PART - A

Experiment No. 1 | Date
Name of the Experiment | Page No.

1) Implement a client & server communication using socket programming.

client.c

```
#include <stdio.h>
#include <netdb.h>
#include <stdlib.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#define MAX 80
#define PORT 6700
#define SA struct sockaddr
void func(int sockfd)
{ char buff[MAX]; int n=0;
  for(;;)
  { bzero(buff,sizeof(buff));
    printf("Enter the string: ");
while((buff[n++]=getchar())!='\n');
write(sockfd,buff,sizeof(buff));
bzero(buff,sizeof(buff));
read(sockfd,buff,sizeof(buff));
pf("from server: %s",buff);
if((strncmp(buff,"exit",4))==0)
{ pf("client exit....\n");
  break; }}}
int main()
{ int sockfd,confd;
  struct sockaddr_in servaddr,cli;
```

chandra's

Name of the Experiment Page No.

```
sockfd = socket (AF_INET, sock_STREAM,0);
if (sockfd==-1)
{ pf("socket creation failed---\n");
  exit(0); }
else
  pf ("socket successfully created---\n");
bzero(&servaddr, sizeof(servaddr));
servaddr.sin_family = AF_INET;
servaddr.sin_addr.s_addr=inet_addr("127.0.0.1"
servaddr.sin_port = htons(PORT);
if (connect(sockfd, (SA*)&servaddr, size(serv
                                    addr))!=0)
{ pf ("connect with server failed");
else {
  pf ("connected to server");
  fun(sockfd);
  close(sockfd); }
```

server.c

```
#include <stdio.h>
#include <refds.h>
#include <netinet/in.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/types.h>
#define MAX 80
#define PORT 5700.
```

chandra's

Experiment No. | Date
Name of the Experiment | Page No.

```
#define SA struct sockaddr.
void func (int sockfd);
{ char buff[MAX]; int n=0;
  for(;;)
  { bzero (buff, MAX);
    read (sockfd, buff, sizeof (buff));
    pf(" from client : %s \n to client: ", buff);
    bzero (buff, MAX);
    write (sockfd, buff, sizeof (buff));
    if (strncmp ("exit", buff, 4) == 0)
    { pf (" server exit ---- \n");
      break; } }
int main()
{ int sockfd, connfd, len;
  struct sockaddr_in servaddr, cli;
  sockfd = socket (AF_INET, SOCK_STREAM, 0);
  if (sockfd == -1)
  { pf ("socket creation failed ");
    exit (0);
  }
  else
    pf (" socket created successfully ");
bzero (& servaddr, sizeof (servaddr));
servaddr.sin_family = AF_INET;
seraddr.sin_addr.s_addr = htonl (INADDR_ANY
servaddr.sin_port = htons (PORT);
```

chandra's

Name of the Experiment

```
if((bind(sockfd,(SA*)&servaddr,sizeof(servaddr))!=0)
{ pf("socket bind failed");
  exit(0);}
else
  pf("socket binded successfully");
if ((listen(sockfd,5))!=0)
{ pf("listen failed");
  exit(0); }
else
  pf("server listening");
  len=sizeof(cli);
connfd=accept(sockfd,(SA*)&cli,&len);
 if (connfd<0)
{ pf("server accept failed");
  exit(0); }
 else
  pf("server accept the client");
func(connfd);
 close(sockfd);
}
```

Output

```
cc client.c -o cli
./cli
socket successfully created...
connected to server
enter the string: hii
from server : hi!
enter the string: 12345
from server : hihi, hello!
enter string
```

Output

```
cc server.c -o serv
./serv
Socket successfully created.....
socket successfully binded.....
server listening.....
server accept the client
From client: hii
To client : hi!
From client: 12345
To client : hii, hello.
```

Experiment No. 2 Date

Name of the Experiment Page No.

2) WAP to implement distance vector routing protocol for a simple topology of routers.

```
#include <stdio.h>
struct node
{ unsigned dist[20];
  unsigned from[20];
} rt[10];
int main()
{ int costmat[20][20];
  int nodes, i, j, k, count=0;
  printf("enter the no. of nodes");
  scanf("%d", &nodes);
  printf("enter the cost matrix");
  for(i=0; i<nodes; i++)
  { for(j=0; j<nodes; j++)
  { scanf("%d", &costmat[i][j]);
    costmat[i][i]=0;
    rt[i].dist[j]= costmat[i][j];
    rt[i].from[j]=j; }}
  do
  { count=0;
    for(i=0; i<nodes; i++)
      for(j=0; j<nodes; j++)
        for(k=0; k<nodes; k++)
  if(rt[i].dist[j]> costmat[i][k]+
      rt[k].dist[j])
  { rt[i].dist[j] = rt[i].dist[k]+ rt[k].dist[j];
```

