SIMULATION OF SLIDING WINDOW FLOW CONTROL PROTOCOLS – STOP & WAIT

ARQ

```
#include <stdio.h>
#include <stdlib.h>
struct frame
{ int info;
  int seq;
} ;
int ak;
int t=5,k;
int disconnect=0;
struct frame p;
char turn='s';
int errorframe=1,errorack=1;
void sender();
void receiver();
void main()
{           p.info=0;
             p.seq=0;
           while(!disconnect)
             {   sender();
                        for(k=1;k<=10000000;k++);
                      receiver();
           }
}
void sender()
{   static int flag=0;
           if(turn=='s')
           {  if(errorack==0)
                      {   printf("SENDER: sent packet with seq NO:%d\n",p.seq);
                       errorframe=rand()%4;
                      printf("%s\n",(errorframe==0?"Error While sending Packet":""));
                      turn='r';

                      }
                      else
                       {
                      if (flag==1) printf("SENDER: Received ACK for packet %d\n",ak);
              if (p.seq==20){  disconnect=1; return;}
                      p.info=p.info+1;
                       p.seq=p.seq+1;
                      printf("SENDER: sent packet with seq NO:%d\n",p.seq);
                       errorframe=rand()%4;
                      printf("%s\n",(errorframe==0?"Error While sending Packet":""));
                       turn='r';
                      flag=1;

                      }
           }
           else
           {          t--;
                       printf("SENDER time reducing\n");
                      if(t==0)
                       {turn='s'  ;
                       errorack=0;
```

```
                    t=5;
                    }
}
}
void receiver()
{   static int frexp=1;
   if(turn=='r')
  {
   if (errorframe!=0)
         { if(p.seq==frexp)
                  {         printf("RECEIVER: Received packet with seq %d\n",p.seq);
                            ak=p.seq;
                            frexp=frexp+1;
                            turn='s';
                            errorack=rand()%4;
                            printf("%s\n",(errorack==0?"Error While sending ACK":""));
                  }
            else
                  {
                            printf("RECEIVER: Duplicated packet with seq %d\n",frexp-1);
                            ak=frexp-1;
                            turn='s';
                            errorack=rand()%4;
                            printf("%s\n",(errorack==0?"Error While sending ACK":""));
                  }
      }
}
}
```

