Date :
Page:

NP Lab

Rishav Karanjit
~~1SI17CS08~~

Sliding window

7. Write a program to implement sliding window protocol between two hosts.
[logic at last] →
Code :-

↳ Server side:

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

#define RECVBUF 20

int AdvWindow = 0;

void main(){
    int std, lfd, port, i, j, status, choice;
    char str[20], str1[20], err[20], advWindow[1024],
    ack_str[20];
    int ack;
    char frame1[20], frame[20];
    int sendsize = 5;
    char *recv_str;
    recv_str = malloc(50);
    memset(recv_str, 0, 20);
    int yes=1;
```

Date :
Page:

```
struct sockaddr_in saddr,caddr;

port =5000;
sfd = socket(AF_INET, SOCK_STREAM, 0);

if (sfd<0)
   perror("Error");

if(setsockopt(sfd, SOL_SOCKET, SO_REUSEADDR,
   &yes, sizeof(int)) ==-1){
      perror ("setsock opt");
}
bzero(&saddr, sizeof(saddr));
saddr.sin_family =AF_INET;
saddr.sin_addr.s_addr= htonl(INADDR_ANY);
saddr.sin_port= htons(port);

lfd = bind(sfd, (struct sockaddr *)&saddr,
      sizeof(saddr));
if(lfd)
   perror("Bind Error");

listen(sfd, 5);

len = sizeof(&caddr);
nfd = accept(sfd, (struct sockaddr *)&caddr,
          &len);
len= -1;
i=0;
```

Date : ....
Page: ....

```
~~fo~~ While (1){
    m~~re~~mset(frame, 0,20);
    recv(lfd, frame, 100, 0);
    if(strcmp(frame~~s~~, "exit")==0)
    {
        printf("\n Exitting !\n");
        break;
    }
    int err=rand() % 8;
    int i5;
    if (err <4){
        memset(frame1, 0,20);
        for(i5=0; i5<err; i5++)
            frame1[i5] = frame[i5];
        recv_str=(char *) strcat(recv_str,frame)
        frame[err]='x';
        printf("\n\nIntroduce error at
         frame= %d Error at %d, Error
        full frame recved=%s --Retransmit", err
        i+err,frame);
      i = i+err;
      ack = i;
    }
    else{
      printf("\n\nRecving frame (Success)= %s,
                 Recving WINDOW: start seqno=%
              -end seq no =%d", frame, i, i+sendsi
    recv_str= (char *) strcat(recv_str, frame);
    i=i+sendsize;
     ack=i;
    }
```

Date : ....
Page: ....

```
    printf("\n Receiver: Sending Ack back to
    sender ack=%d ", ack);
    sprintf(ack_str, "%d", ack);
    send (lfd, ack_str, strlen(ack_str), 0);
  }
  printf("\nReceived Final str at Destination %s\n",
                                        recv_str);
  close(sfd);
}
```

↳ client side

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

void main(){
    int sfd, len, choice, i, j, status, sendsize, port,
        recvsize, temp1;
    char str[20], frame[20], temp[20], ack[20],
        sendwin[20];
    char *msg = "network programming";
    struct sockaddr_in saddr, caddr;
    port = 5000;
```

Date : ....
Page: ....