chandra's

Experiment No. | Date
Name of the Experiment | Page No.

```
rt[r].from[i]=t;
  count++;}}
while(count!=0);
for(i=0;i<nodes;i++)
{pf("for router %d",i+1);
 for(j=0;j<nodes;j++)
 {pf("node %d via %d distance %d",j+1,rt[j].
    from[j]+1,rt[i].dist[j]);
}}
  pf("\n");
}
```

output

```
cc p2.c -o p2

./p2
enter the no of nodes: 3
enter the cost matrix: 0 10 4
                       10 0 3
                       4 3 0

for router 1
  node 1 via 1 distance 0
  node 2 via 3 distance 7
  node 3 via 3 distance 4
for router 2
  node 1 via 3 distance 7
  node 2 via 2 distance 0
  node 3 via 3 distance 3

for router 3
  node 1 via 1 distance 4
  node 2 via 2 distance 3
  node 3 via 3 distance 0
```

Q3 write a prog to implement error detection & correction concept using check sum & hamming code

```
#include<stdio.h>
unsigned fields[10];
unsigned short checksum()
{ int i, sum=0;
printf("enter the IP header into in 16 bit words");
for(i=0;i<9;i++)
{ printf("field %d:",i+1);
scanf("%x",&field[i]);
sum=sum+(unsigned short fields[i]);
while(sum>>16)
sum = sum & 0xFFFF) + (sum>>16);}
sum=~sum;
return (unsigned short) sum;

int main()
{ unsigned short result1, result2;
result1 = checksum();
printf("computed checksum at sender
%x", & result+1);
result2 = checksum();
printf("computed checksum at receiver
%x", result2);
if(result1 == result2){ printf("No error\n");
else{ printf("oh no! error detected");}
```

chandra's

Name of the Experiment: Hamming code

```
#include<stdio.h>
void main()
{ int data[10];
  int data_atrec[10],c,c1,c2,c3,i;
  pf("sender");
  sf("%d %d %d %d",&data[0],&data[1],
     &data[2],&data[4]);
data[6]=data[0]^data[2]^data[4];
data[5]=data[0]^data[1]^data[4];
data[3]=data[0]^data[1]^data[2];
pf("Encoded data");
for(i=0;i<7;i++)
    pf("%d",data[i]);
 pf("receiver)
 for(i=0;i<7;i++)
    sf("%d",&datagatrec[i]);
c1=data_atrec[6]^data_atrec[4]^
   data_atrec[2]^data_atrec[0];
c2=data_atrec[5]^data_atrec[4]^
   data_atrec[1]^data_atrec[0];
c3=data_atrec[3]^data_atrec[2]^
   data_atrec[1]^data_atrec[0];
  c=c3*4+c2*2+c1;
pf("syndrome bits:%d %d %d",c1,c2,c3);
 if(c==0)
 { pf("No error while transmission
      of data");}
```

chandra's

```
else
{ Pf ("error position: %d", c);
  pf ("data sent");
  for(i=0; i<7; i++)
    pf ("%d", data[i]);
  pf ("data received");
  for (i=0; i<7; i++)
    pf("%d", data-atrecv[i]);
  pf ("correct msg is:");
if (dataat rec[7-c]==0)
      data-atrec[7-c]=1;
else
      data-atrec[7-c]=0;
for(i=0; i<7; i++)
    pf("%d", data-atrec[i]);
}
```

Output:

```
cc checksum.c
./checksum
```

Enter the IP header info in 16-bit

field 1: 1123
field 2: 1145
field 3: 85ba
field 4: 35a6
field 5: 334ef
field 6: eue4
field 7: 4a36
field 8: 8763
field 9: 3214

computed checksum at sender 26 after IP header info in 16-bit

field 1: 4123
field 2: 1145
field 3: 86a6
field 4: 356a
field 5: 334e1
field 6: 8ue4
field 7: 4a3V
field 8:
field 9:

Competed checksum at receiver 3ac7.

Output

```
cc hamming.c -o hamming
./hamming
sender: 1 0 1 0
Encoded Data: 1010010
Receiver: 1 1 1 0 1 0 1
syndrome bits: 0 1 1
error position: 6

Data sent: 1 0 1 0 0 1 0
Data received: 1 1 1 0 1 0 1
```

Name of the Experiment

Page No.