GO BACK N

SERVER.C

```
#include<stdio.h>
#include<string.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<netdb.h>
//structure definition for designing the packet.
struct frame
{
int packet[40];
};
//structure definition for accepting the acknowledgement.
struct ack
{
int acknowledge[40];
};
int main()
{
int serversocket;
struct sockaddr_in serveraddr,clientaddr;
socklen_t len;
struct frame f1;
int windowsize,totalpackets,totalframes,i=0,j=0,framesend=0,k,l,buffer;
struct ack acknowledgement;
char req[50];
serversocket=socket(AF_INET,SOCK_DGRAM,0);
bzero((char*)&serveraddr,sizeof(serveraddr));
```

```
serversocket=socket(AF_INET,SOCK_DGRAM,0);
bzero((char*)&serveraddr,sizeof(serveraddr));
serveraddr.sin_family=AF_INET;
serveraddr.sin_port=htons(5018);
serveraddr.sin_addr.s_addr=INADDR_ANY;
bind(serversocket,(struct sockaddr*)&serveraddr,sizeof(serveraddr));
bzero((char*)&clientaddr,sizeof(clientaddr));
len=sizeof(clientaddr);
//connection establishment.
printf("\nwaiting for client connection");
recvfrom(serversocket,req,sizeof(req),0,
(struct sockaddr*)&clientaddr,&len);
printf("\nThe client connection obtained\t%s\n",req);
//sending request for windowsize.
printf("\nSending request for window size\n");
sendto(serversocket,"REQUEST FOR WINDOWSIZE",sizeof("REQUEST FOR WINDOWSIZE"),0,
(struct sockaddr*)&clientaddr,sizeof(clientaddr));
//obtaining windowsize.
printf("Waiting for the window size\n");
recvfrom(serversocket,(char*)&windowsize,sizeof(windowsize),0,
(struct sockaddr*)&clientaddr,&len);
printf("\nThe window size obtained as:\t %d \n",windowsize);
printf("\nObtaining packets from network layer \n");
printf("\nTotal packets obtained :%d\n",(totalpackets=windowsize*5));
printf("\nTotal frames or windows to be transmitted :%d\n",(totalframes=5));
//sending details to client.
printf("\nSending total number of packets \n");
sendto(serversocket,(char*)&totalpackets,sizeof(totalpackets),0,
(struct sockaddr*)&clientaddr,sizeof(clientaddr));
recvfrom(serversocket,req,sizeof(req),0,
(struct sockaddr*)&clientaddr,&len);
printf("\nSending total number of frames \n");
sendto(serversocket,(char*)&totalframes,sizeof(totalframes),0,
(struct sockaddr*)&clientaddr,sizeof(clientaddr));
recvfrom(serversocket,req,sizeof(req),0,
(struct sockaddr*)&clientaddr,&len);
printf("\n Press enter to start the process \n");
fgets(req,2,stdin);
//starting the process of sending
while(i<totalpackets)
{
//initialising the transmit buffer.
bzero((char*)&f1,sizeof(f1));
printf("\nInitializing the transmit buffer \n");
printf("\n The frame to be send is %d with packets:",framesend);
buffer=i;
j=0;
//Builting the frame.
while(j<windowsize && i<totalpackets)
{
printf("%d",i);
f1.packet[j]=i;
j++;
i++;
}
printf("sending frame %d\n",framesend);
//sending the frame.
```

```
sendto(serversocket,(char*)&f1,sizeof(f1),0,
(struct sockaddr*)&clientaddr,sizeof(clientaddr));
//Waiting for the acknowledgement.
printf("Waiting for the acknowlegment\n");
recvfrom(serversocket,(char*)&acknowledgement,sizeof(acknowledgement),0,
(struct sockaddr*)&clientaddr,&len);
//Checking acknowledgement of each packet.
j=0;
k=0;
l=buffer;
while(j<windowsize && l<totalpackets)
{
if(acknowledgement.acknowledge[j]==-1)
{
printf("\nnegative acknowledgement received for packet:%d \n",f1.packet[j]);
printf("\nRetransmitting from packet:%d \n",f1.packet[j]);
i=f1.packet[j];
i=f1.packet[j];
k=l;
break;
}
j++;
l++;
}
if(k==0)
{
printf("\n Positive acknowledgement received for all packets,within the frame:%d \n",framesend);
}
framesend++;
printf("\n press enter to proceed \n");
fgets(req,2,stdin);
}
printf("\nAll frames sends successfully\n Closing connection with the client \n");
close(serversocket);
}
```

CLIENT.C

```
#include<stdio.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<netdb.h>
#include<string.h>
//structure definition for accepting the packets.
struct frame
{
int packet[40];
};
//structure definition for constructing the acknowledgement frame
struct ack
{
int acknowledge[40];
};
int main()
{
int clientsocket;
struct sockaddr_in serveraddr;
socklen_t len;
```

```
struct hostent *server;
struct frame f1;
int windowsize,totalpackets,totalframes,i=0,j=0,framesreceived=0,k,l,buffer;
struct ack acknowledgement;
char req[50];
clientsocket=socket(AF_INET,SOCK_DGRAM,0);
bzero((char*)&serveraddr,sizeof(serveraddr));
serveraddr.sin_family=AF_INET;
serveraddr.sin_port=htons(5018);
server=gethostbyname("127.0.0.1");
bcopy((char*)server->h_addr,(char*)&serveraddr.sin_addr.s_addr,
sizeof(server->h_addr));
//establishing the connection.
printf("sending request to the server\n");
sendto(clientsocket,"HI IAM CLIENT",sizeof("HI IAM CLIENT"),0,
(struct sockaddr*)&serveraddr,sizeof(serveraddr));
printf("\nWaiting for reply\n");
recvfrom(clientsocket,req,sizeof(req),0,
(struct sockaddr*)&serveraddr,&len);
printf("\n The server has to send :\t%s\n",req);
//accepting window size from the user.
printf("\nenter the window size\n");
scanf("%d",&windowsize);
//sending the window size.
printf("\n sending window size\n");
sendto(clientsocket,(char*)&windowsize,sizeof(windowsize),0,
(struct sockaddr*)&serveraddr,sizeof(serveraddr));
//collecting details from server.
printf("\n waiting for the server response\n");
recvfrom(clientsocket,(char*)&totalpackets,sizeof(totalpackets),0,
(struct sockaddr*)&serveraddr,&len);
printf("\nTotal packets are :\t%d\n",totalpackets);
sendto(clientsocket,"RECEIVED",sizeof("RECEIVED"),0,
(struct sockaddr*)&serveraddr,sizeof(serveraddr));
recvfrom(clientsocket,(char*)&totalframes,sizeof(totalframes),0,
(struct sockaddr*)&serveraddr,&len);
printf("\n total number of frames or windows are:\t%d\n",totalframes);
sendto(clientsocket,"RECEIVED",sizeof("RECEIVED"),0,
(struct sockaddr*)&serveraddr,sizeof(serveraddr));
//starting the process.
printf("\nstarting the process of receiving\n");
while(i<totalpackets)
{
//initialising the receive buffer.
printf("\nInitializing the received buffer\n");
printf("\nThe expected frame is %d with packets:",framesreceived);
j=0;
buffer=i;
while(j<windowsize && i<totalpackets)
{
printf("%d",i);
i++;
j++;
}
printf("\nwaiting for the frame\n");
//accepting the frame.
recvfrom(clientsocket,(char*)&f1,sizeof(f1),0,
```

```
(struct sockaddr*)&serveraddr,&len);
printf("\n received frame %d\n\n enter -1 to send negative acknowledgement for the following packets
\n",framesreceived);
//constructing the acknowledgement frame.
j=0;
l=buffer;
k=0;
while(j<windowsize && l<totalpackets)
{
printf("\npacket:%d\n",f1.packet[j]);
//accepting acknowledgement from the user.
scanf("%d",&acknowledgement.acknowledge[j]);
if(acknowledgement.acknowledge[j]==-1)
{
if(k==0)
{
i=f1.packet[j];
k=1;
}
}
j++;
l++;
}
framesreceived++;
//sending acknowledgement to the server.
sendto(clientsocket,(char*)&acknowledgement,sizeof(acknowledgement),0,
(struct sockaddr*)&serveraddr,sizeof(serveraddr));
}
printf("\nall frames received successfully\n closing connection with the server\n");
close(clientsocket);
}
```