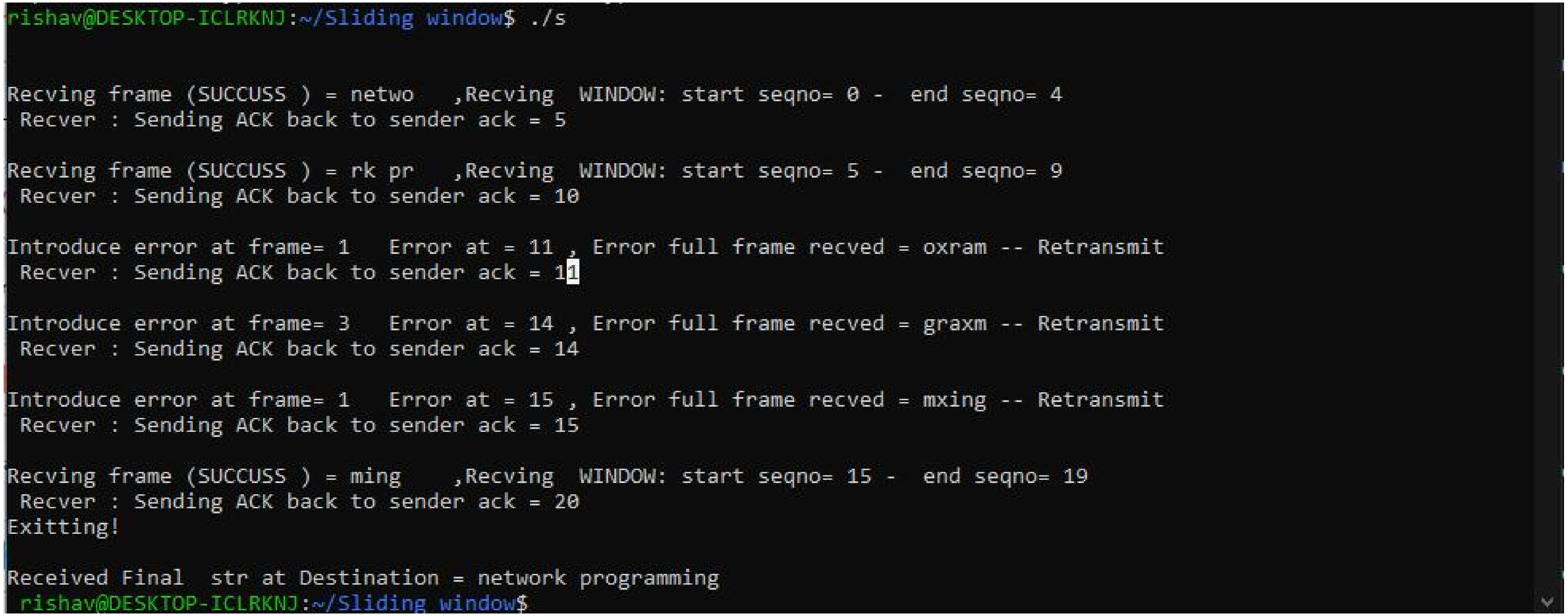
```
std = socket (AF_INET, SOCK_STREAM, 0);
if ( std<0)
    perror("Error");
bzero(&saddr, sizeof(saddr));
saddr . sin_family = AF_INET;
 inet_pton (AF_INET, "127.0.0.1", &saddr.sin_addr);
saddr. sin_port = htons(port);

connect (std, (struct sock addr *) &saddr,
        sizeof(saddr));
printf( "\n msg= %s", msg);
printf ("\n len = %d", strlen(msg));
printf ("\n len= %d", strlen(msg));
printf("enter the text").
i=0;
sendsize =5;
while (i< strlen(msg)) {
   memset(frame, 0,20);
   strncpy (frame, msg+i, sendsize);
   printf ("\n\n Sending frame = %s, Sending
          WINDOW: start seqno %d - end seqno=
          %d", frame, i, i+sendsize -1);
  send (std, frame, strlen(frame), 0);
  printf("\nsending data and wait for ack");
  memset (ack, 0,20);
  recv(std, ack, 100,0);
  sscanf (ack, "%d", &status).
  printf("\n recvd ack no = %d", status);
  i = status;
}
~~write(std, "Exit", sizeof("Exit"));~~
printf(" \nExiting"! \n").
close(std);
}
```

Date : ....
Page: ....

