R.V. COLLEGE OF ENGINEERING

(Autonomous Institution affiliated to VTU, Belagavi)
BENGALURU - 560 059.

Laboratory Certificate

This is to certify that.....SHUVAM MITRA.....has satisfactorily completed the course of Experiments in.....Practical ..... Prescribed by the Department of ..... for the ..... Semester of BE Graduate Programme during the year 20 -20

Signature of Head of the Department
Date :

Signature of the Faculty in-charge

50

Name of the Candidate .....SHUVAM MITRA.....

USN .....1RV18CS165.....

Server terminal

./server 127.0.0.1

Soket was created

Binding socket

127.0.0.1 connected

A request for filename test.txt Recieved

Request complete

Client terminal

./client 127.0.0.1

socket created

Connection accepted

Filename : test.txt

contents :

Hi Hello !

EOF

Name of Experiment..... Date..... Page No

Experiment No..... Experiment Result.....

1) Implement a client server comm. using socket prog.

// Server

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

int main()
{ int sock, create_socket, new_socket, addrlen, fd;
  int bufsize = 1024;
  char * buffer = malloc (bufsize);
  char fname [256];
  struct sockaddr_in address;
  if ((create_socket = socket (AF_INET, SOCK_STREAM, 0)) > 0)
        printf (" created ")
  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
        printf ("Bind ");
  listen (create_socket, 3);
  addrlen = sizeof (struct sockaddr_in);
  new_socket = accept (create_socket, (struct sockaddr *) & address, & addrlen);
  if (new_socket > 0)
        printf (" the client %s is connected \n", inet_ntoa (address . sin_addr));
  recv (new_socket, fname, 255, 0);
```

Teacher's Signature : _____

Name of Experiment.....

Experiment No.....

Date.....

Experiment Result.....

Page No

```
if((fd= open(fname, O_RDONLY))<0)
{
    perror("File open failed"); exit(0); }
while((cnt = read(fd, buffer, bufsize))>0)
        send(new_socket, buffer, cnt, 0);
close(new socket);
return;
}

// Client

int main(int argc, char * argv[])
{ int create_socket;
  int bufsize = 1024;
  char* buffer = malloc(bufsize);
  char fname[256]= "test.txt";
  struct sockaddr_in address;
  if((create_socket = socket(AF_INET, SOCK_STREAM, 0)) >0)
                printf("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))
                printf("Accepted");
  send(create_socket, fname, sizeof(fname), 0);
      close(create_socket);
}
```

Teacher's Signature : _____

Output :

```
./distvec
No. of vertices : 3
Enter adjacency matrix
0 2 999
2 0 3
999 3 0
```

Router 0

```
0          Cost 0
1 ← 0      Cost 2
2 ← 1 ← 0  Cost 5
```

Router 1

Name of Experiment..... Date.....

Experiment No..... Experiment Result..... Page No

2) Distance vector routing for simple topology of routers

```
#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; } } }}
int main()
{ scanf("%d", &n)
   int i, j, n;
   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);
      for(i=0; i<n; i++)
```

Teacher's Signature : _____

Name of Experiment..... Date..... Page No

Experiment No..... Experiment Result.....

```
if (i != source)
  {j = i;
  while (p[j] != -1)
    { printf ("%d <- ", j);
      j = p[j];
    }}
printf ("%d\t Cost %d \n\n", source, d[i]);
}
```

Checksum:

Enter IP header in 16 bit word

1
2
3
4
5
6
7
8
9

completed checksum at sender ffd2

Enter IP header info in 16 bit word

1
2
3
4
5
6
7
8
9

Completed checksum at receiver ffd2

No error

Hamming

enter 4 bit data word 1 2 3 4

The 7 bit hamming code word:

1 2 3 4 0 1 0

enter 7 bit received code

1 2 3 4 0 1 1

Syndrome: 0 0 1

error in 1st bit from right

Corrected code:

1 2 3 4 0 1 0

Name of Experiment..... Date..... Page No

Experiment No..... Experiment Result.....

3) Write a program to determine error detection and correction using checksum and hamming.

```
//Checksum
#include<stdio.h>
unsigned fields[10];
unsigned short checksum()
{ int i;
  int sum=0;
  for(i=0; i<9; i++)
  { scanf("%x",&fields[i]);
  sum=sum+(unsigned)fields[i];
  while(sum>>16)
  sum=(sum&0xFFFF)+(sum>>16); }

  sum=~sum;
  return(unsigned short)sum; }

int main()
{ unsigned short result1, result2;
  //sender
  result1=checksum();
  printf("Computed checksum at sender %x\n", result1);
  //Receiver
  result2=checksum();
  printf("%x\n", result2);
  if(result1!=result2)
```

Teacher's Signature : _____

Name of Experiment.....
Experiment No.....
Date.....
Experiment Result.....
Page No

```
printf ("No error ");
else
printf ("error ");

// Hamming code
#include <stdio.h>
#include <stdlib.h>
int main()
{ int a[4], b[4], r[3], s[3], i, q[3], c[7];
  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;
for(i=2; i>=0; i--)
{ printf("%d\t", a[i]);}
for(i=2; i>=0; i--)
  printf("%d\t", r[i]);

for(i=7; i>0; i--)
  scanf("%d", &c[i]);

b[3]=b[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[2]+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 : _____

Name of Experiment..... Date..... Page No

Experiment No..... Experiment Result.....

```
for (i=2; i>=0; i--)
    printf ("%d", s[i])
if (s[2]==0) && (s[1]==0) && (s[0]==0))
    printf ("Error free");
if ((s[2]==1) && (s[1]==1) && (s[0]==1))
{   printf ("Error in received codeword .7th");
    if (c[7]==0)
      c[7]=1;
    else
      c[7]=0;
    printf ("%d\t", c[i]); }

if ((s[2]==1) && (s[1]==1) && (s[0]==0))
{   printf ("6th bit");
    if (c[6]==0)
      c[6]=1;
    else
      c[6]=0;
    for (i=7; i>0; i--)
    printf ("%d", c[i]) }
    if
if ((s[2]==1) && (s[1]==0) && (s[0]==1))
{   printf ("5th");
    if (c[5]==0)
      c[5]=1;
    el
      c[5]=0;
    printf ("%d", c[i]) }
```

Teacher's Signature : _____

Name of Experiment..... Date..... Page No

Experiment No..... Experiment Result.....

```
if ((s[2] == 1) && (s[1] == 0 || s[0] == 0))
{
    printf("4M");
    if (c[4] == 0)
      c[4] = 1;
    else
      c[4] = 0;
    printf("%d", c[i]);
}

if ((s[2] == 0) && (s[1] == 1) && (s[0] == 1))
{
      // 3rd bit
      if (c[3] == 0)
        c[3] = 1;
      else
        c[3] = 0;
}

if ((s[2] == 0) && (s[1] == 1) && (s[0] == 0))
{
      // 2nd bit
}

if ((s[2] == 0) && (s[1] == 0) && (s[0] == 1))
{
      // 1st bit
}
```

Teacher's Signature : _____

Sender:

./sender

Listener ./listener

RVCE - CSE

RVCE - CSE

RVCE - CSE

RVCE - CSE

Name of Experiment.....
Date.....
Experiment No.....
Experiment Result.....
Page No

4) Multicast routing mechanism

//listener

```
#define HELLO_PORT 12345
#define HELLO_GROUP "225.0.0.37"
#define MSGBUFSIZE 25
int main (int argc, char *argv[])
{ struct sockaddr_in addr;
  int fd, nbytes, addrlen;
  struct ip_mreq mreq;
  char msgbuf[MSGBUF];
  u_int yes = 1;
if( (fd = socket (AF_INET, SOCK_DGRAM, 0)) < 0)
  setsockopt (fd, SOL_SOCKET, SO_REUSADDR, &yes, sizeof(yes)
  memset (&addr, 0, sizeof (addr));
  addr.sin_family = AF_INET;
  addr.sin_addr.s_addr = htonl (INADDR_ANY);
  addr.sin_port = htons (HELLO_PORT);
  bind (fd, (struct sockaddr *) &addr, sizeof (addr)) <
  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) )
  while(1)
  { addrlen = sizeof (addr);
    if ( (nbytes = recvfrom( fd, msgbuf, MSGBUF, 0, (struct sockaddr) &addr
                                    &addrlen)) < 0)
      { puts (msgbuf); }
  }
}
```

Teacher's Signature : _____

Name of Experiment..... Date.....

Experiment No..... Experiment Result..... Page No

```
//Sender
#DEFINE HELLOPORT 12345
#DEFINE MG "225.0.0.37"
int main (int argc, char * argv[])
{ struct ip_mreq mreq;
  int fd, cnt;
  char * message = "RVCE-CSE";
  fd=socket (AF_INET, SOLK_DGRAM, 0)
  memset (&addr, 0, sizeof (addr));
  addr.sin_family = AF_INET;
  addr.sin_addr.s_addr = inet_addr (MG);
  addr.sin_port = htons (HELLO PORT);
  while(1)
  { sendto (fd, message, sizeof(msg), 0, (struct sockaddr *) &addr,
                                    sizeof(addr))
  {
    sleep (1);
  }
}
```

Teacher's Signature : _____

Server:

./server
Socket created
Binding 0.0.0.0
The client 127.0.0.1 connected on port 16
The client 127.0.0.1 connected on port 17
Inside child
Client: Hi
Me: Hi C1
Client Hi
Me: Hi C2

Client 1

./client 127.0.0.1
Me: Hi
Server: Hi C1

Client 2

./client 127.0.0.1
Me: Hi
Server: Hi C2

Name of Experiment..... Date..... Page No

Experiment No..... Experiment Result.....

5) Concurrent chat server

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

void str_echo(int connfd)
{ int n; int buffer = 1024;
  char *buf = malloc(buffer);
  while((n = recv(connfd, buffer, bufsize, 0))>0){
      fputs("client", stdout)
      fputs("buffer", stdout)
      fputs("buffer", stdout)
  if(fgets(buffer, bufsize, stdin)!=NULL)
      {send(connfd, buffer, sizeof(buffer), 0);
          bzero(bufer, 1024);}

int main
{int cont, listfd, connfd, addrlen, addrlen2, fd, pid, addrlen3;
 struct sockaddr_in address, di_address;
 if(listenfd = socket(AF_INET, SOCK_STREAM, 0)>0)
      printf("created");
 address.sin_family = AF_INET;
 address.sin_addr.s_addr = INADDR_ANY;
 address.sin_port = htons(1600);
```

Teacher's Signature : _____

Name of Experiment..... Date..... Page No

Experiment No..... Experiment Result.....