SELECTIVE REPEAT

SERVER.C

```
#include<stdio.h>
#include<string.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<netdb.h>
//structure definition for designing the packet.
struct frame
{
int packet[40];
};
//structure definition for accepting the acknowledgement.
struct ack
{
int acknowledge[40];
};
int main()
{
int serversocket;
struct sockaddr_in serveraddr,clientaddr;
socklen_t len;
struct frame f1;
int windowsize,totalpackets,totalframes,i=0,j=0,framesend=0,k,l,buffer;
struct ack acknowledgement;
```

```
char req[50];
serversocket=socket(AF_INET,SOCK_DGRAM,0);
bzero((char*)&serveraddr,sizeof(serveraddr));
serversocket=socket(AF_INET,SOCK_DGRAM,0);
bzero((char*)&serveraddr,sizeof(serveraddr));
serveraddr.sin_family=AF_INET;
serveraddr.sin_port=htons(5018);
serveraddr.sin_addr.s_addr=INADDR_ANY;
bind(serversocket,(struct sockaddr*)&serveraddr,sizeof(serveraddr));
bzero((char*)&clientaddr,sizeof(clientaddr));
len=sizeof(clientaddr);
//connection establishment.
printf("\nwaiting for client connection");
recvfrom(serversocket,req,sizeof(req),0,
(struct sockaddr*)&clientaddr,&len);
printf("\nThe client connection obtained\t%s\n",req);
//sending request for windowsize.
printf("\nSending request for window size\n");
sendto(serversocket,"REQUEST FOR WINDOWSIZE",sizeof("REQUEST FOR WINDOWSIZE"),0,
(struct sockaddr*)&clientaddr,sizeof(clientaddr));
//obtaining windowsize.
printf("Waiting for the window size\n");
recvfrom(serversocket,(char*)&windowsize,sizeof(windowsize),0,
(struct sockaddr*)&clientaddr,&len);
printf("\nThe window size obtained as:\t %d \n",windowsize);
printf("\nObtaining packets from network layer \n");
printf("\nTotal packets obtained :%d\n",(totalpackets=windowsize*5));
printf("\nTotal frames or windows to be transmitted :%d\n",(totalframes=5));
//sending details to client.
printf("\nSending total number of packets \n");
sendto(serversocket,(char*)&totalpackets,sizeof(totalpackets),0,
(struct sockaddr*)&clientaddr,sizeof(clientaddr));
recvfrom(serversocket,req,sizeof(req),0,
(struct sockaddr*)&clientaddr,&len);
printf("\nSending total number of frames \n");
sendto(serversocket,(char*)&totalframes,sizeof(totalframes),0,
(struct sockaddr*)&clientaddr,sizeof(clientaddr));
recvfrom(serversocket,req,sizeof(req),0,
(struct sockaddr*)&clientaddr,&len);
printf("\n Press enter to start the process \n");
fgets(req,2,stdin);
//starting the process of sending
while(i<totalpackets)
{
//initialising the transmit buffer.
bzero((char*)&f1,sizeof(f1));
printf("\nInitializing the transmit buffer \n");
printf("\n The frame to be send is %d with packets:",framesend);
buffer=i;
j=0;
//Builting the frame.
while(j<windowsize && i<totalpackets)
{
printf("%d",i);
f1.packet[j]=i;
j++;
i++;
```

