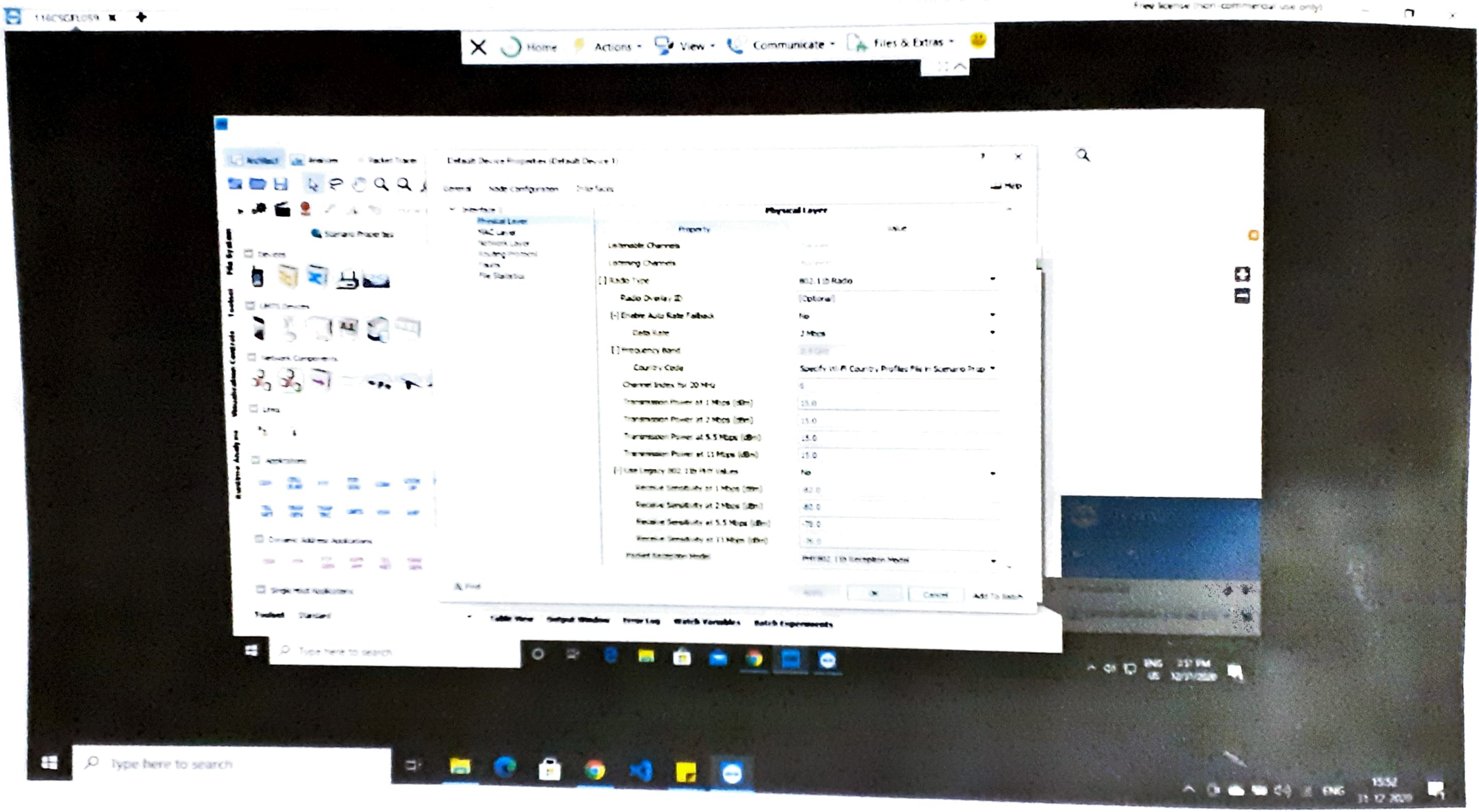
Expt. No. _____ Page No. _____

3. Set up an IEEE 802.11 network with at least 2 access points. Apply CBR, VBR applications between device belonging to same access point and different access points. Provide roaming of any device. Vary the number of access points and devices. Find out the delay in MAC layer, packet drop probability.

