

```
unsigned int s1;
s=socket(AF_INET,SOCK_STREAM,0);
ser.sin_family=AF_INET;
ser.sin_port=6500;
ser.sin_addr.s_addr=inet_addr("127.0.0.1");
bind(s,(struct sockaddr *) &ser, sizeof(ser));
listen(s,1);
n=sizeof(cli);
sock=accept(s,(struct sockaddr *)&cli, &n);
printf("\nTCP Connection Established.\n");
s1=(unsigned int) time(NULL);
srand(s1);
strcpy(b,"Time Out ");
recv(sock,a,sizeof(a),0);
f=atoi(a);
while(1)
{
for(i=0;i<W;i++)
{
recv(sock,a,sizeof(a),0);
if(strcmp(a,b)==0)
{
break;
}
}
i=0;
while(i<W)
{
j=rand()%P1;
if(j<P2)
{
send(sock,b,sizeof(b),0);
break;
}
else
{
alpha9(c);
if(c<=f)
{
printf("\nFrame %s Received ",a);
```

```
send(sock,a,sizeof(a),0);
}
else
{
break;
}
c++;
}
if(c>f)
{
break;
}
i++;
}
}
close(sock);
close(s);
return 0;
}
void alpha9(int z)
{
int k,i=0,j,g;
k=z;
while(k>0)
{
i++;
k=k/10;
}
g=i;
i--;
while(z>0)
{
k=z%10;
a[i]=k+48;
i--;
z=z/10;
}
a[g]='\0';
}
```

Client.c

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<string.h>
#include<time.h>
#include<stdlib.h>
#include<ctype.h>
#include<arpa/inet.h>
#define W 5
#define P1 50
#define P2 10
char a[10];
char b[10];
void alpha9(int);
int main()
{
    struct sockaddr_in ser,cli;
    int s,n,sock,i,j,c=1,f;
    unsigned int s1;
    s=socket(AF_INET,SOCK_STREAM,0);
    ser.sin_family=AF_INET;
    ser.sin_port=6500;
    ser.sin_addr.s_addr=inet_addr("127.0.0.1");
    bind(s,(struct sockaddr *)&ser, sizeof(ser));
    listen(s,1);
    n=sizeof(cli);
    sock=accept(s,(struct sockaddr *)&cli, &n);
    printf("\nTCP Connection Established.\n");
    s1=(unsigned int) time(NULL);
    srand(s1);
    strcpy(b,"Time Out ");
    recv(sock,a,sizeof(a),0);
    f=atoi(a);
    while(1)
    {
        for(i=0;i<W;i++)
        {
```

```
recv(sock,a,sizeof(a),0);
if(strcmp(a,b)==0)
{
break;
}
}
i=0;
while(i<W)
{
j=rand()%P1;
if(j<P2)
{
send(sock,b,sizeof(b),0);
break;
}
else
{
alpha9(c);
if(c<=f)
{
printf("\nFrame %s Received ",a);
send(sock,a,sizeof(a),0);
}
else
{
break;
}
c++;
}
}
if(c>f)
{
break;
}
i++;
}
}
close(sock);
close(s);
return 0;
}
```

```

void alpha9(int z)
{
int k,i=0,j,g;
k=z;
while(k>0)
{
i++;
k=k/10;
}
g=i;
i--;
while(z>0)
{
k=z%10;
a[i]=k+48;
i--;
z=z/10;
}
a[g]='\0';
}

```