```
bind (listenfd, (struct sockaddr *) &address, sizeof())
listen (listenfd, 2);
getsockname( listenfd, (struct sockaddr*)&address, &addrlen);
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_addrlen, &addrlen2);
  if (pid = fork ()) == 0)
  { printf ("child");
    close (listenfd);
    str_echo (connfd);
    exit(0);}
  close(connfd);
  return 0;

//client
void str_cli ( FILE *fp, int sockfd)
{ int bufsize = 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 : _____

Name of Experiment..... Date..... Page No

Experiment No..... Experiment Result.....

```
    fputs ("Me", stdout);
    print ("EOF");
}
int main (int argc, int *argv[])
{ int create_socket;
  struct sockaddr_in address;
  create_socket = socket (AF_INET, SOCK_STREAM, 0);
  address.sin_family = AF_INET;
  address.sin_port = htons (1600);
  inet_pton (AF_INET, argv[1], &address.sin_addr);
  if (connect (create_socket, (struct sockaddr *) &address, sizeof (address)
  
  sendfile (stdin, create_socket);
  return (close (create_socket));
}
```

Teacher's Signature : _____

Server

./server 127.0.0.1

Client1

Hi

Hi

C1

C1

Client2

Hi2

Hi2

C2

C2

Name of Experiment..... Date..... Page No

Experiment No..... Experiment Result.....

6) Concurrent and iterative echo using both connection oriented and connectionless.

i) Concurrent TCP

```
// Server
void str_echo (int sockfd)
{ ssize_t n;
  char buf[1000];
again:
  while ((n = read (sockfd, buf, 1000))>0)
  { write (sockfd, buf, n);
    memset (buf, '\0', strlen (buf));
  }
if (n<0 && errno == EINTR)
        goto again;
  else if (n==0)
    perror ("Str_echo: read");
int main (int argc, char * argv[])
{ int listenfd, connfd;
  pid_t childpid;
  socklen_t clilen;
  struct sockaddr_in cliaddr, servaddr;
  listenfd = socket (AF_INET, SOCK_STREAM, 0);
  bzero (&servaddr, sizeof (servaddr));
  servaddr.sin_family = AF_INET;
  servaddr.sin_addr.s_addr = htonl (INADDR_ANY);
  servaddr.sin_port = htons (SERV_PORT);
bind (listenfd, (struct sockaddr*) &servaddr, sizeof (addr));
```

Teacher's Signature : _____

Name of Experiment..... Date.....

Experiment No..... Experiment Result..... Page No

```
listen(listenfd, LISTENQ);
for(;;) {
  clilen = sizeof(cliaddr);
  connfd = accept(listenfd, (struct sockaddr *) &cliaddr, &clilen);
  if( (childpid=fork()) == 0)
  { close(listenfd);
    str_echo(connfd);
    exit(0);
  }
  close(connfd); } }

//client
void str_cli(FILE *fp, int sockfd)
{ char sendline[MAXLINE], recvline[MAXLINE];
  while (fgets(sendline, MAX, fp))
  { write(sockfd, sendline, strlen(sendline));
    if(read(sockfd, recvline, MAX) )
      fputs(recvline, stdout);
  }}

int main(int argc, char *argv[])
{ int sockfd; struct sockaddr_in servaddr;
  if(argc != 2) exit(0);
  sockfd = socket(AF_INET, SOCK_STREAM, 0);
  bzero(&servaddr, sizeof(servaddr));
  servaddr.sin_family = AF_INET;
  servaddr.sin_port = htons(SERV_PORT);
  inet_pton(AF_INET, argv[1], &servaddr.sin_addr);
```

Teacher's Signature : _____

Server

./server_it 127.0.0.1

Client ./client_it 127.0.0.1

Hi1

Hi1

C1

C1

^Z stop//

>> ./client_it 127.0.0.1

Hi2

Hi2

C2

C2

Name of Experiment..... Date..... Page No

Experiment No..... Experiment Result.....

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

2) Iterative TCP :

// Server
#DEFINE SERV_PORT 9002
#DEFINE LISTENQ 1024
void str_echo (int sockfd)
{ ssize_t n; char buf[1000];
again:
while ((n = read (sockfd, buf, 1000)) > 0)
{ write (sockfd, buf, n); memset (buf, '\0', strlen(buf)); }

int main (int argc, char * argv[])
{ int listenfd, connfd;
socklen_t clilen;
struct sockaddr_in cliaddr, servaddr;
listenfd = socket (AF_INET, SOCK_STREAM, 0);
bzero (&servaddr, sizeof (servaddr));
servaddr.sin_family = AF_INET;
servaddr.sin_addr.s_addr = htonl (INADDR_ANY);
servaddr.sin_port = htons (SERV_PORT);
bind (listenfd, (struct sockaddr *) &servaddr, sizeof (servaddr));
listen (listenfd, LISTENQ);
for (;;)
{ clilen = sizeof (cliaddr);

Teacher's Signature : _____

Name of Experiment..... Date..... Page No.

Experiment No..... Experiment Result.....

```
    connfd = accept (listenfd, (struct sockaddr*) &cliaddr, &clilen);
    str_echo (connfd);
    close (connfd); }

//Client
#define MAXLINE 1000
#define SERV_PORT 9002
void str_cli(FILE *fp, int sockfd)
{ char sendline[MAX], recvline[MAX];
  while (sockfd, sendline, strlen(sendline));
    fp read (sockfd, recvline, MAX)
      fputs (recvline, stdout);
    }
}
int main (int argc, char* argv[2])
{ int sockfd;
  struct sockaddr_in servaddr;
  sockfd = socket (AF_INET, SOCK_STREAM, 0);
  bzero (&servaddr, sizeof (servaddr));
  servaddr.sin_port = htons (SERV_PORT);
  inet_pton (AF_INET, argv[1], &servaddr.sin_addr);
  connect (sockfd, (struct sockaddr*) &servaddr, sizeof (servaddr));
  str_cli (stdin, sockfd);
    exit (0);
}
```

Teacher's Signature : _____

Server:

./server 127.0.0.1

Client

./client 127.0.0.1

Hi

Hi

Hello

Hello

Name of Experiment..... Date..... Page No

Experiment No..... Experiment Result.....

3) UDP echo server

```
// Server
#define SERV_PORT 9002
#define MAX 1024
void dg_echo (int sockfd, struct sockaddr *pcliaddr, socklen_t clila)
{ int n; socklen_t len;
  char msg [MAX];
  for (;;)
  { len = clilen;
    n = recvfrom (sockfd, msg, MAX, 0, pcliaddr, &len)
    sendto (sockfd, msg, strlen(msg), 0, pcliaddr, len);
  }};

int main (int argc, char * * argv)
{ int sockfd; struct sockaddr_in servaddr;
  sockfd = socket (AF_INET, SOCK_DGRAM, 0);
  bzero (&servaddr, sizeof (servaddr));
  servaddr.sin_family = AF_INET;
  servaddr.sin_addr.s_addr = htonl (INADDR_ANY);
  servaddr.sin_port = htons (SERV_PORT);
  bind (sockfd, (struct sockaddr *)&servaddr, sizeof (servaddr));
  dg_echo (sockfd, (struct sockaddr *) &cliaddr, sizeof (cliaddr));
}
```

Teacher's Signature : _____

Name of Experiment..... Date..... Page No.

Experiment No..... Experiment Result.....

```
// Client
void dg_cli(FILE *fp, int sockfd, const struct sockaddr *
            pservaddr, socklen_t servlen)
{ int n; char sendline[MAX], recvline[MAX];
  while(fgets(sendline, MAX, fp) != NULL
    { sendto(sockfd, sendline, strlen(sendline), 0, pservaddr,
             servlen);
      n = recvfrom(sockfd, recvline, MAX, 0, NULL, NULL);
      recvline[n] = 0;
      fputs(recvline, stdout);
    }
}
int main(int argc, char *argv[])
{ int sockfd; struct sockaddr_in servaddr;
  bzero(&servaddr, sizeof(servaddr));
  servaddr.sin_family = AF_INET;
  servaddr.sin_port = htons(SERV_PORT);
  inet_pton(AF_INET, argv[1], &servaddr.sin_addr);
  sockfd = socket(AF_INET, SOCK_DGRAM, 0);
  dg_cli(stdin, sockfd, (struct sockaddr *) &servaddr,
         sizeof(servaddr));
  exit(0);
}
```

Teacher's Signature : _____

Server:

Socket created
Binding socket
Client 127.0.0.1 connected
Comand : ls
client client.c
server server.c
Comand : date
Mon, Jan 11
//end

Client:

Socket created
connection accepted 127.0.0.1
Enter command : ls
Enter command : date
Enter command : end

Name of Experiment..... Date..... Page No

Experiment No..... Experiment Result.....

7) Remote command execution

// Server

```
int main()
{ int conf, create_socket, new_sok, addrlen, fd;
  int bufsiz = 1024; char * buffer = malloc (bufsize);
  char recv [256]; struct sockaddr_in address;
  create_socket = socket(AF_INET, SOCK_STREAM, 0)
  address.sin_family = AF_INET;
  address.sin_addr.s_addr = INADDR_ANY;
  address.sin_port = htons (15000);
  bind (create_socket, (struct sockaddr *) & address, sizeof (address))
  listen (create_socket, 3);
  addrlen = sizeof (struct sockaddr_in);
  new_socket = accept( create_socket, (struct sockaddr *) & address,
                                                    (addrlen);
  int bytes;
  while ((bytes = recv (new_socket, receiver, 255, 0)) > 0)
  {receiver [bytes] = '\0';
    if (strcmp (receiver, "end") == 0)
    { close (new_socket);
      close (create_socket);
      exit (0); }
    else
      system (receiver);
  }
  close (new_socket);
  close (create_socket);
}
```

Teacher's Signature : _____

Name of Experiment.....

Experiment No.....

Date.....

Experiment Result.....