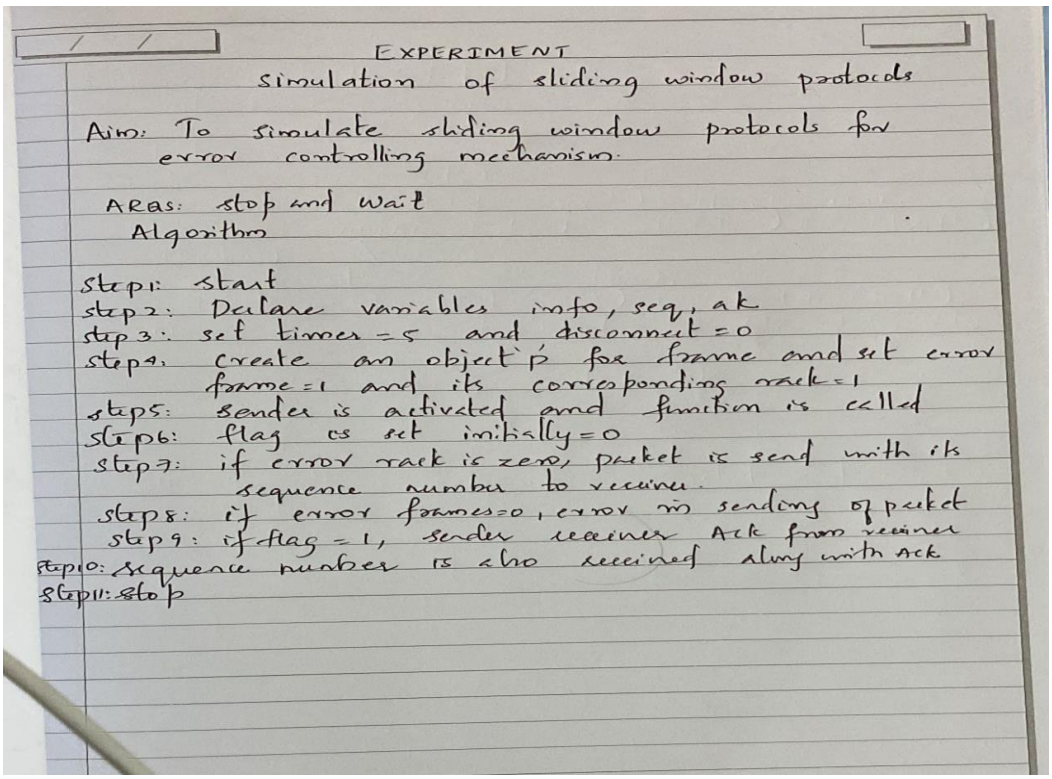
```
}
printf("sending frame %d\n",framesend);
//sending the frame.
sendto(serversocket,(char*)&f1,sizeof(f1),0,
(struct sockaddr*)&clientaddr,sizeof(clientaddr));
//Waiting for the acknowledgement.
printf("Waiting for the acknowlegment\n");
recvfrom(serversocket,(char*)&acknowledgement,sizeof(acknowledgement),0,
(struct sockaddr*)&clientaddr,&len);
//Checking acknowledgement of each packet.
j=0;
k=0;
l=buffer;
while(j<windowsize && l<totalpackets)
{
if(acknowledgement.acknowledge[j]==-1)
{
printf("\nnegative acknowledgement received for packet:%d \n",f1.packet[j]);
printf("\nRetransmitting from packet:%d \n",f1.packet[j]);
i=f1.packet[j];
i=f1.packet[j];
k=l;
break;
}
j++;
l++;
}
if(k==0)
{
printf("\n Positive acknowledgement received for all packets,within the frame:%d \n",framesend);
}
framesend++;
printf("\n press enter to proceed \n");
fgets(req,2,stdin);
}
printf("\nAll frames sends successfully\n Closing connection with the client \n");
close(serversocket);
}
```