P4) Implement a simple multicast routing mechanism

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <time.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#define hello_port 12345
#define hello_group "225.0.0.37"
int 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))
{ perror("socket");
    exit(1);
}
memset(&addr, 0, sizeof(addr));
addr.sin_family = AF_INET;
addr.sin_addr.s_addr = inet_addr(hello_gro
addr.sin_port = htons(hello_port);
```

chandra's

```
while (1)
{ if (sendto (fd, message, sizeof (message), 0,
      (struct sockaddr *) &addr,
  sizeof (addr)) < 0)
{ perror ("sendto");
  exit (1); }
  sleep (1); }
  return (0); }
```

Receiver.c

```
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#include <time.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#define hello-port 12345
#define hello-group "225.0.0.37"
#define Msgbuffsize 25

int main (int argc, char *argv[])
{ struct sockaddr_in addr;
int fd, nbytes, addrlen;
struct ip_mreq mreq;
```

chandra's

```
char msgbuf [msgbufsize];
u_int yes=1;
if((fd=socket(AF_INET, SOCK_DGRAM,0))<0)
{perror("socket");
 exit(1);}
if (setsockopt(fd,SOL_SOCKET,SO_REUSEADDR,
        &yes, sizeof(yes))<0)
{perror("reusing ADDR failed");
 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);
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);
}
```

chandra's

Experiment No.
Name of the Experiment

Date
Page No.

```
while(1)
{ addr_len = sizeof(addr);
if(nbytes = recvfrom(fd, msgbuf,
  msgbufsize, 0, (struct sockaddr *)&addr
  &addr_len)) < 0)
{
  perror("recvfrom");
  exit(0);
}
  puts(msgbuf);
}
}
```

Output

cc sender.c

./a.out

output

cc receiver.c

./a.out

RVCE-CSE

RVCE-CSE

RVCE-CSE

RVCE-CSE

RVCE-CSE

RVCE-CSE

Experiment No. 5
Name of the Experiment
Date
Page No.

Aim: WAP to implement concurrent char server that allows current logged in users to communicate one with other.

client.c

```
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/stat.h>
#include <netinet/in.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <arpa/inet.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("EOF");}
int main(int argc, char *argv[])
{ int create_socket;
struct sockaddr_in address;
if((create_socket=socket(AF_INET, SOCK_STREAM, 0))>0)
```

chandra's

Experiment No.
Name of the Experiment
Date
Page No.

```
pf("The socket was created");
address.sin_family = AF_INET;
address.sin_port = htons(15001);
inet_pton(AF_INET, argv[1], &address.sin_addr);
if(connect(create_socket, (socket sockaddr *)
  &address, sizeof(address)) == 0)
pf("The connection was accepted with the
  server %s", argv[1]);
else
    pf("error in connect() \n");
str_cli(stdin, create_socket);
return close(create_socket);
}
```

Server.c

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<sys/select.h>
#include<netinet/in.h>
#include<errno.h>
#include<arpa/inet.h>
void str_echo(int connfd, FILE *fp)
{ int n, bufsize=1024;
  char *buffer = malloc(bufsize);
```

chandra's

```
again:
while ((n=recv(connfd, buffer, bufsize, 0))>0)
{fputs(buffer, stdout);
if(fgets(buffer, bufsize, fp)!=NULL)
    send(connfd, buffer, n, 0);
}
int main()
{int count, listenfd, connfd, addrlen,
  addrlen2, fd, pid, addrlen3;
struct sockaddr_in addr, cli_addr;
if((listenfd=socket(AF_INET, SOCK_STREAM, 0))
{ pf("The socket was created\n"); }
addr.sin_family=AF_INET;
addr.sin_addr.s_addr=INADDR_ANY;
addr.sin_port=htons(15001);
pf("The address before bind:\n...\n");
  ine=_rto a(address.sin_addr));
listen(listenfd, 3);
pf("server is listening\n");
getsockname(listenfd, (struct sockaddr*)
  &addr, &addrlen, 3);
pf("The server is local address:\n...
  &port:%d\n", inet_ntoa(addr.sin_addr),
  htons(addr.sin_port));
while(1)
{ addrlen=sizeof(struct sockaddr_in);
connfd=accept(listenfd, (struct sockaddr*)
  &cli_address, &addrlen);
```

chandra's

Experiment
Name of the Experiment
Date
Page No.

```
addr len2 = sizeof (struct sockaddr_in);
int i = getpeername (connfd, (struct sockad
& cli_address, &addr len);
pf ("the client %s is connected on port
    %d \n", inet_ntoa (cli_addr.sin_ad
ntons (cli_addr.sin_port));
if ((pid == fork()) == 0)
{ pf ("inside child \n");
  close (listenfd);
  str_echo (connfd, stdin);
  exit (0);
}
close (connfd); }
return (0);
}
```