Page No

```
//Client

int main (int argc, char *argv[])
{ int create_socket, int bufsize = 1024;
  char *buffer = malloc(bufsize);
  char com[256];
  struct sockaddr_in address;
  create_socket = socket (AF_INET, SOCK_STREAM, 0);
  address.sin_family = AF_INET;
  address.sin_port = htons (15000);
  inet_pton (AF_INET, argv[1], &address.sin_addr);
  connect (create_socket, (struct sockaddr *) &address, sizeof (address))
  while(1)
  { printf("%s", com);
    send (create_socket, com, sizeof (com), 0);
  }
  close (create_socket);
}
```

Teacher's Signature : _____

RSA ./rsa

Enter text : Hello

p and q = 13 and 23

n = 299 phi = 264

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

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

Encrypted message :

137 51 147 147 227

Decrypted message :

hello.

Diffie :

Alice : 6 Bob : 15

~~private key : 8~~

Alice's public key : 8

Bob's public key : 19

Alice's secret key : 2

Bob's secret key : 2

Name of Experiment..... Date..... Page No

Experiment No..... Experiment Result.....

8) Encrypt and decrypt using RSA and exchange key securely using Diffie-Hellman key exchange protocol.

```
//RSA
#include <stdio.h>
#include <math.h>
#include <stdlib.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 (i = 2; i<a; i++)
  { if (a%i == 0)
      return 0; }
  return i;
}
  long int encrypt ( long int ch, long int n, long int e)
  { long int i; temp = ch;
    for (i=1; i<e; i++)
    { temp = (temp * (ch)) %n;
      return temp%n; }
```

Teacher's Signature : _____

Name of Experiment..... Date..... Page No

Experiment No..... Experiment Result.....

```
long int decryptt( long int ch, long int n, long int dd)
{   long int i, temp = ch;
    for (i=1; i<dd; i++)
    {   temp = (temp * ch) % n;
    return temp;
}
int main()
{ long int i, int len; long int p, q, n, phi, e, d, cipher[50];
  char text[50];
  scanf("%s", text);
  len = strlen(text)
  do {
      p = rand()%30; }
  while(!isprime(p));
  do { q = rand()%30;
    } while(!isprime(q));
  n = p*q; phi = (p-1)(q-1);
  do {
      e = rand()%phi; } while(gcd(phi, e)!=1);
  do { d = rand()%phi
    } while((d*e)%phi)!=1);
  // print all of them
  for(i=0; i<len; i++)
  { cipher[i] = encrypt(text[i], n, e);
    printf("%d", cipher[i]);
```

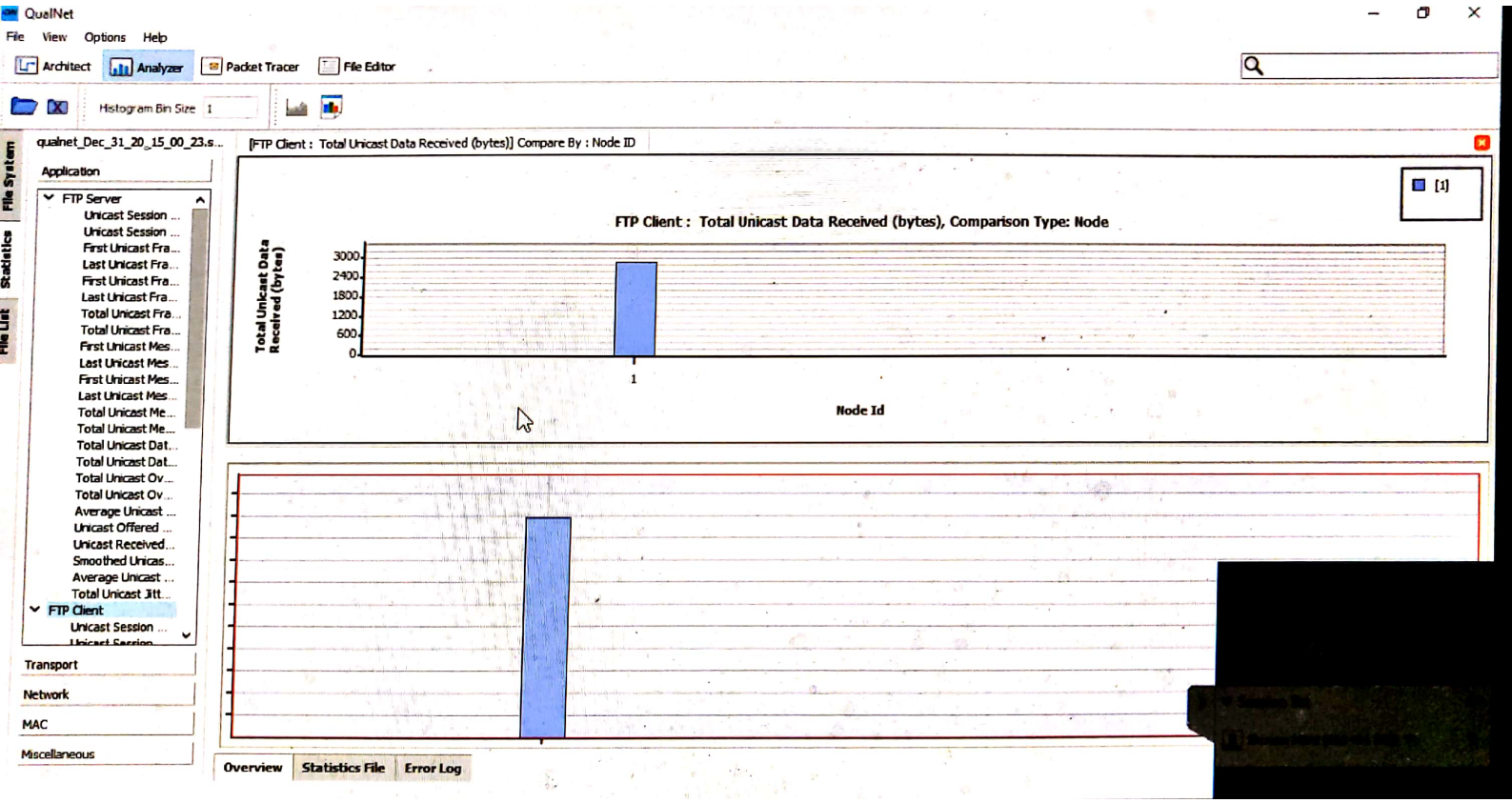
Teacher's Signature : _____

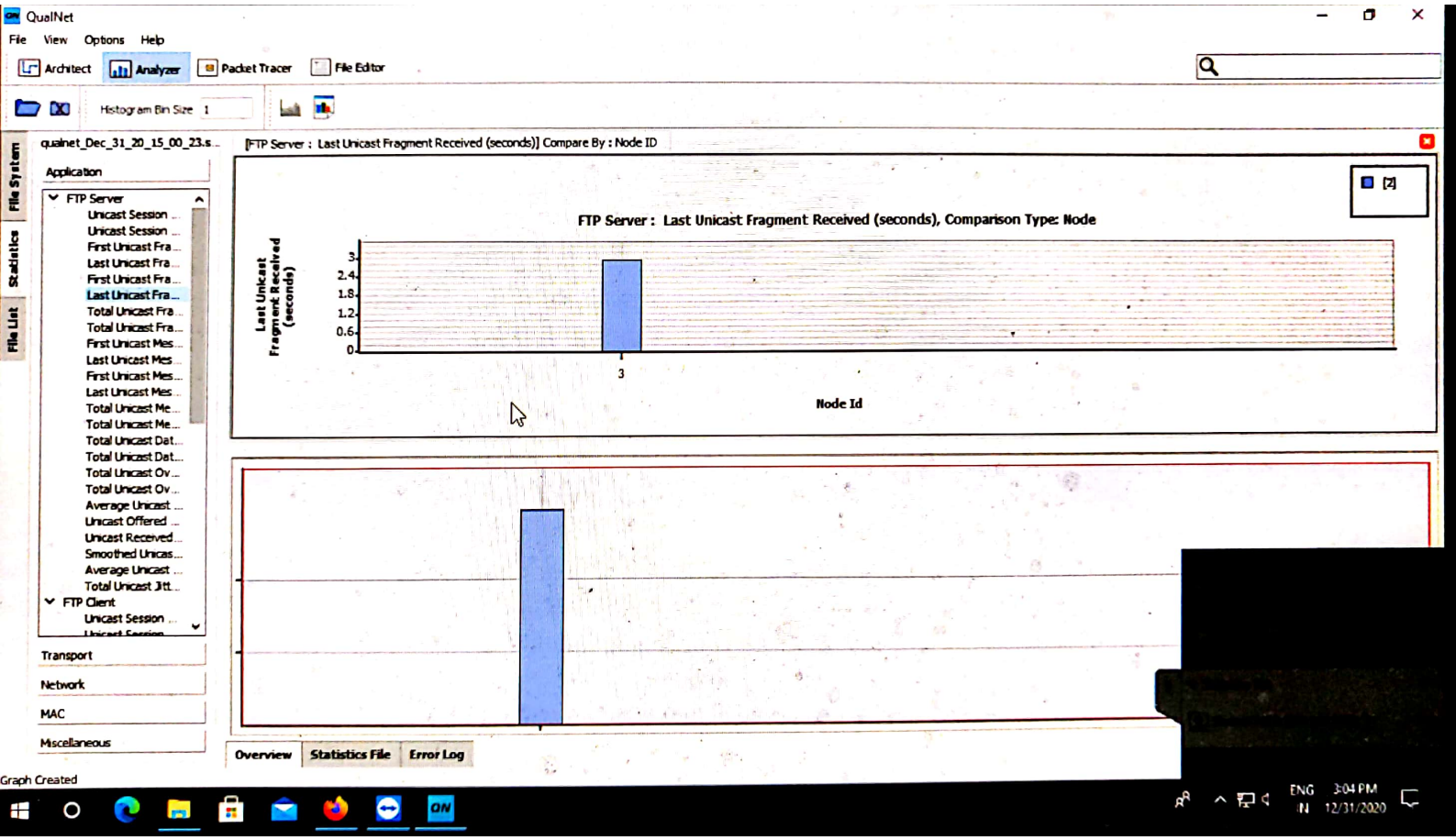
Name of Experiment..... Date..... Page No