Logic:-

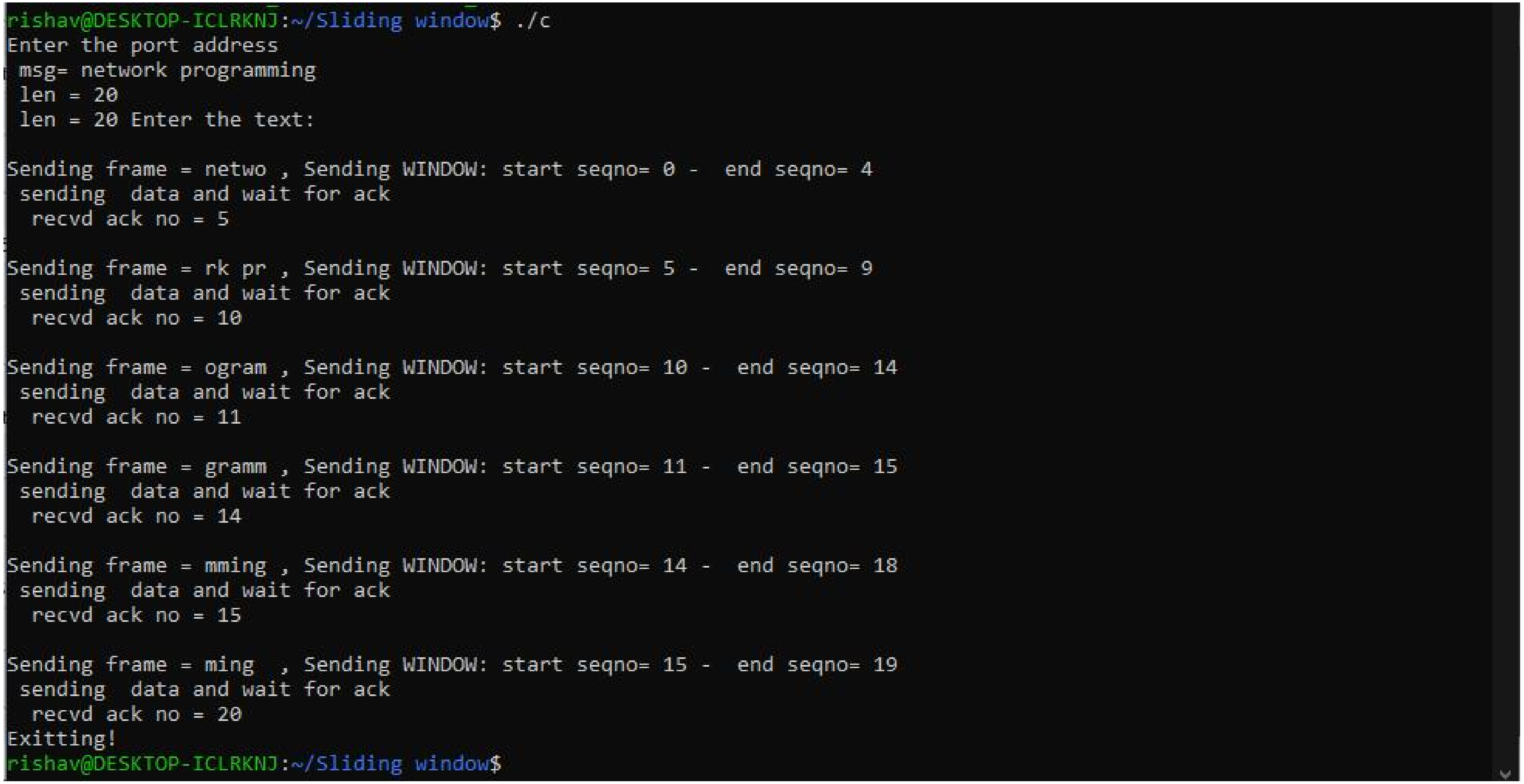
- In this protocols, the sender has a buffer called sending window
- The receiver have buffer called the receiving window
- The size of ~~rece~~ receiving window is the maximum number of frames that the receiver can accept.
- ~~First~~, Receiver requests for certain frame
- Then, sender sends the requested frame to receiver
- Receiver send ack if frame received successfully otherwise it will send the frame no. for which ~~error~~ receiver had not received.
- Sender slides the window to right if ack received
- If error occurs, sender resends the frame for which ack was received

Sender's Side Output



```
rishav@DESKTOP-ICLRKNJ:~/Sliding window$ ./s


Recving frame (SUCCUSS ) = netwo   ,Recving  WINDOW: start seqno= 0 -  end seqno= 4
 Recver : Sending ACK back to sender ack = 5

Recving frame (SUCCUSS ) = rk pr   ,Recving  WINDOW: start seqno= 5 -  end seqno= 9
 Recver : Sending ACK back to sender ack = 10

Introduce error at frame= 1   Error at = 11 , Error full frame recved = oxram -- Retransmit
 Recver : Sending ACK back to sender ack = 11

Introduce error at frame= 3   Error at = 14 , Error full frame recved = graxm -- Retransmit
 Recver : Sending ACK back to sender ack = 14

Introduce error at frame= 1   Error at = 15 , Error full frame recved = mxing -- Retransmit
 Recver : Sending ACK back to sender ack = 15

Recving frame (SUCCUSS ) = ming    ,Recving  WINDOW: start seqno= 15 -  end seqno= 19
 Recver : Sending ACK back to sender ack = 20
Exitting!

Received Final  str at Destination = network programming
 rishav@DESKTOP-ICLRKNJ:~/Sliding window$
```

Client's Side Output



```
rishav@DESKTOP-ICLRKNJ:~/Sliding window$ ./c
Enter the port address
 msg= network programming
 len = 20
 len = 20 Enter the text:

Sending frame = netwo , Sending WINDOW: start seqno= 0 -  end seqno= 4
 sending  data and wait for ack
  recvd ack no = 5

Sending frame = rk pr , Sending WINDOW: start seqno= 5 -  end seqno= 9
 sending  data and wait for ack
  recvd ack no = 10

Sending frame = ogram , Sending WINDOW: start seqno= 10 -  end seqno= 14
 sending  data and wait for ack
  recvd ack no = 11

Sending frame = gramm , Sending WINDOW: start seqno= 11 -  end seqno= 15
 sending  data and wait for ack
  recvd ack no = 14

Sending frame = mming , Sending WINDOW: start seqno= 14 -  end seqno= 18
 sending  data and wait for ack
  recvd ack no = 15

Sending frame = ming  , Sending WINDOW: start seqno= 15 -  end seqno= 19
 sending  data and wait for ack
  recvd ack no = 20
Exitting!
rishav@DESKTOP-ICLRKNJ:~/Sliding window$
```