output

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

Socket successfully created ----
connected to the server.
enter the string: hii
from server : hello
enter the string : 12345
from server: 2## $ %. &**
enter the string: hi hello how are y
```

Output: cc server.c -o seru

./seru

Socket successfully created.....
socket successfully binded.....
server listening.....
server accept the client
from client: hii
To client: hello
from client: 12345
To Client: 2#$%4*
from client: hii hello how are you

To client:

Experiment No. Date
Name of the Experiment Page No.

inet_pton(AF_INET, argv[1], &address.sin_addr);
if (connect(create_socket, (struct sockaddr *) &addr, sizeof(address))==0)
printf("connection accepted by the server\n");
else
printf("error in connection");
while(1)(stdin, create_socket);
return close(create_socket);
}

server.c

#include<sys/types.h>
#include<sys/socket.h>
#include<arpa/inet.h>
void echo(int connfd)
{ int n, buffsize=1024;
char *buffer = malloc(buffsize);
if(n<0)
goto again;
}
int main()
{ int count, listenfd, connfd;
struct sockaddr_in address, cli_address;
if((listenfd=socket(AF_INET, SOCK_STREAM,0))>0)

chandra's

Experiment No. | Date
Name of the Experiment | Page No.

```
pf("The socket was created");
address.sin_family = AF_INET;
address.sin_port = htons(5001);
if (bind(listenfd, (struct sockaddr*)
    &address, sizeof(address))==0)
pf("Binding socket");
listen(listenfd, 3);
pf("Iterative server is listening");
return 0;
}
```

Output

cc serverc -o serv ./serv

the socket was created

binding socket

Iterative server is listening

the client 127.0.0.1 is connected on port 5711

Client1 : socket created

the connection accepted with server

hii

hii

hello

hello

Client2 : socket was created

The connection accepted with server

hii

hello

hai

how are you?

UDP - interactive program

client.c

```
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
void str_cli(FILE *fp, int sockfd,
(struct sockaddr*) &serv_address, intservle
int bufsize = 1024, cont;
char *buffer = malloc(bufsize);
int addrlen = sizeof(struct sockaddr_in);
while(fgets(buffer, bufsize, fp) != NULL)
{ if((cont = recvfrom(sockfd, buffer,
        bufsize, 0, NULL, NULL)>0))
{ fputs(buffer, stdout);
}
pf("EOF");}
int main(int argc, char *argv[])
{ int sockfd;
  struct sockaddr_in serv_addr;
if((sockfd = socket(AF_INET, SOCK_DGRAM,0)
pf("the sock created");
str_cli(stdin, sockfd, (struct sockaddr*)
    &serv_address, sizeof(serv_address));
  exit(0);
}
```

chandra's

Name of the Experiment

server.c

```
#include<sys/types.h>
#include<sys/socket.h>
#include<sys/stat.h>
#include<arpa/inet.h>
void str_echo(int sockfd, struct sockaddr *
    cli_address, int clilen)
{ int n, bufsize=1024;
 char* buffer=malloc(bufsize);
 for(;;)
 { addrlen=clilen;
n=recvfrom(sockfd, buffer, bufsize, 0,
    cli_address, &addrlen);
 sendto(sockfd, buffer, n, 0, cli_address,
    addrlen);
}}
int main()
{ int sockfd;
 struct sockaddr_in serv_address,
    cli_address;
 if((sockfd=socket(AF_INET, SOCK_DGRAM
        0))>0
    pf("the socket created");
 serv_address.sin_family=AF_INET;
 serv_address.sin_addr.s_addr=INADDR_
pf("the address before binding \n"
 inet_ntoa(serv_address.sin_addr))
```

chandra's

Output

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

The socket was created
address before bind 0.0.0.0----
Binding socket

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

The socket was created
hii
hii
hello
hello
123
123

Experiment No. | Date

Name of the Experiment | Page No.

```
if (bind (sockfd, (struct sockaddr *)
    & serv_address)) == 0)
pf ("binding socket");
}
```

Experiment No. 7 Date
Name of the Experiment Page No.

27) Implementation of remote command execution using socket system calls.

client.c

```
#include <stdio.h>
#include <sys/socket.h>
#include <sys/stat.h>
#include <unistd.h>
#include <fcntl.h>
void str_cli (FILE *fp, int sockfd)
{ int bufsize=1024, cont;
  char *buffer = malloc (bufsize);
while (fgets(buffer, bufsize, fp) != NULL)
{ if ((cont = recv(sockfd, buffer, bufsize, fp)))
  }
  }}
  pf ("EOF");
  }
int main(int argc, char *argv[])
{ int create_socket;
  struct sockaddr_in address;
if ((create_socket = socket(AF_INET,
  sock_stream, 0)) > 0)
  pfc"The socket was created");
address.sin_family = AF_INET;
inet_pton(AF_INET, argv[1], &address.sin_addr)
```

chandra's

Experiment No. | Date
Name of the Experiment | Page No.