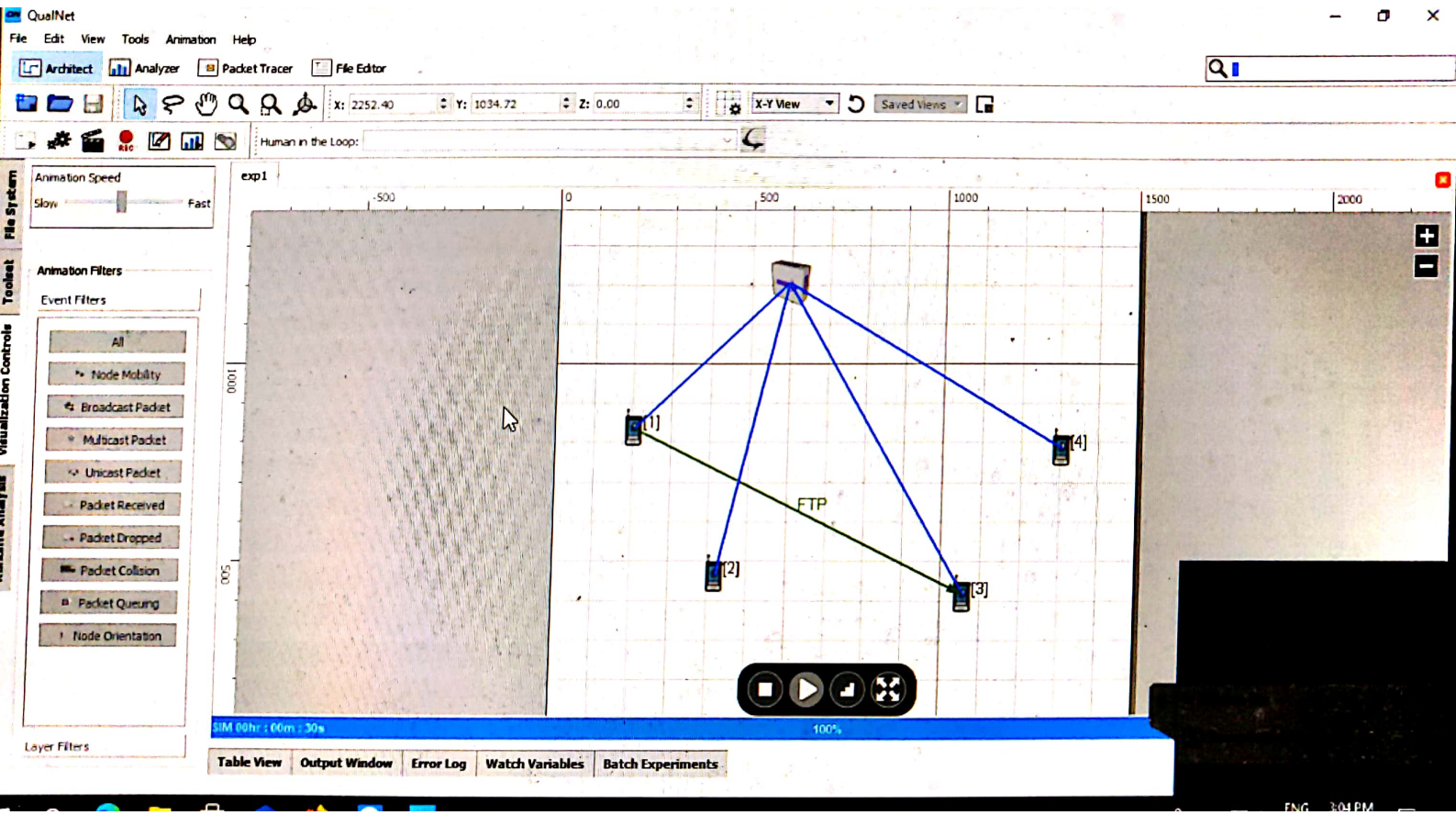
Experiment No..... Experiment Result.....

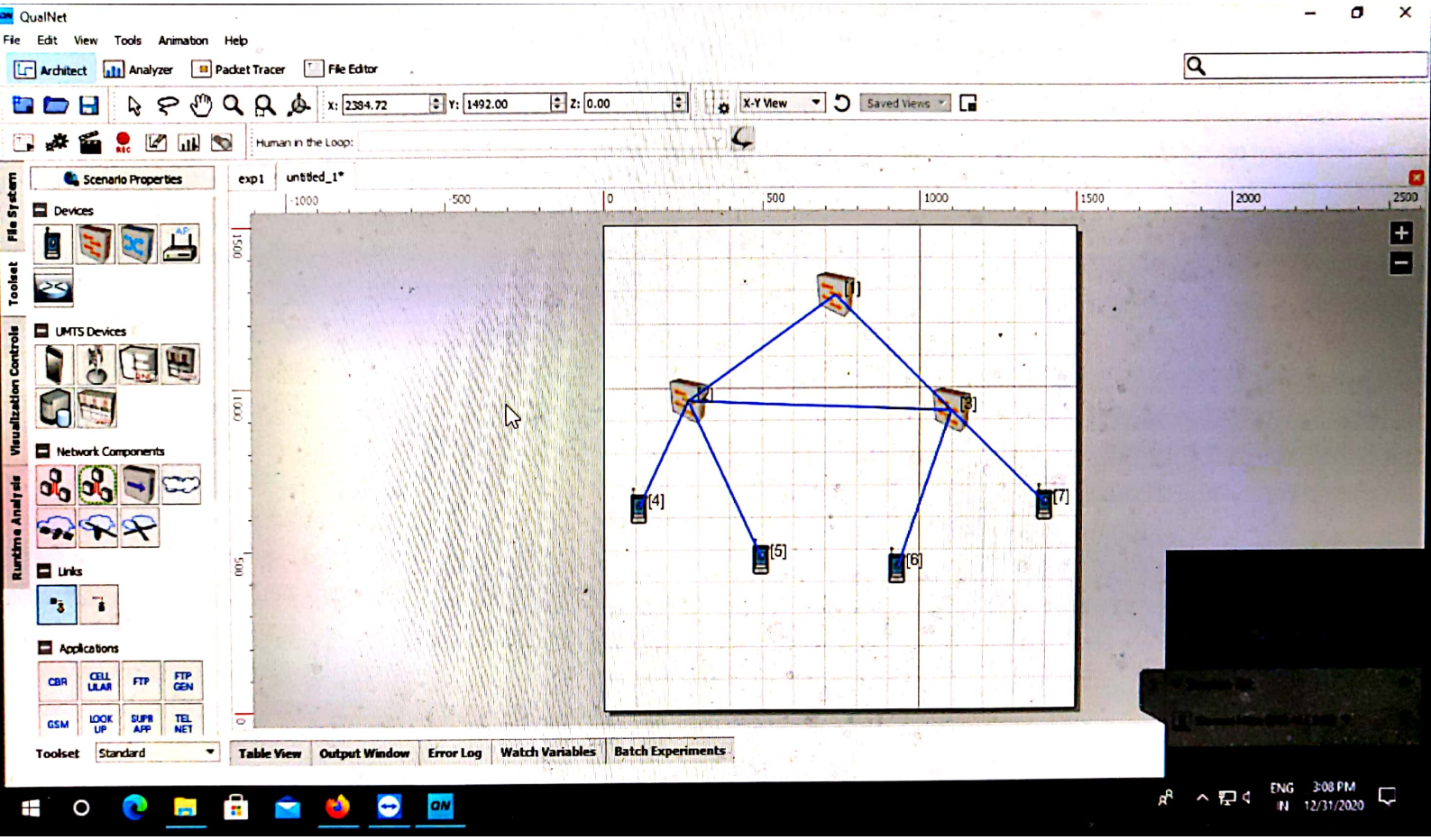
```
for (i=0;i<len;++i)
{ text[i] = decrypt(cipher[i], n, d);
    printf("%c", text[i]);
}
return 0;
}
//Diffie Hellman

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 h, G, x, a, y, b, ka, kb;
  h=23;
  G=9;
  a=4;
  x=power(G, a, h);
  b= 3;
  y = power(G, b, h);
  ka = power(y, a, h);
  kb = power(x, b, h);
  printf("%lld %lld", ka, kb);

  return 0;
}
```

