Aayush Shah
19BCE245
2 October 2021

Practical 4

DLL protocols

- **Implement selective repeat sliding window protocol.**

- **Code :**

sender.cpp

```
#include<iostream>

#include<stdio.h>

 #include<sys/types.h>

#include<netinet/in.h>

#include<netdb.h>


#define cls() printf("33[H33[J")

                                                    //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;

 sockaddr_in serveraddr;

 socklen_t len;

 hostent * server;

 frame f1;

 int
windowsize,totalpackets,totalframes,i=0,j=0,framesreceived=0,k
,l,m,repacket[40];

 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("\nSending request to the client.\n");

 sendto(clientsocket,"HI I AM CLIENT.",sizeof("HI I AM
CLIENT."),0,(sockaddr*)&serveraddr,sizeof(serveraddr));

  printf("\nWaiting for reply.\n");

 recvfrom(clientsocket,req,sizeof(req),0,
(sockaddr*)&serveraddr,&len);

 printf("\nThe server has send:\t%s\n",req);

                                                   //accepting
window size from the user.

 printf("\nEnter the window size:\t");

 scanf("%d",&windowsize);

                                                  //sending the
window size.

 printf("\n\nSending the window size.\n");

 sendto(clientsocket,(char*)&windowsize,sizeof(windowsize),0,
(sockaddr*)&serveraddr,sizeof(serveraddr));

 cls();

                                                   //collecting
details from server.

 printf("\nWaiting for the server response.\n");
```

```
 recvfrom(clientsocket,
(char*)&totalpackets,sizeof(totalpackets),0,
(sockaddr*)&serveraddr,&len);

 printf("\nThe total packets are:\t%d\n",totalpackets);

 sendto(clientsocket,"RECEIVED.",sizeof("RECEIVED."),0,
(sockaddr*)&serveraddr,sizeof(serveraddr));

  recvfrom(clientsocket,
(char*)&totalframes,sizeof(totalframes),0,
(sockaddr*)&serveraddr,&len);

 printf("\nThe total frames/windows are:\t%d\n",totalframes);

 sendto(clientsocket,"RECEIVED.",sizeof("RECEIVED."),0,
(sockaddr*)&serveraddr,sizeof(serveraddr));

                                                  //starting
the process.

 printf("\nStarting the process of receiving.\n");

 j=0;

 l=0;

 while(l<totalpackets)

 {                                                  //
initialising the receive buffer.

   printf("\nInitialising the receive buffer.\n");

   printf("\nThe expected frame is %d with packets:
",framesreceived);

     for(m=0;m<j;m++)

   {                                        //readjusting for
packets with negative acknowledgement.

    printf("%d  ",repacket[m]);
```

```
   }
   while(j<windowsize && i<totalpackets)
   {
    printf("%d  ",i);
    i++;
    j++;
   }

   printf("\n\nWaiting for the frame.\n");
                                                    //accepting
the frame.
   recvfrom(clientsocket,(char*)&f1,sizeof(f1),0,
(sockaddr*)&serveraddr,&len);
   printf("\nReceived frame %d\n\nEnter -1 to send negative
acknowledgement for the following packets.\n",framesreceived);
                                                      //
constructing the acknowledgement frame.
   j=0;
   m=0;
   k=l;
   while(m<windowsize && k<totalpackets)
   {
    printf("\nPacket: %d\n",f1.packet[m]);
```

```
                                                    //accepting
acknowledgement from the user.
    scanf("%d",&acknowledgement.acknowledge[m]);
     if(acknowledgement.acknowledge[m]==-1)
    {
     repacket[j]=f1.packet[m];
     j++;
    }
    else
    {
     l++;
    }
    m++;
    k++;
   }
   framesreceived++;
                                                    //sending
acknowledgement to the server.
 sendto(clientsocket,
(char*)&acknowledgement,sizeof(acknowledgement),0,
(sockaddr*)&serveraddr,sizeof(serveraddr));
  cls();
 }
```

```
  printf("\nAll frames received successfully.\n\nClosing
connection with the server.\n");

// close(clientsocket);

}
```

receiver.cpp

```
#include<iostream>

#include<stdio.h>


#include<sys/types.h>

#include<netinet/in.h>

#include<netdb.h>


#define cls() printf("33[H33[J")


                                                                    //
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;
 sockaddr_in serveraddr,clientaddr;
 socklen_t len;
 int
windowsize,totalpackets,totalframes,framessend=0,i=0,j=0,k,l,m
,n,repacket[40];
 ack acknowledgement;
 frame f1;
 char req[50];


 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,(sockaddr*)&serveraddr,sizeof(serveraddr));


 bzero((char*)&clientaddr,sizeof(clientaddr));

 len=sizeof(clientaddr);


                                                                  //
connection establishment.

 printf("\nWaiting for client connection.\n");

 recvfrom(serversocket,req,sizeof(req),0,
(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,
(sockaddr*)&clientaddr,sizeof(clientaddr));


                                                                  //
obtaining windowsize.
```

```
 printf("\nWaiting for the windowsize.\n");

 recvfrom(serversocket,
(char*)&windowsize,sizeof(windowsize),0,
(sockaddr*)&clientaddr,&len);

 cls();

 printf("\nThe windowsize obtained as:\t%d\n",windowsize);



 printf("\nObtaining packets from network layer.\n");

 printf("\nTotal packets obtained:\t%d\n",
(totalpackets=windowsize*5));

 printf("\nTotal frames or windows to be transmitted:\t%d\n",
(totalframes=5));



                                                          //
sending details to client.

 printf("\nSending total number of packets.\n");

 sendto(serversocket,
(char*)&totalpackets,sizeof(totalpackets),0,
(sockaddr*)&clientaddr,sizeof(clientaddr));

 recvfrom(serversocket,req,sizeof(req),0,
(sockaddr*)&clientaddr,&len);



 printf("\nSending total number of frames.\n");

 sendto(serversocket,
(char*)&totalframes,sizeof(totalframes),0,
(sockaddr*)&clientaddr,sizeof(clientaddr));
```

```
 recvfrom(serversocket,req,sizeof(req),0,
(sockaddr*)&clientaddr,&len);



 printf("\nPRESS ENTER TO START THE PROCESS.\n");

 fgets(req,2,stdin);

 cls();



 j=0;

 l=0;                                                    //
starting the process of sending

 while( l<totalpackets)

 {

                                                          //
initialising the transmit buffer.

  bzero((char*)&f1,sizeof(f1));

  printf("\nInitialising the transmit buffer.\n");

  printf("\nThe frame to be send is %d with packets:
\t",framessend);

                                                          //
Builting the frame.

  for(m=0;m<j;m++)

  {

            //including the packets for which negative
acknowledgement was received.

   printf("%d  ",repacket[m]);
```