```
if(connect(create_socket, sizeof(caddr))==0)
pf("connection accepted by server");
else
    pf("Error in connect");
str_cli(stdin, create_socket);
return close(create_socket);
}
```

server.c

```
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<arpa/inet.h>
void remote_command(int connfd, int port)
{ int n, bufsize=1024;
  char *buffer = malloc(bufsize);
do
{ while(n=recv(connfd, buffer, bufsize, 0)>0)
{ send(connfd, buffer, n, 0);
    pf("port:%d\n", port);
    system(buffer);
}} while(n<0);
}
```

Experiment No. Date

Name of the Experiment Page No.

```
int main()
{ int count, listenfd, connfd, addrelen,
  fd, pid;
  struct sockaddr_in address;
  if ((listenfd= socket(AF_INET, SOCK_STR
        EAM, 0))>0)
   pf("the socket was created");
  address.sin_family =AF_INET;
  address.sin_port = htons(5001);
  if (bind (listenfd, (struct sockaddr*)
   pf("binding socket");
   listen (listenfd, 3);
   pf("server is listening");
   close (connfd);
}
```

output:

```
cc client.c -o cli
./cli 127.0.0.1
the socket was created
the connection was accepted
with the server 127.0.0.1
ls
ls -l

cc server.c -o serv
./serv
the socket was created
Binding socket
server is listening
the Client 127.0.0.1 is connected
Inside child
port: 60175
total 157
-rwxrwxrwx 1 ab 12960 Jan cli
-rwxrwxrwx 1 ab 13000 Jan serv
-rwxrwxrwx 1 ab 14211 Jan output
```

Experiment No. 8 | Date
Name of the Experiment | Page No.

P8) WAP to encrypt & decrypt the data using RSA & exchange the key using Deffie-Hellman key exchange protocol

RSA.c

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
long int p, q, n, t, flag, e[100], d[100],
temp[100], j, m[100], en[100], i;
char msg[100];
int prime(long int);
void ce();
long int cd(long int);
void encrypt();
void decrypt();
void main()
{ printf("enter first prime no");
  scanf("%d", &p);
  flag = prime(p);
  if (flag==0)
  { printf("wrong input");
    getchar();
    exit(1);
  }
```

Experiment No. Date
Name of the Experiment Page No.

```
pf("enter another prime no");
sf("%d",&q);
flag=prime(q);
if(flag==0||p==q)
{ pf("wrong input");
  getchar();
  exit();
}pf("Enter msg");
  fflush(stdin);
  sf("%s",msg);
for(i=0;msg[i]!=NULL;i++)
  m[i]=msg[i];
  n=p*q
  t=(p-1)*(q-1);
ce();
pf("possible values of e & d are");
for(i=0;i<j-1;i++);
  pf("%d\t%d\n",e[i],d[i]);
  encrypt();
  decrypt();
}
  int prime(long int pr)
{ int i;
  j=sqrt(pr);
for(i=2;i<j;i++)
{ if(pr%i==0)
  return 0;}
```

chandra's

```
return i; }
void ce()
{ int k=0;
for(i=2; i<t; i++)
    if (t%i==0)
        continue;
    flag=prime(i);
if (flag==1&&i!=p&&i!=q)
{ e[k]=i;
  flag = cd(e[k])
  if (flag>0)
  { d[k]=flag; k++;
    if (k==99)
        break;
  }}}
long int cd(long int x)
  { long int k=1;
  while(1)
  { k= k+t;
    if (k%x==0)
  return (k/x); }}
void encrypt()
{ long int pt,ct,key=e[0],k,len; i=0;
  len= strlen(msg);
while (i!=len) { pt=m[i]; pt=pt-96;
                 k=1; }}
```