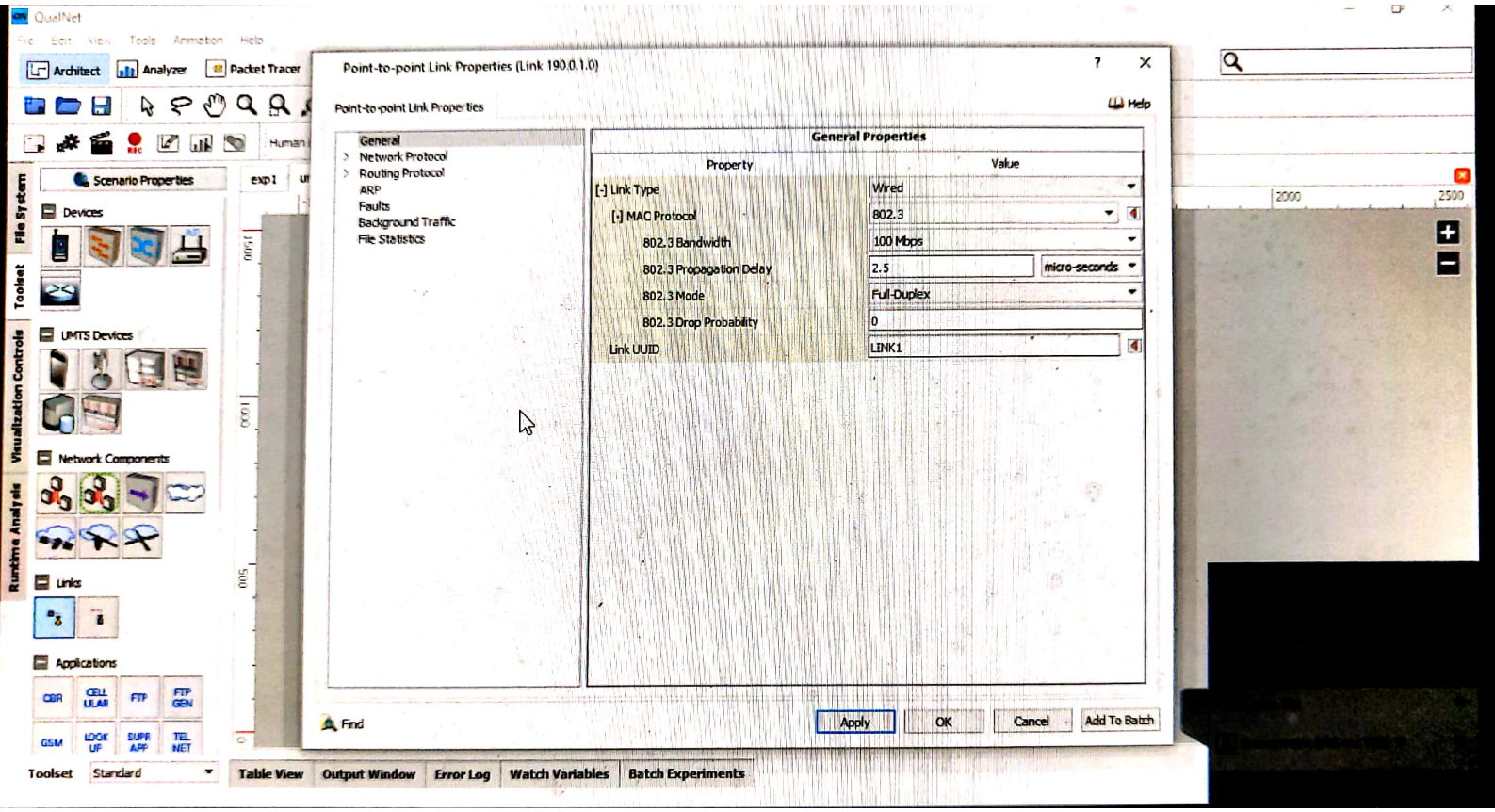
Teacher's Signature : _____

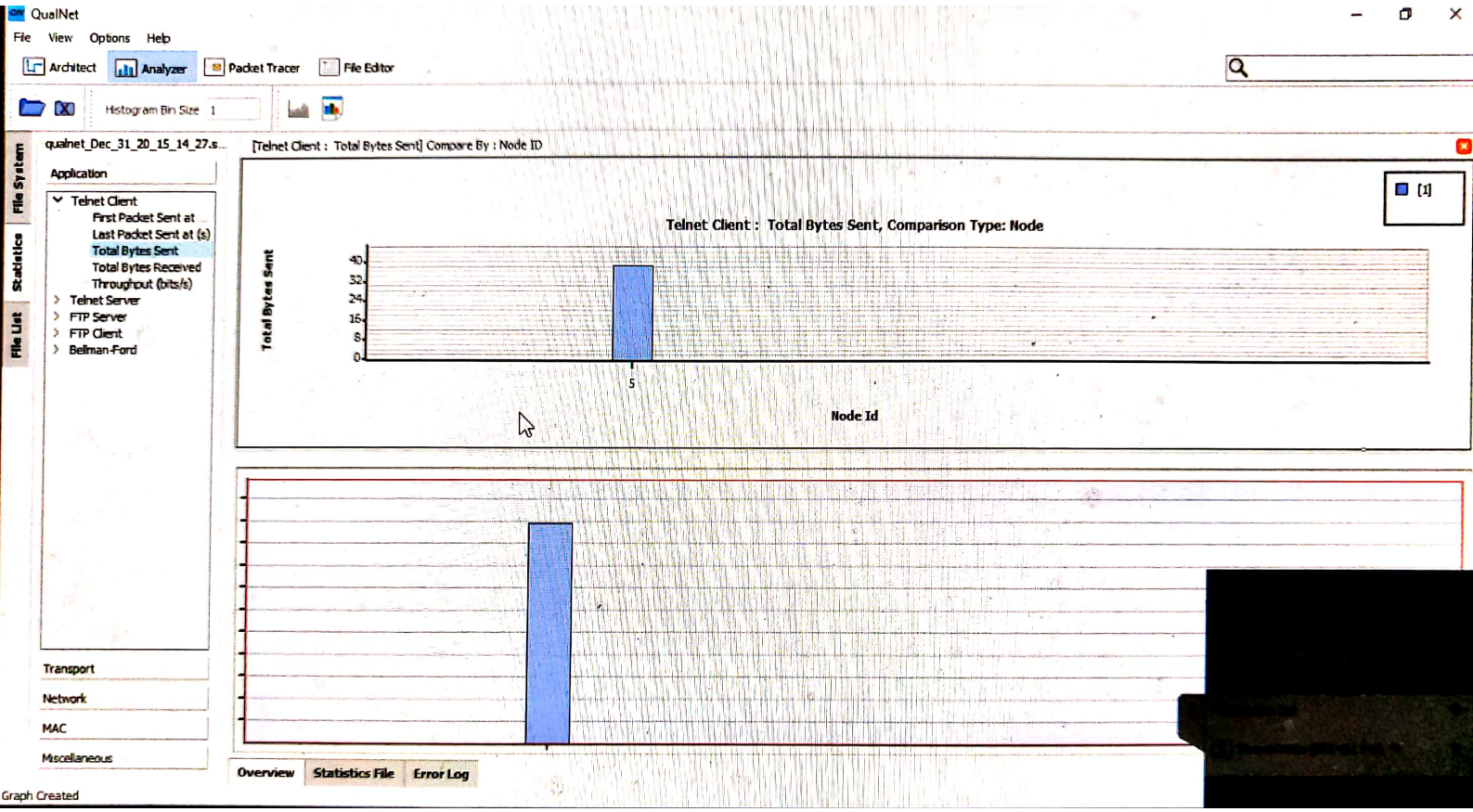


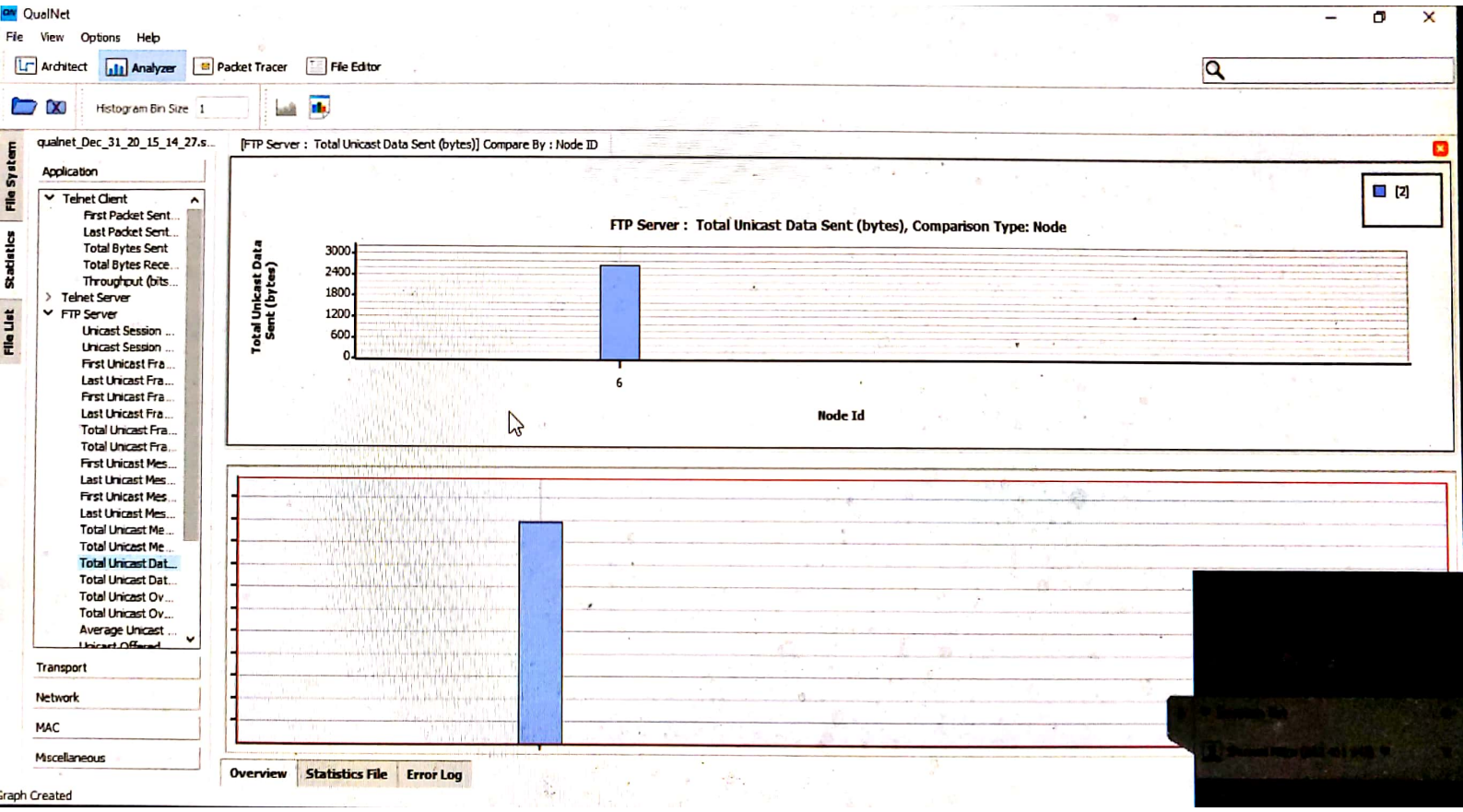












Name of Experiment..... Date..... Page No

Experiment No..... Experiment Result.....

PART-B

1) Setup an IEEE 802.3 network with a) hub b) switch c) Hierarchy of switch. Apply FTP, Telnet application between nodes. Vary the bandwidth, queue size and observe the packet drop.