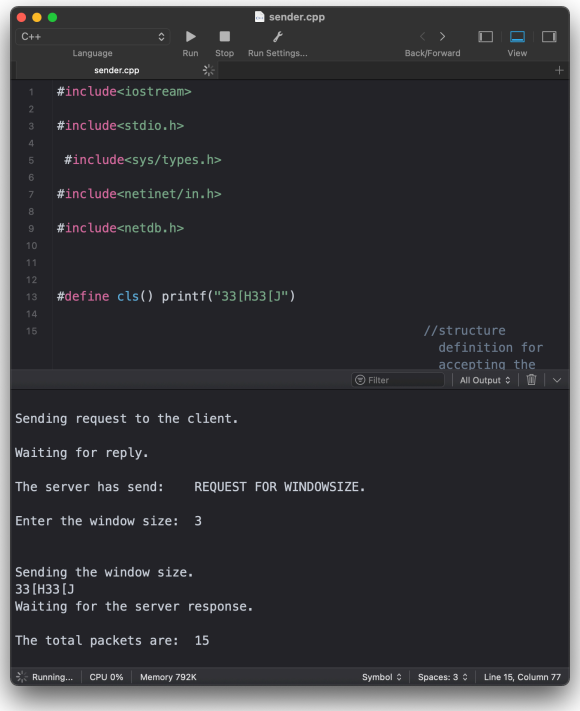
```
   f1.packet[m]=repacket[m];
  }

  while(j<windowsize && i<totalpackets)
  {
   printf("%d  ",i);
   f1.packet[j]=i;
   i++;
   j++;
  }
  printf("\nSending frame %d\n",framessend);

                                                                    //
sending the frame.
  sendto(serversocket,(char*)&f1,sizeof(f1),0,
(sockaddr*)&clientaddr,sizeof(clientaddr));

                                                                    //
Waiting for the acknowledgement.
  printf("\nWaiting for the acknowledgement.\n");
  recvfrom(serversocket,
(char*)&acknowledgement,sizeof(acknowledgement),0,
(sockaddr*)&clientaddr,&len);
  cls();
```

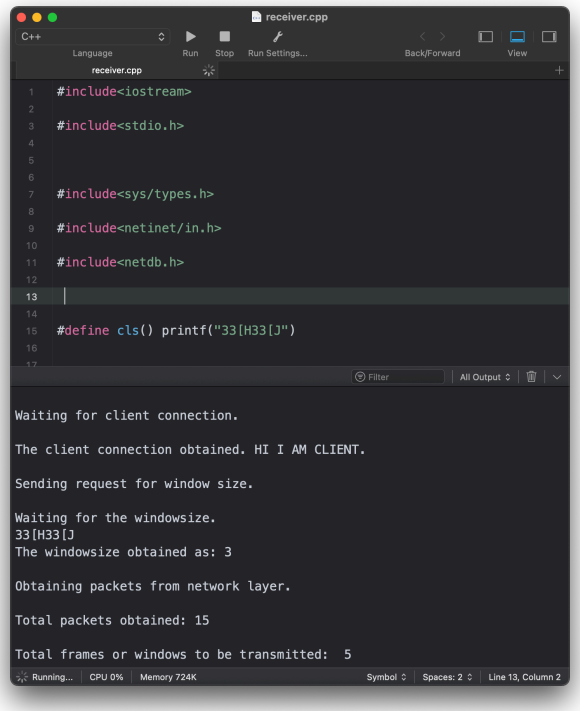
```
                                                                    //
Checking acknowledgement of each packet.
  j=0;
  k=0;
  m=0;
  n=l;
  while(m<windowsize && n<totalpackets)
  {
   if(acknowledgement.acknowledge[m]==-1)
   {
    printf("\nNegative acknowledgement received for packet:
%d\n",f1.packet[m]);
    k=1;
    repacket[j]=f1.packet[m];
    j++;
   }
   else
   {
    l++;
   }
   m++;
   n++;
```

```
  }


  if(k==0)

  {

 printf("\nPositive acknowledgement received for all packets
within the frame: %d\n",framessend);

  }


  framessend++;

  printf("\nPRESS ENTER TO PROCEED……\n");

  fgets(req,2,stdin);

  cls();

 }


 printf("\nAll frames send successfully.\n\nClosing connection
with the client.\n");

// close(serversocket);

}
```

- **Execution :**

1. Executing sender.cpp and receiver.cpp files simultaneously, giving frame length :





2. Giving acknowledgment through sender.cpp and receiving packets in receiver.cpp :

> Whenever negative acknowledgment received, packet will be sent again.