Experiment No. Date
Name of the Experiment Page No.

```
void decrypt ()
{ long int pt, ct, key = d[0], k, temp; i=0;
  while (en[i] != -1) {
      ct = temp[i]; k=1;
  pt = k*96; m[i] = pt; i++; }
  m[i] = -1;
pf (" the decrypt msg is ");
for (i=0; m[i] != -1; i++)
    pf ("%c", m[i]); }

Diffie-Hellman.c
#include <stdio.h>
  long int power (int a, int b, int mod)
{ long long int t;
  if (b == 1)  return a;
  t = power (a, b/2, mod);
  if (b%2 == 0)  return (t*t)%mod;
  else return (((t*t)%mod)*a)%mod; }

long
long long int calculate key (int a, int x, int n)
{ return power (a, x, n); }

int main ()
{ int n, g, x, a, y, b;
  pf (" enter value of n & g ");
  sf ("%d %d", &n, &g);
```

chandra's

Experiment No. Date

Name of the Experiment Page No.

```
pf("enter the value of x for 1st person");
sf("%d", &x);
a = power(g, x, n);
pf("enter the value of y for 2nd person");
sf("%d", &y);
b = power(g, y, n);
pf("key for the 1st person is = %d", power(b, x
pf("key for the 2nd person is = %d", power(a, y, n
return 0;
}
```