① → design of layout with one hub / switch connected to four devices via connecting lines

② → Graph of Total Unicast Data Sent (Y) Node ID (Bytes)

③ → Graph of Total Unicast Data received (Y) Node ID (Bytes)

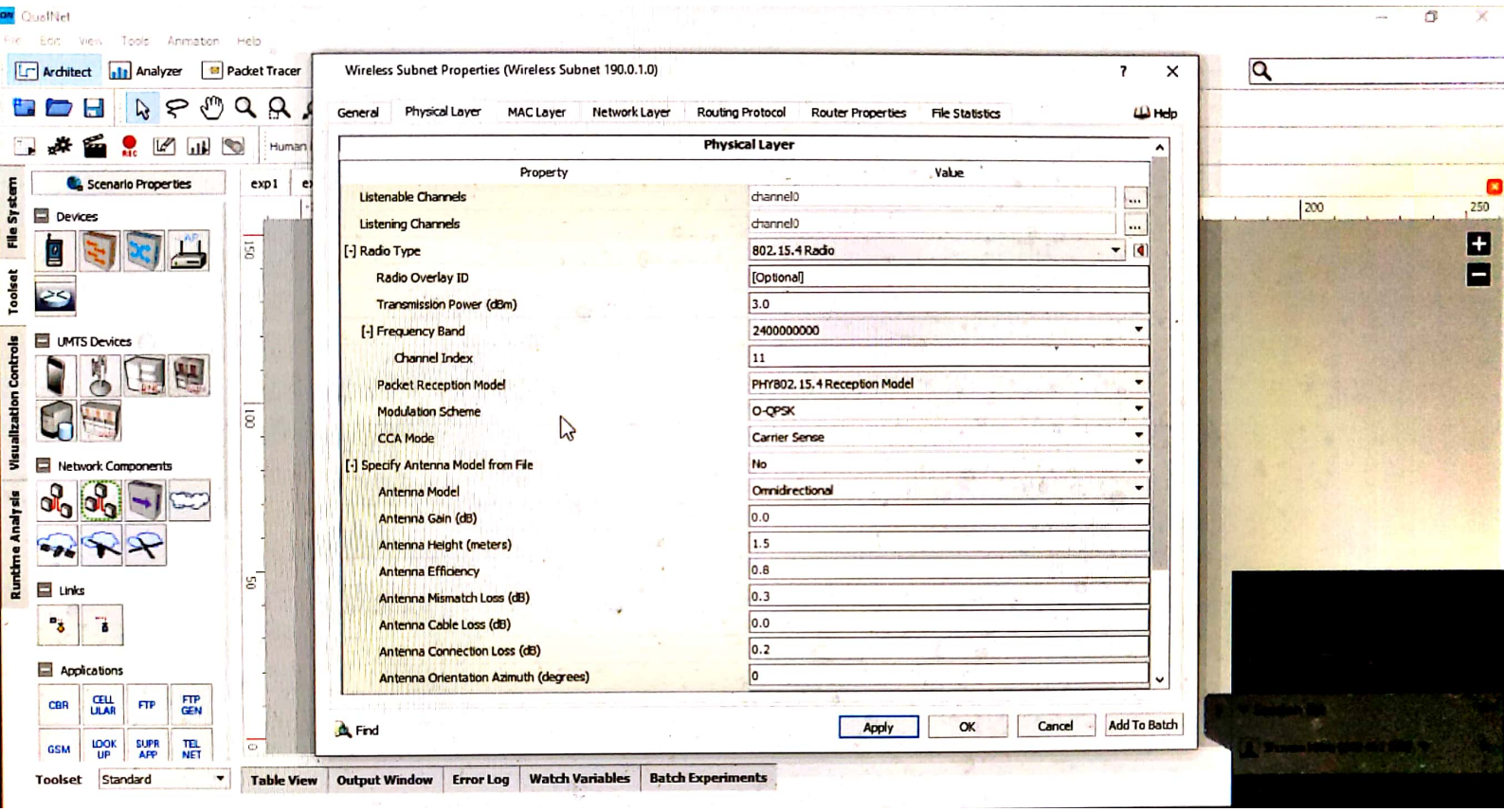
④ → Design layout of three switches connected to each other and to four devices, out of which FTP is applied on 4 and 6 devices and TELNET on 5 & 7

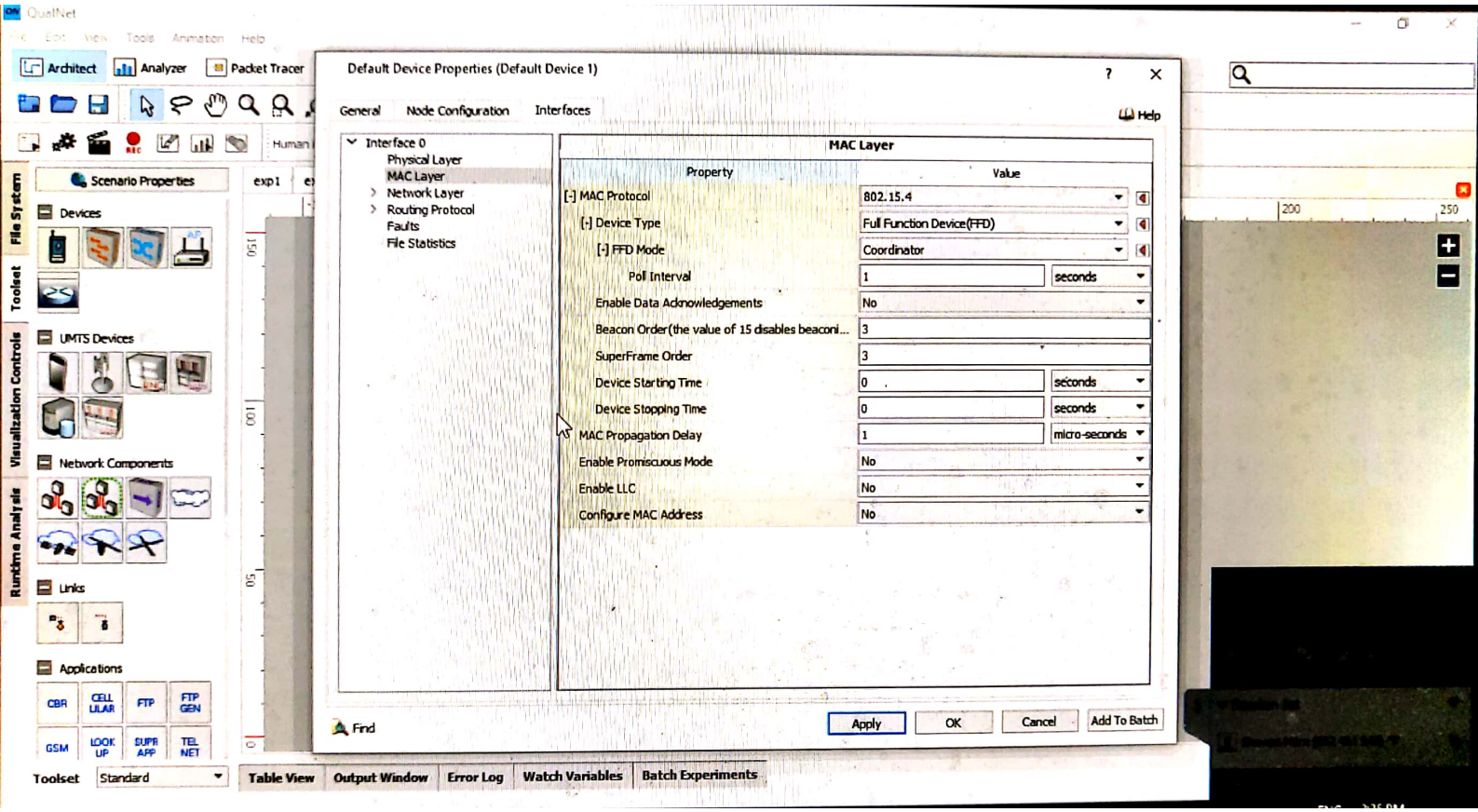
⑤ → Configuring the connecting lines to 802.3 MAC Protocol

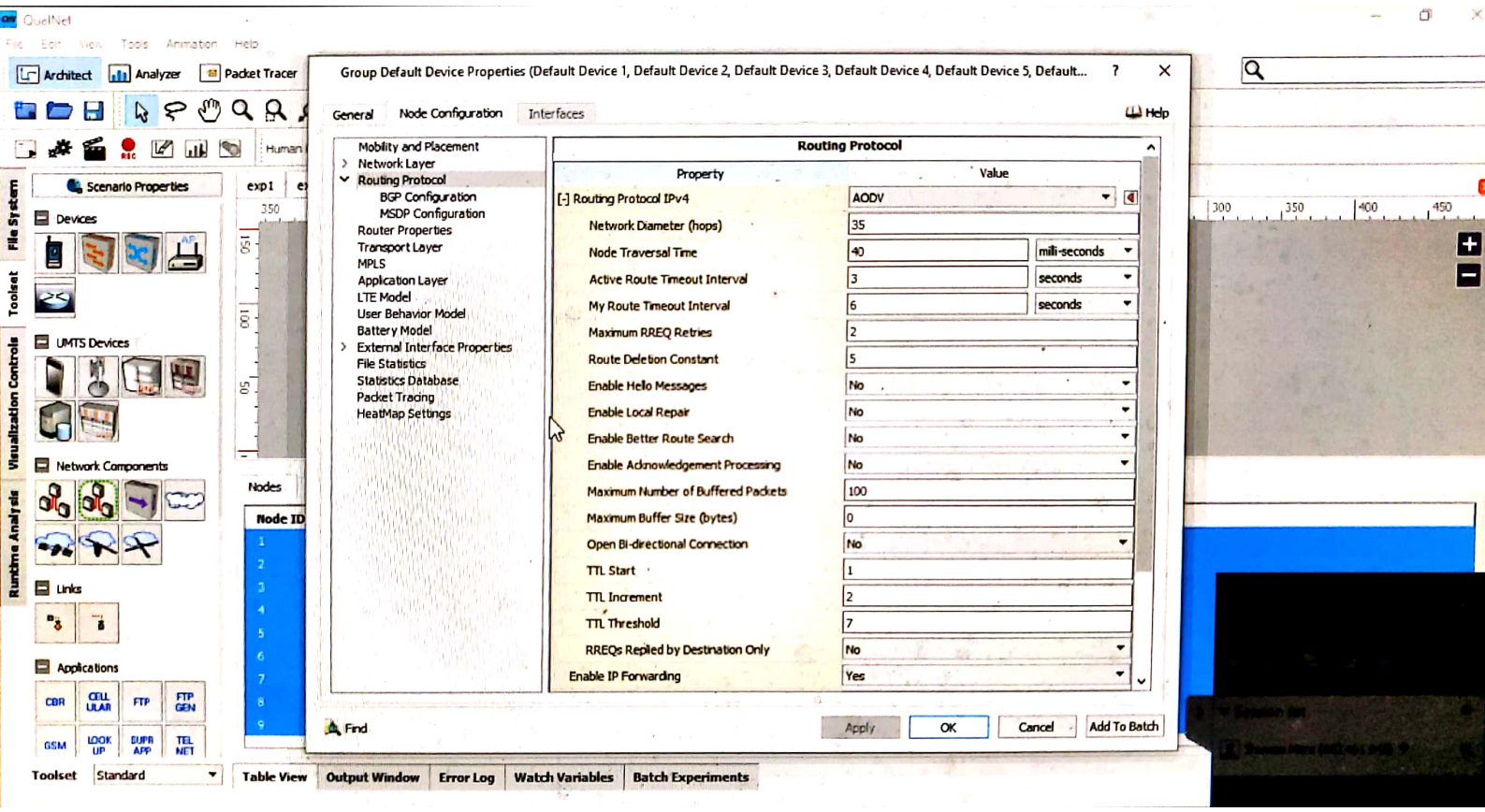
⑥ → Checking the config properties of the devices and ensuring it is 802.3 (MAC)