```

Wed 06:55
p1920@administrator-rusa: ~/unnt/CN
File Edit View Search Terminal Help
The packet number 1 is not received
resending packet 1
The recieved packet is 0
All packets sent successfully
p1920@administrator-rusa: ~/unnt/CN$ ./a.out
1.Selective repeat ARQ
2.Goback ARQ
3.exit
Enter your choice:2
Enter the no. of packets to be sent:2
Floating point exception (core dumped)
p1920@administrator-rusa: ~/unnt/CN$ ./a.out
1.Selective repeat ARQ
2.Goback ARQ
3.exit
Enter your choice:2
Enter the no. of packets to be sent:3
Enter data for packets[1]2
Enter data for packets[2]1
Enter data for packets[3]5
The packet number 1 is not received
2 resending from packet 1
Received data of packet 1 is 2
Received data of packet 2 is 1
Received data of packet 3 is 5
all packets sent successfully
p1920@administrator-rusa: ~/unnt/CN$

```

3. Selective repeat ARQ

Reciver.c

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<string.h>
#include<time.h>
#include<stdlib.h>
#include<ctype.h>
#include<arpa/inet.h>
#define W 5
#define P1 50
#define P2 10
char a[10];
char b[10];
void alpha9(int);
void alp(int);
int main()
{
    struct sockaddr_in ser,cli;
    int s,n,sock,i,j,c=1,f;
    unsigned int s1;
    s=socket(AF_INET,SOCK_STREAM,0);
    ser.sin_family=AF_INET;
    ser.sin_port=6500;
    ser.sin_addr.s_addr=inet_addr("127.0.0.1");
    bind(s,(struct sockaddr *)&ser, sizeof(ser));
    listen(s,1);
    n=sizeof(cli);
    sock=accept(s,(struct sockaddr *)&cli, &n);
    printf("\nTCP Connection Established.\n");
    s1=(unsigned int) time(NULL);
    srand(s1);
    strcpy(b,"Time Out ");
    recv(sock,a,sizeof(a),0);
    f=atoi(a);
    while(1)
    {
        for(i=0;i<W;i++)
        {
```

```
recv(sock,a,sizeof(a),0);
if(strcmp(a,b)==0)
{
break;
}
}
i=0;
while(i<W)
{
L:
j=rand()%P1;
if(j<P2)
{
alp(c);
send(sock,b,sizeof(b),0);
goto L;
}
else
{
alpha9(c);
if(c<=f)
{
printf("\nFrame %s Received ",a);
send(sock,a,sizeof(a),0);
}
else
{
break;
}
c++;
}
if(c>f)
{
break;
}
i++;
}
}
close(sock);
close(s);
```

```
return 0;
}
void alpha9(int z)
{
int k,i=0,j,g;
k=z;
while(k>0)
{
i++;
k=k/10;
}
g=i;
i--;
while(z>0)
{
k=z%10;
a[i]=k+48;
i--;
z=z/10;
}
a[g]='\0';
}
void alp(int z)
{
int k,i=1,j,g;
k=z;
b[0]='N';
while(k>0)
{
i++;
k=k/10;
}
g=i;
i--;
while(z>0)
{
k=z%10;
b[i]=k+48;
i--;
z=z/10;
}
```



```
}  
b[g]='\0';  
}
```