output : cc RSA.C -lm

./a.out

enter first prime number

17

enter another prime number

11

enter message : 88

possible values of e & d are

3	107
7	23
13	37
19	59
23	7
29	149
31	31
37	13
41	121
43	67
47	141

The encrypted msg is : 22

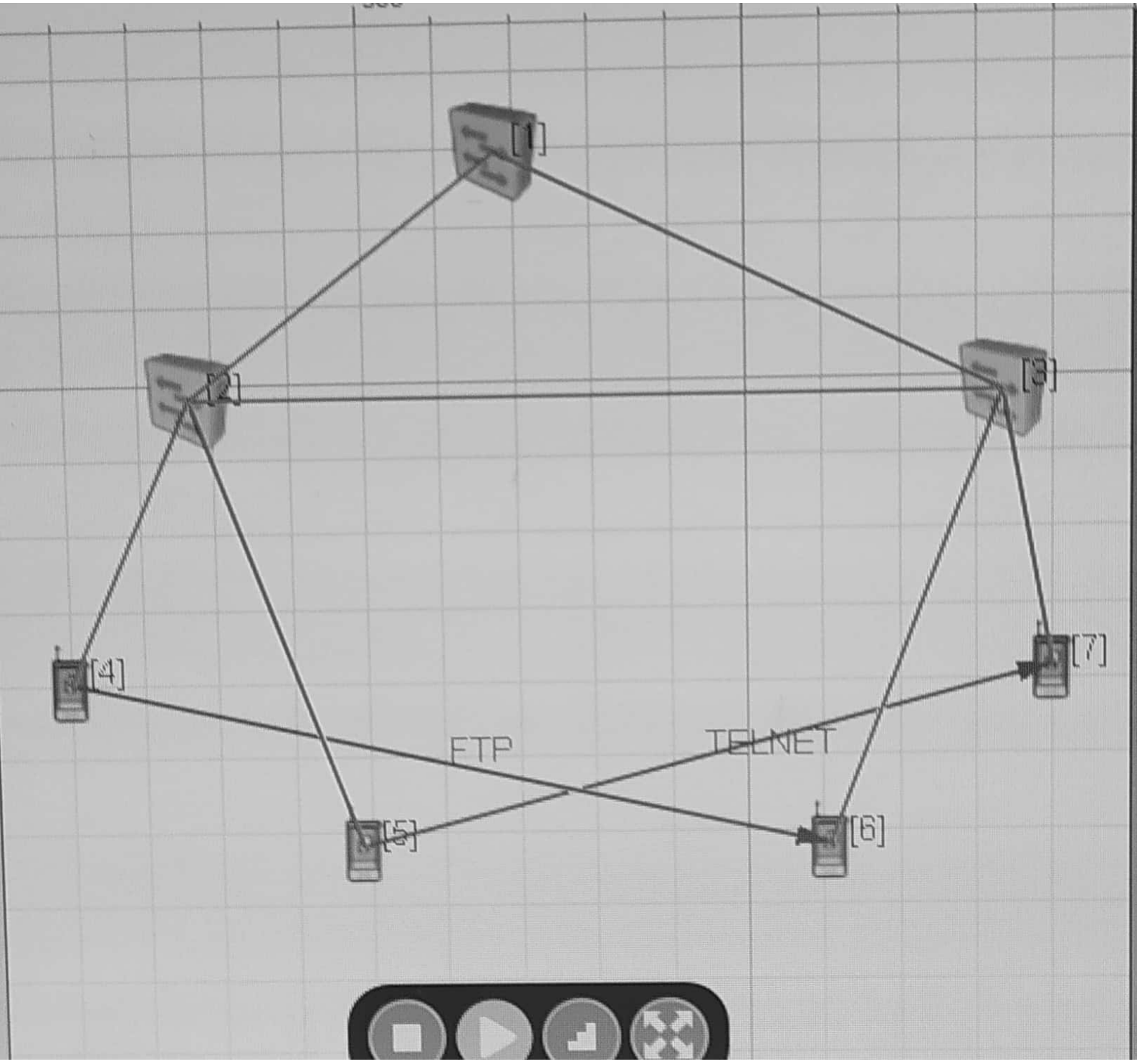
the decrypted msg is : 88

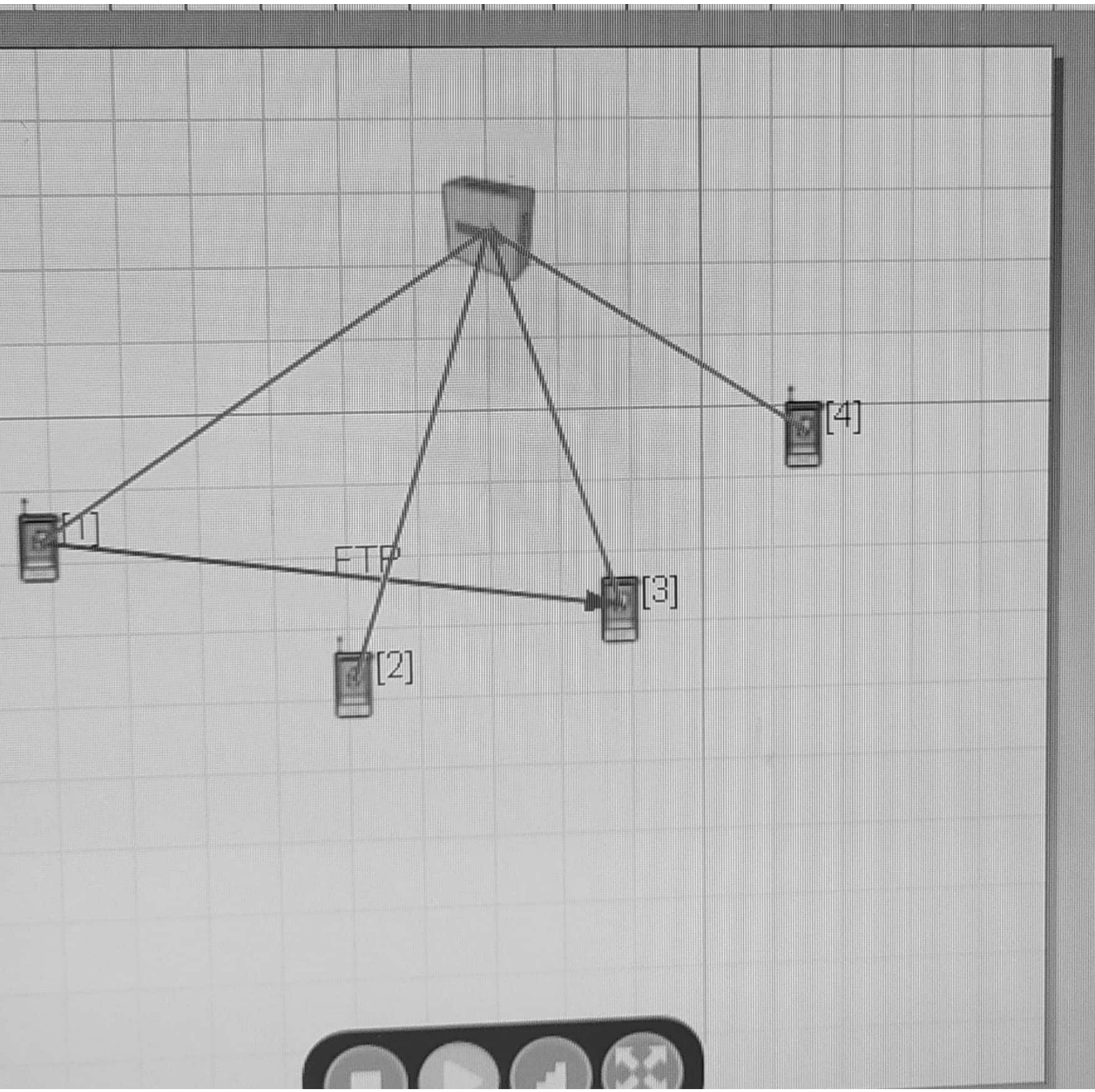
Output

```
cc deffie_hellman.c
./a.out
enter the value of n & g: 23 7
enter the value of x for 1st person: 6
enter the value of y for 2nd person: 5
key for first person: 12
key for second person: 12
```