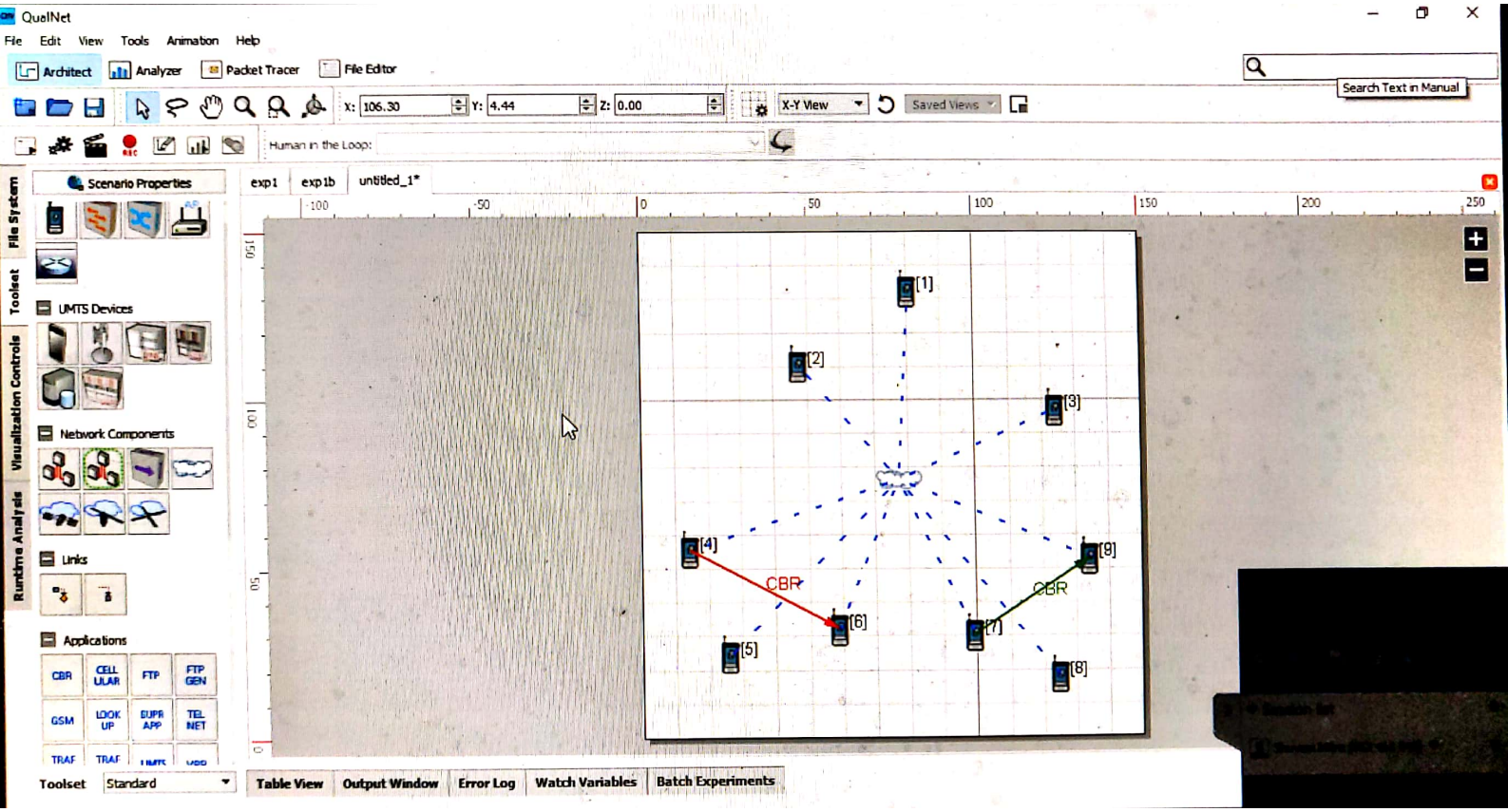
⑦ → FTP Unicast data sent (Y) Node ID (Bytes)