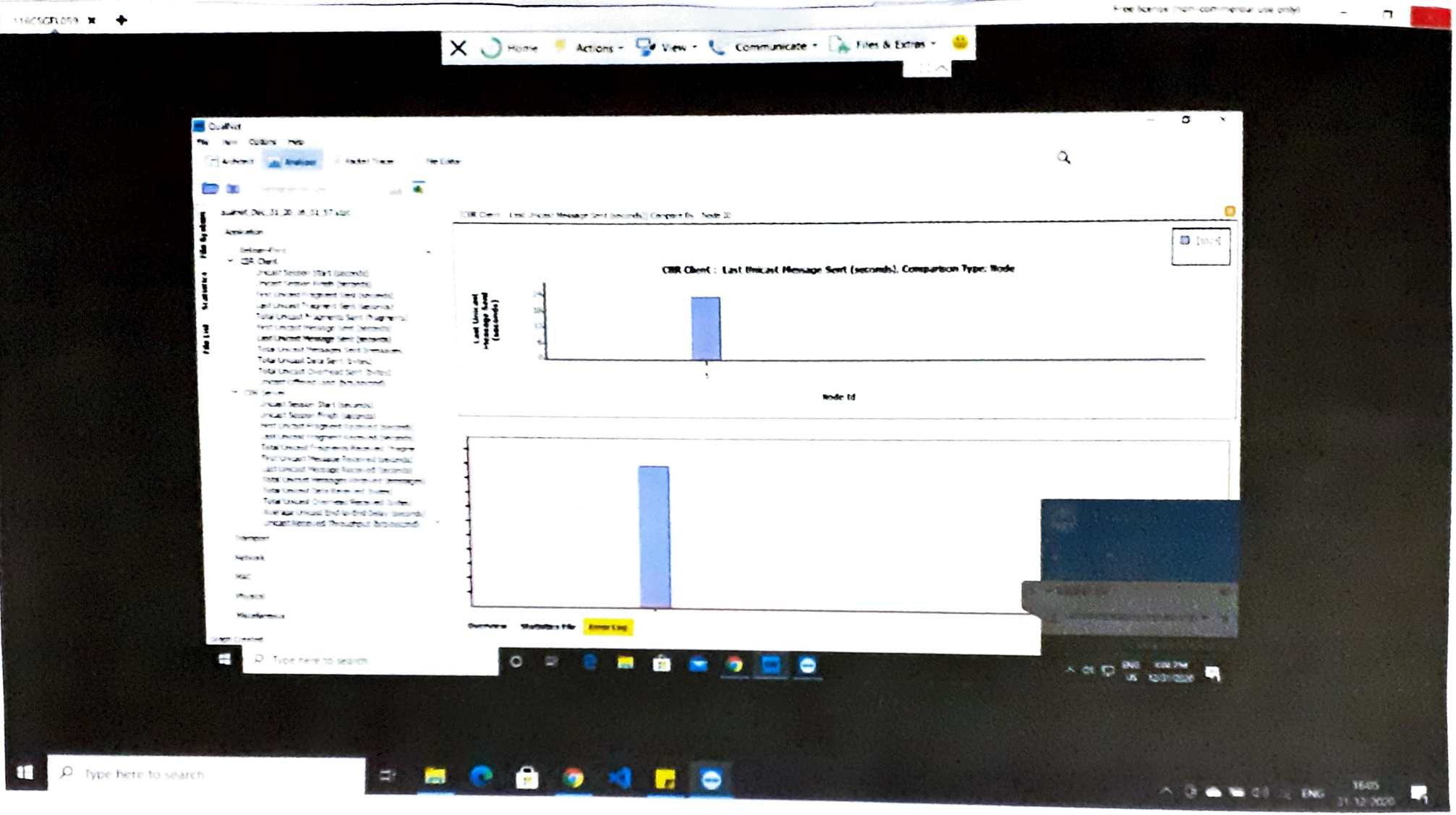
Teacher's Signature _____



Network topology - Device 3, 4, 5 connected to Access point (AP) 1 and Device 6, 7, 8 connected to AP2. CBR application between nodes 5 and 6.



Default Device Properties - For AP 1 and 2 change the radio type to 802.11b radio



Statistics : CBR Client - Last Unicast Message sent (seconds) (3 seconds) - Comparison type : Node.