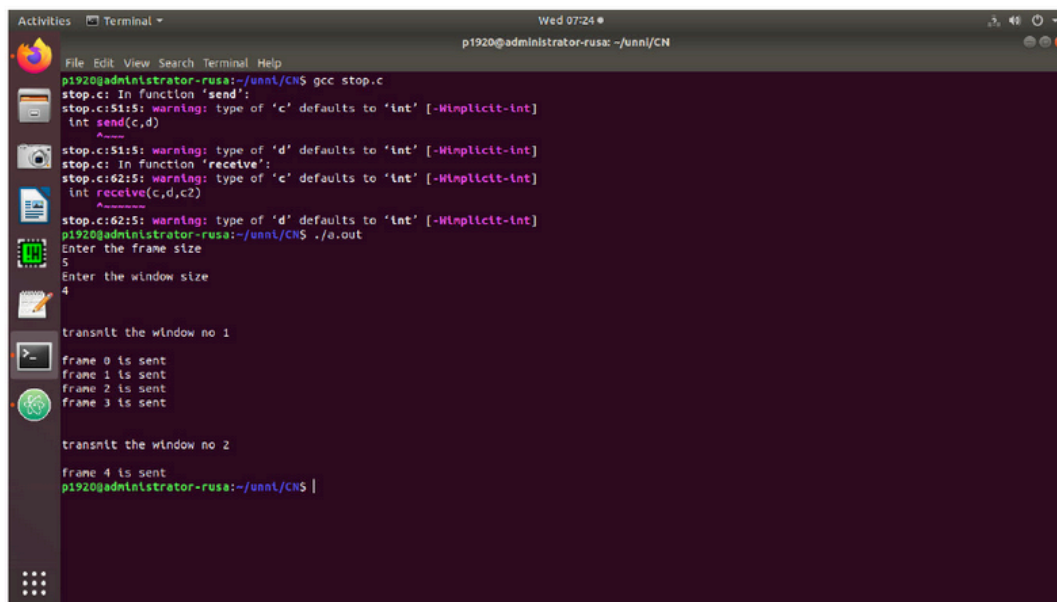
Client.c

```
#include<stdio.h>  
#include<sys/types.h>  
#include<sys/socket.h>  
#include<netinet/in.h>  
#include<string.h>  
#include<time.h>  
#include<stdlib.h>  
#include<ctype.h>  
#define W 5  
char a[10];  
char b[10];  
void alpha9(int);  
int con();  
int main()  
{  
    int s,f,wl,c=1,x,i=0,j,n,p=0,e=0;  
    struct sockaddr_in ser;  
    s=socket(AF_INET,SOCK_STREAM,0);  
    ser.sin_family=AF_INET;  
    ser.sin_port=6500;  
    ser.sin_addr.s_addr=inet_addr("127.0.0.1");  
    connect(s,(struct sockaddr *) &ser, sizeof(ser));  
    printf("\nTCP Connection Established.\n");  
    printf("\nEnter the number of Frames: ");  
    scanf("%d",&f);  
    alpha9(f);  
    send(s,a,sizeof(a),0);  
    strcpy(b,"Time Out ");  
    while(1)  
    {  
        for(i=0;i<W;i++)  
        {  
            alpha9(c);  
            send(s,a,sizeof(a),0);
```

```
if(c<=f)
{
printf("\nFrame %d Sent",c);
c++;
}
}
i=0;
wl=W;
while(i<W)
{
recv(s,a,sizeof(a),0);
p=atoi(a);
if(a[0]=='N')
{
e=con();
if(e<f)
{
printf("\nNAK %d",e);
printf("\nFrame %d sent",e);
i--;
}
}
else
{
if(p<=f)
{
printf("\nFrame %s Acknowledged",a);
wl--;
}
else
{
break;
}
}
if(p>f)
{
break;
}
i++;
}
```

```
if(wl==0 && c>f)
{
send(s,b,sizeof(b),0);
break;
}
else
{
c=c-wl;
wl=W;
}
}
close(s);
return 0;
}
void alpha9(int z)
{
int k,i=0,j,g;
k=z;
while(k>0)
{
i++;
k=k/10;
}
g=i;
i--;
while(z>0)
{
k=z%10;
a[i]=k+48;
i--;
z=z/10;
}
a[g]='\0';
}
int con()
{
char k[9];
int i=1;
while(a[i]!='\0')
{
```

```
k[i-1]=a[i];  
i++;  
}  
k[i-1]='\0';  
i=atoi(k);  
return i;  
}
```



```
Activities Terminal Wed 07:24 p1920@administrator-rusa: ~/unnt/CN  
File Edit View Search Terminal Help  
p1920@administrator-rusa:~/unnt/CN$ gcc stop.c  
stop.c: In function 'send':  
stop.c:51:5: warning: type of 'c' defaults to 'int' [-Wimplicit-int]  
    int send(c,d)  
    ^~~~~~  
stop.c:51:5: warning: type of 'd' defaults to 'int' [-Wimplicit-int]  
stop.c: In function 'receive':  
stop.c:62:5: warning: type of 'c' defaults to 'int' [-Wimplicit-int]  
    int receive(c,d,c2)  
    ^~~~~~  
stop.c:62:5: warning: type of 'd' defaults to 'int' [-Wimplicit-int]  
p1920@administrator-rusa:~/unnt/CN$ ./a.out  
Enter the frame size  
5  
Enter the window size  
4  
  
transmit the window no 1  
  
frame 0 is sent  
frame 1 is sent  
frame 2 is sent  
frame 3 is sent  
  
transmit the window no 2  
  
frame 4 is sent  
p1920@administrator-rusa:~/unnt/CN$ |
```

Experiment 6

Implement and simulate algorithm for Distance Vector Routing protocol

Aim: To implement and simulate algorithm for Distance vector routing protocol

Description:

This algorithm is iterative, and distributed. Each node receives information from its directly attached neighbors, performs some calculations and results to its neighboring nodes. This process of updating the information goes on until there is no exchange of information between neighbors.

Algorithm:

(adapted from Computer Networking – A top down approach by Kurose and Rose)

Bellman Ford algorithm is applied.

Let $dx(y)$ be the cost of the least cost path from node x to node y . Then Bellman Ford equation states that

$$dx(y) = \min\{ c(x,v) + dv(y) \}$$

v

where v is a neighbour of node x . $dv(y)$ is the cost of the least cost path from v to y . $c(x,v)$ is the cost from x to neighbour v . The least cost path has a value equal to minimum of $c(x,v) + dv(y)$ over all its neighbours v . The solution of Bellman Ford equation provides entries in node x 's forwarding table.

Distance vector (DV) algorithm

At each node x

Initialization:

for all destinations y in N :

$$D_x(y) = c(x,y) \text{ /* if } y \text{ is not a neighbour of } x, \text{ then } c(x,y) = \infty \text{ */}$$

```
for each neighbour w,  
send distance vector  $D_x = \{ D_x(y): y \text{ in } N \}$  to w  
loop:  
for each y in N:  
 $D_x(y) = \min \{ c(x,v) + D_v(y) \}$ 
```