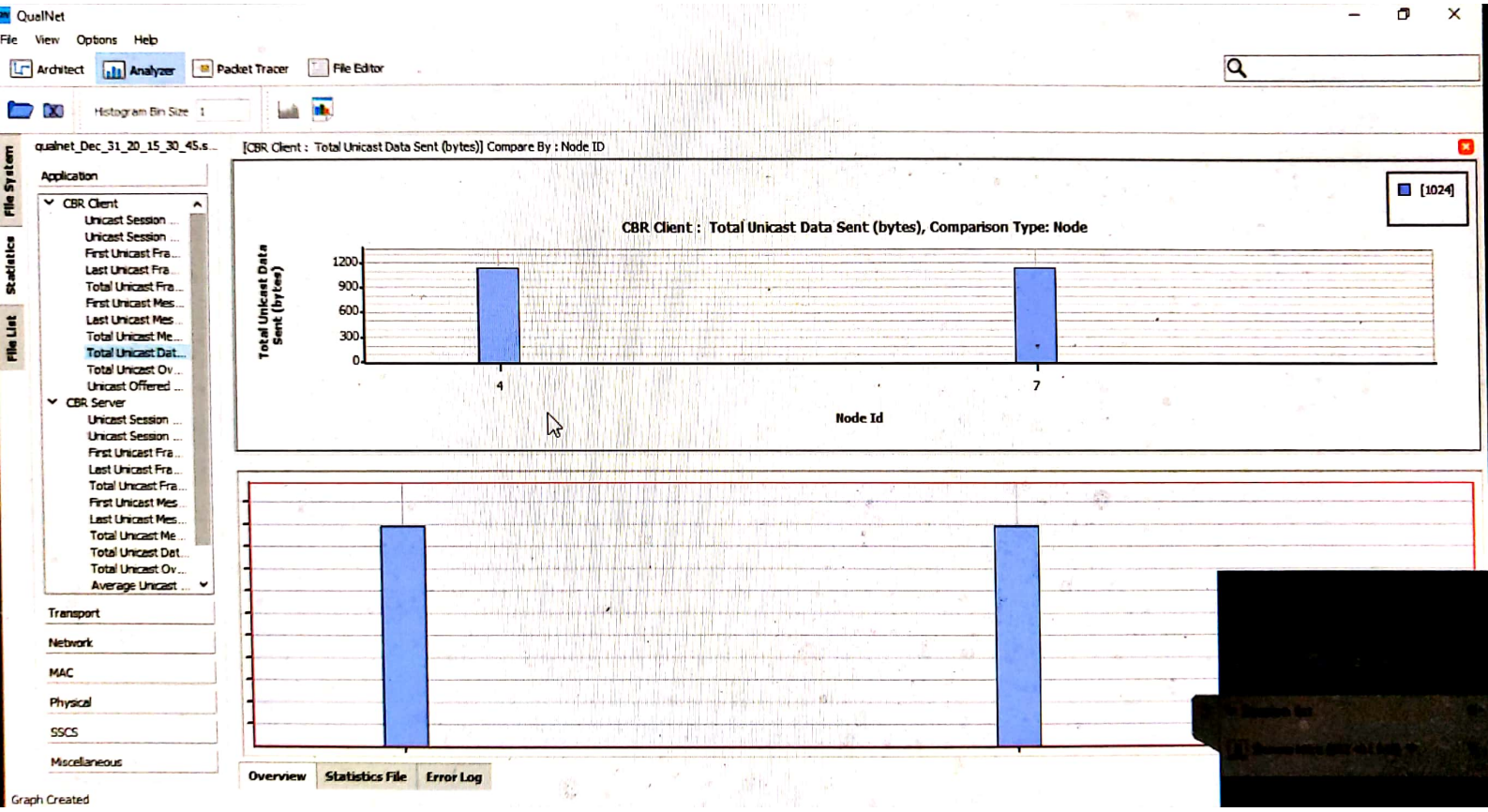
Teacher's Signature : _____

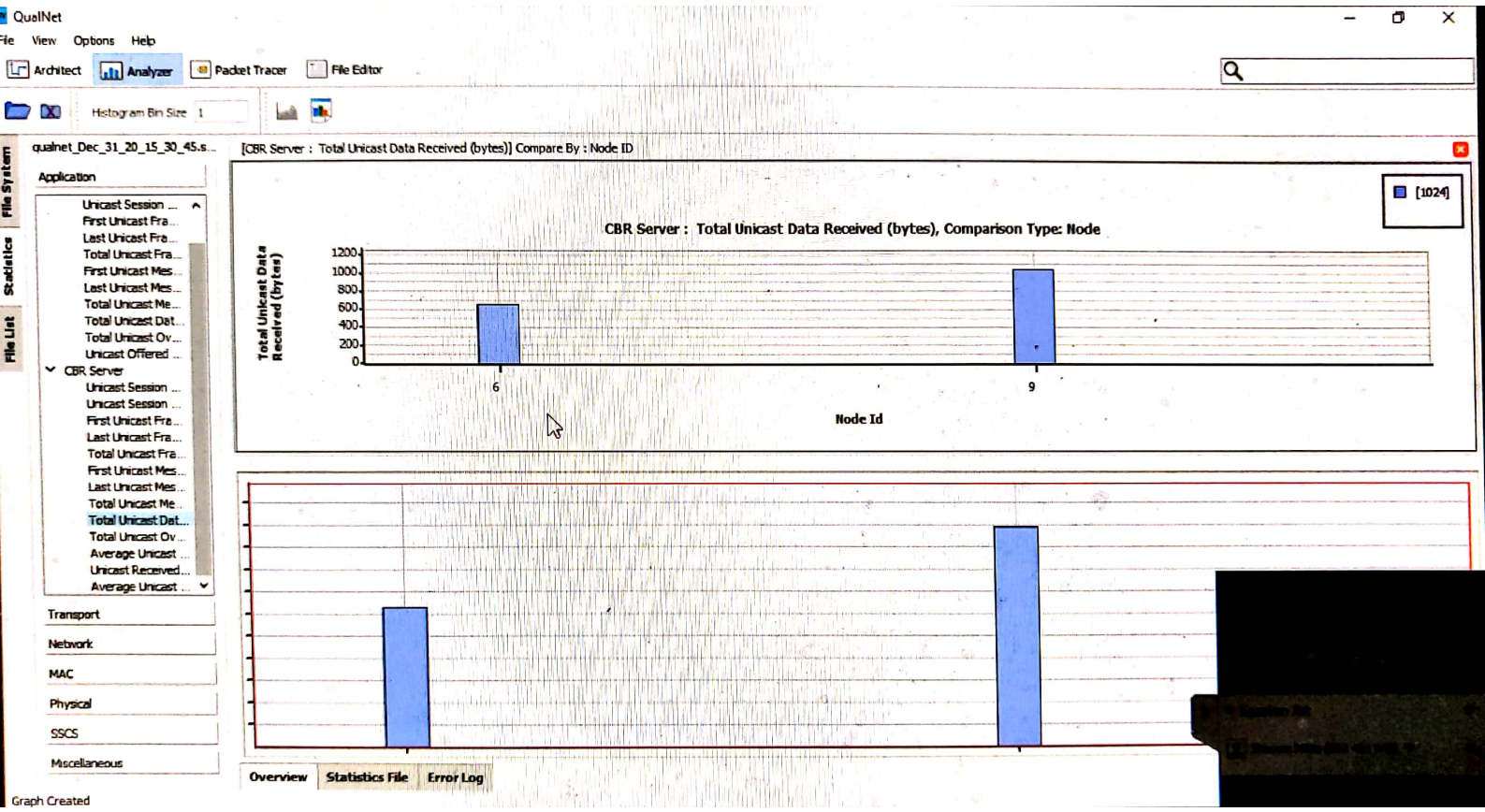












Name of Experiment..... Date..... Page No

Experiment No..... Experiment Result.....

2) Setup an IEEE 802.11 network with atleast two access points. Apply the CBR, VBR applications between devices belonging to same access points 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.

① → Design layout of the required network with 2 access points

② → Application of CBR protocol to send 6 devices and creating a path.

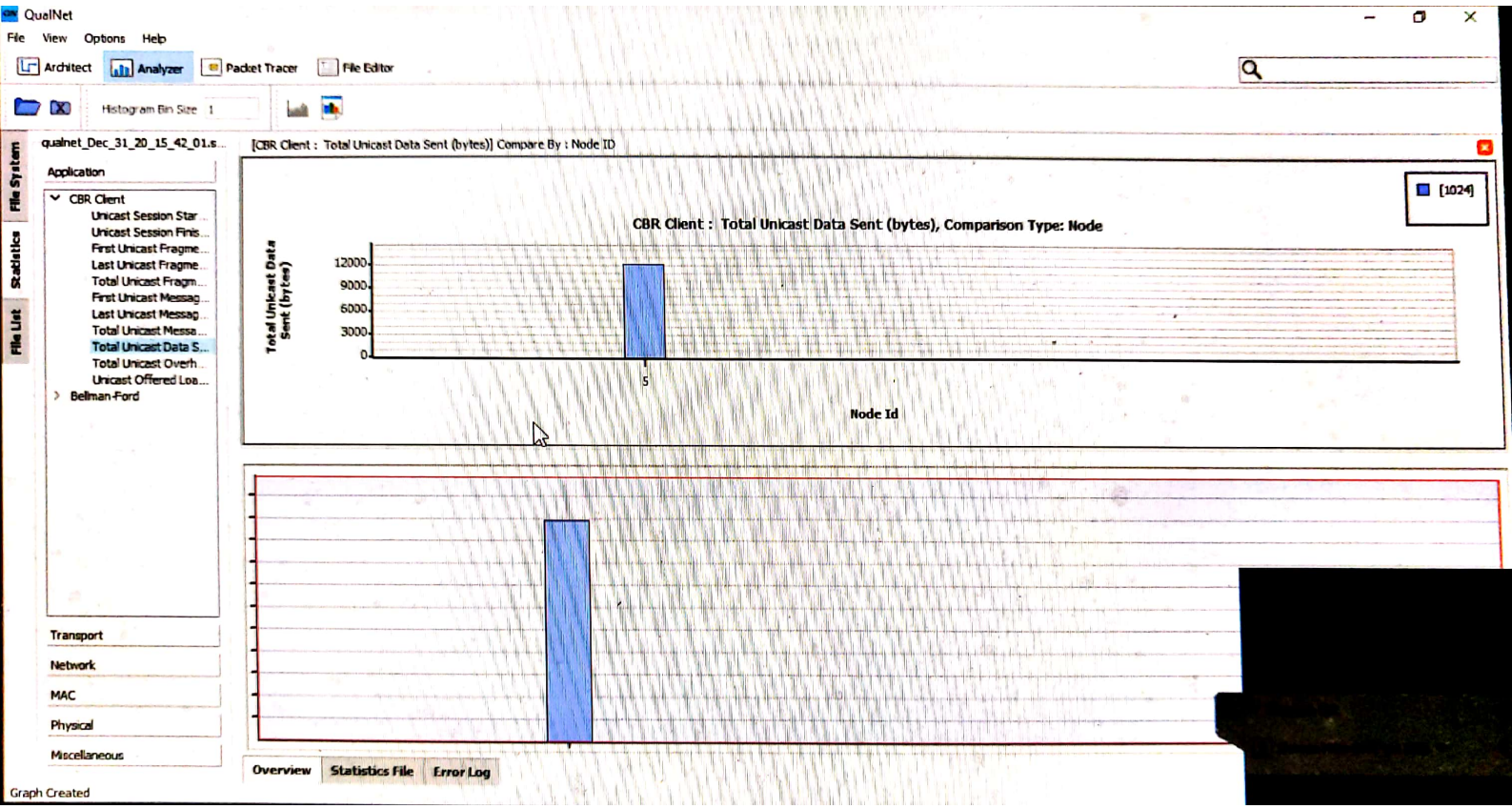
③ → Watching the packets transfer in network.

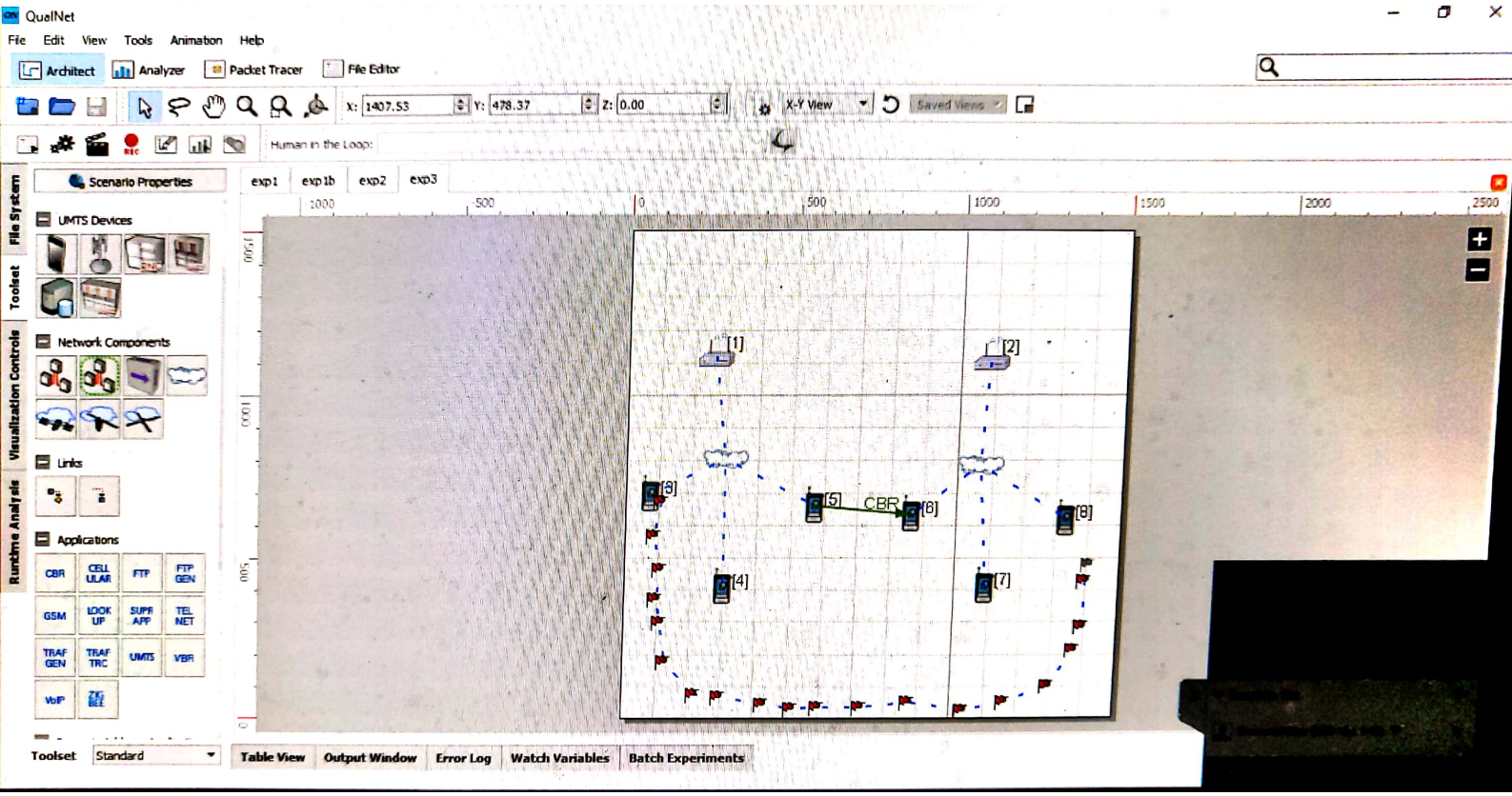
④ → Configuration parameters

⑤ → Graph of CBR clients Against access points

⑥ Node IDs

Teacher's Signature : _____





Name of Experiment..... Date..... Page No

Experiment No..... Experiment Result.....

3) Setup a wireless sensor network with atleast two device co-ordinators and nodes. Provide Constant Bit Rate (CBR), Variable bit rate (VBR) applications between several nodes. Increase no. of co-ordinators and nodes in the same area and observe the performance of Physical and MAC layers.

① → Design layout for the required network with one main co-ordinator and two PAN co-ordinate.

② → Upload data sent in Zigbee ⓥ Node ID (Zigbee)

Teacher's Signature : _____