CLIENT.C

```
#include<stdio.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<netdb.h>
#include<string.h>
//structure definition for accepting the packets.
struct frame
{
int packet[40];
};
//structure definition for constructing the acknowledgement frame
struct ack
{
int acknowledge[40];
};
int main()
{
```

```
int clientsocket;
struct sockaddr_in serveraddr;
socklen_t len;
struct hostent *server;
struct frame f1;
int windowsize,totalpackets,totalframes,i=0,j=0,framesreceived=0,k,l,buffer;
struct ack acknowledgement;
char req[50];
clientsocket=socket(AF_INET,SOCK_DGRAM,0);
bzero((char*)&serveraddr,sizeof(serveraddr));
serveraddr.sin_family=AF_INET;
serveraddr.sin_port=htons(5018);
server=gethostbyname("127.0.0.1");
bcopy((char*)server->h_addr,(char*)&serveraddr.sin_addr.s_addr,
sizeof(server->h_addr));
//establishing the connection.
printf("sending request to the server\n");
sendto(clientsocket,"HI IAM CLIENT",sizeof("HI IAM CLIENT"),0,
(struct sockaddr*)&serveraddr,sizeof(serveraddr));
printf("\nWaiting for reply\n");
recvfrom(clientsocket,req,sizeof(req),0,
(struct sockaddr*)&serveraddr,&len);
printf("\n The server has to send :\t%s\n",req);
//accepting window size from the user.
printf("\nenter the window size\n");
scanf("%d",&windowsize);
//sending the window size.
printf("\n sending window size\n");
sendto(clientsocket,(char*)&windowsize,sizeof(windowsize),0,
(struct sockaddr*)&serveraddr,sizeof(serveraddr));
//collecting details from server.
printf("\n waiting for the server response\n");
recvfrom(clientsocket,(char*)&totalpackets,sizeof(totalpackets),0,
(struct sockaddr*)&serveraddr,&len);
printf("\nTotal packets are :\t%d\n",totalpackets);
sendto(clientsocket,"RECEIVED",sizeof("RECEIVED"),0,
(struct sockaddr*)&serveraddr,sizeof(serveraddr));
recvfrom(clientsocket,(char*)&totalframes,sizeof(totalframes),0,
(struct sockaddr*)&serveraddr,&len);
printf("\n total number of frames or windows are:\t%d\n",totalframes);
sendto(clientsocket,"RECEIVED",sizeof("RECEIVED"),0,
(struct sockaddr*)&serveraddr,sizeof(serveraddr));
//starting the process.
printf("\nstarting the process of receiving\n");
while(i<totalpackets)
{
//initialising the receive buffer.
printf("\nInitializing the received buffer\n");
printf("\nThe expected frame is %d with packets:",framesreceived);
j=0;
buffer=i;
while(j<windowsize && i<totalpackets)
{
printf("%d",i);
i++;
j++;
}
```

```
printf("\nwaiting for the frame\n");
//accepting the frame.
recvfrom(clientsocket,(char*)&f1,sizeof(f1),0,
(struct sockaddr*)&serveraddr,&len);
printf("\n received frame %d\n\n enter -1 to send negative acknowledgement for the following packets
\n",framesreceived);
//constructing the acknowledgement frame.
j=0;
l=buffer;
k=0;
while(j<windowsize && l<totalpackets)
{
printf("\npacket:%d\n",f1.packet[j]);
//accepting acknowledgement from the user.
scanf("%d",&acknowledgement.acknowledge[j]);
if(acknowledgement.acknowledge[j]==-1)
{
if(k==0)
{
i=f1.packet[j];
k=1;
}
}
j++;
l++;
}
framesreceived++;
//sending acknowledgement to the server.
sendto(clientsocket,(char*)&acknowledgement,sizeof(acknowledgement),0,
(struct sockaddr*)&serveraddr,sizeof(serveraddr));
}
printf("\nall frames received successfully\n closing connection with the server\n");
close(clientsocket);
}
```



EXPERIMENT

simulation of sliding window protocols

Aim: To simulate sliding window protocols for error controlling mechanism.

ARQs: stop and wait

Algorithm

step1: start
step2: Declare variables info, seq, ak
step3: set timer = 5 and disconnect = 0
step4: create an object p for frame and set error frame = 1 and its corresponding rack = 1
step5: sender is activated and function is called
step6: flag is set initially = 0
step7: if error rack is zero, packet is send with its sequence number to receiver.
step8: if error frames=0, error in sending of packet
step9: if flag = 1, sender receiver Ack from receiver
step10: sequence number is also received along with Ack
step11: stop