PART – B

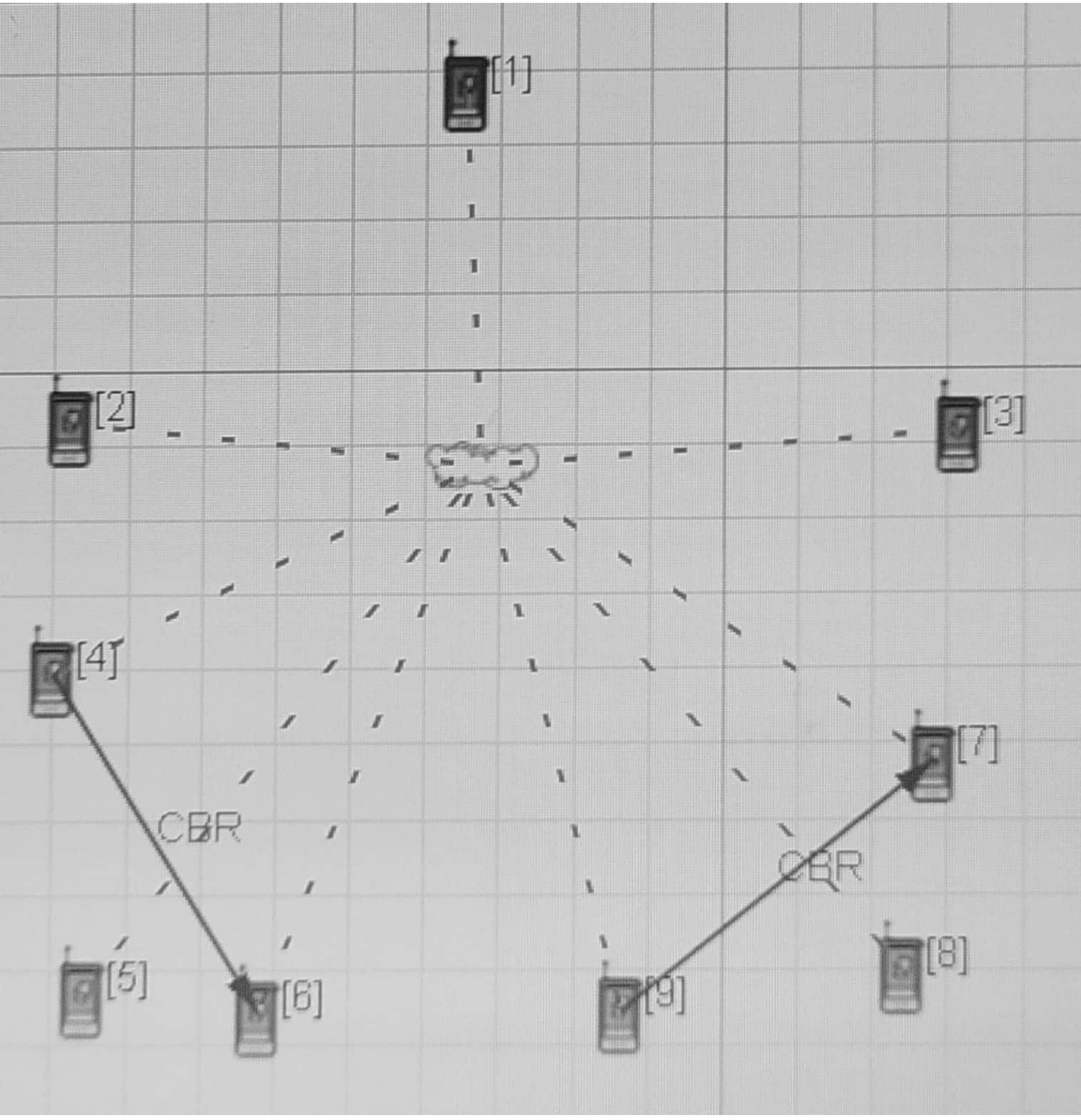
1) Setup an IEEE 802.3 with
a) Hub b) switch c) hierarchy of switch.
Apply the FTP, telnet applications
b/n nodes. Vary the number of nodes.
Vary the bandwidth, queue size and
observe the packet drop probability.





2) Setup a wireless sensor network with atleast two devices co-ordinator & nodes provide constant bit rate(CBR) variant bit rate (VBR) applicatn b/n several nodes. Increase the no. of co-ordinators & nodes in the same area & observe the performance at physical & MAC layers.

chandra's



3) setup an IEEE 802.11 network with at least two AP. Apply the CBR, VBR applications b/n devices belonging to same access points & different access points, provide roaming at any device. Vary the number of AP & devices. Find out the delay in MAC layer, packet drop probability.