Experiment- 7
SIMULATION OF SLIDING WINDWO FLOW CONTROL PROTOCOL

AIM: To simulate Sliding Window Flow Control Protocol

TOOLS USED: *GCC compiler, socket API in Debian platform for C-Language*

DESCRIPTION: The most important functions of Data Link and Transport layers are **Flow Control** and **Error Control** collectively known as link control.

Flow Control: Is a technique that allows sender and receiver with different speed characteristics to communicate each other. It is a set of procedures that tells the sender how much data it can transmit before it must wait for an acknowledgement from the receiver. If ensures that a transmitting station such as a server with higher processing capability, does not overwhelm a receiving station, such as a desktop system, with lesser processing capacity. This is where there is an orderly flow of transmitted data between the source and destination.

There are two methods for flow control namely stop-and-wait and Sliding-window. Stop-and-wait is also known as Request/Reply sometimes. Stop-and-wait flow control requires each data packet to be acknowledged by the remote host/process before the next packet is sent. Sliding window algorithms, used by TCP, permit multiple data packets to be in simultaneous transmit, making more efficient use of network bandwidth.

Sliding Window: The efficiency of stop-and-wait is very poor when **a>1(**a: propagation delay**)**. It can be greatly improved by sl making use of full-duplex line. To keep track of the frames, sender station sends sequentially numbered frames. Since the sequence number to be used occupies a field in the frame, it should be of limited size. If the header of the frame allows **k** bits, the sequence numbers range from **0** to 2^k**-1**. Sender maintains a list of sequence numbers that is allowed to send (**sender window**). The size of the sender's window is at most 2^k**-1** that is allowed to send (**sender window**). The sender is provided with a buffer equal to the window size. Receiver also maintains a window of 2^k**-1** size. The receiver acknowledges a frame by sending an ACK frame that includes the sequence number of the next frame expected. This is also explicitly announces that it is prepared to receive the next **N** frames, beginning with the number specified. This scheme can be used to acknowledge multiple frames. It could receive frames 2, 3, 4 but withhold ACK until frame 4 has arrived. The receiver needs a buffer size of **1**.

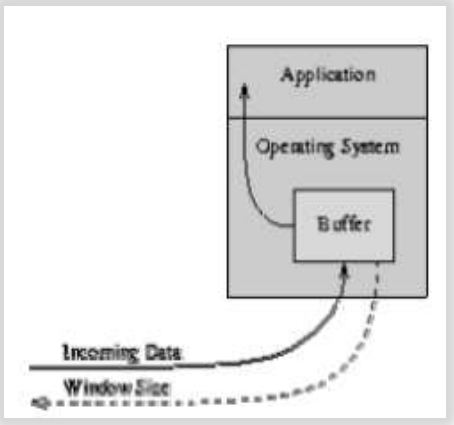

Figure 6.1Buffer in sliding window

Sliding window algorithm is a method of flow control for network data transfer, TCP uses this protocol. A sliding window algorithm places a buffer between the application program and the network data flow as shown in Figure 6.1. For TCP, the buffer is typically in an operating system kernel, but this is more of an implementation detail than a hard-and-fast requirement.

Data received from the network is stored in the buffer, from where the application can read at its own pace. As the application reads data, buffer space is freed up to accept more input from the network. The *window* is the amount of data that can be "read ahead"- the size of the buffer. Less the amount of valid data stored in it. *Window announcements* are used to inform the remote host of the current *window size*.

Sender sliding Window: At any instant, the sender is permitted to send frames with sequence numbers in a certain range (the sending window) as shown in Figure 6.2.

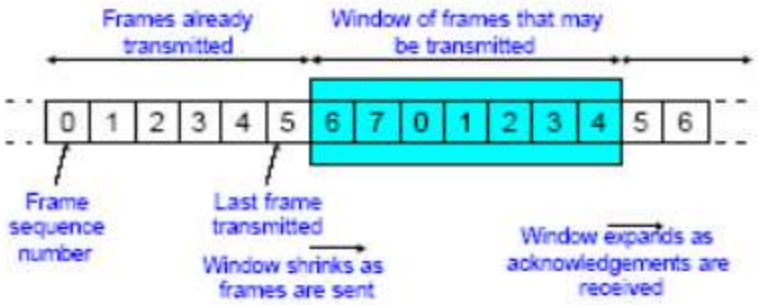

Figure 6.2: Sender's window

Receiver sliding window: The receiver always maintains a window of size 1 as shown in Figure 6.3. it looks for a specific frame (frame 4 as shown in the figure) to arrive in a specific order. If it receives any other frame (out of order), it is discarded and it needs to be resent. However, the receiver window also slides by one as the specific frame is received and accepted as shown in the figure. The receiver acknowledges a frame by sending an ACK frame that includes the sequence number of the next frame expected. This is also explicitly announces that it is prepared to receive the next N frames, beginning with the number specified. This scheme can be used to acknowledge multiple frames. It could receive frames 2, 3, 4 but withhold ACK until frame 4 has arrived. By returning a ACK with sequence number 5 ,it acknowledges frames 2, 3, 4 at one time the receiver needs a buffer of size 1.

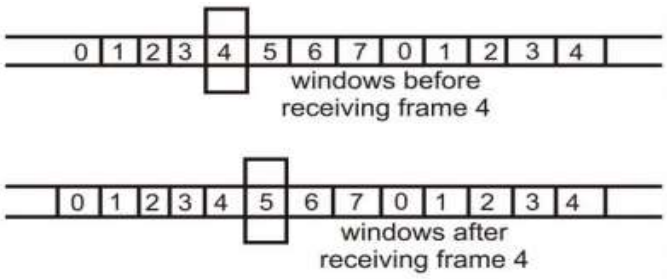

Figure 6.3: Receiver sliding window

On the other hand, if the local application if the local application can process data at the rate it's being transferred; sliding window still gives us an advantage. If the window size is larger than the packet size, then multiple packets can be outstanding in the network, since the sender knows that buffer space is available on the receiver to hold all of them. Ideally, a steady state condition can be reached where a series of packets (in the forward direction) and window announcements (in the reverse direction) are constantly in transit. As each new window announcement is received by the sender, more data packets are transmitted. As the application reads data from the buffer (remember, we're assuming the application can keep up with the network), more window announcements are generated. Keeping a series of data packets in transit ensures the efficient use of network resources.

Transmission time: Time it takes for a station to transmit a frame (normalized to a value of 1).

Propagation delay: The time it takes for a bit to travel from sender to receiver .

Stop-and-wait protocol:

- a < 1 :The frame is sufficiently long such that the first bits of the frame arrive at the destination before the source has completed transmission of the frame.
- a > 1: Sender completes transmission of the entire frame before the leading bits of the frame arrive at the receiver.
- The link utilization **U =** $\frac{1}{(1+2a)}$ where a = *Propagation time / transmission time*

Sliding Window Flow Control:

- Allows transmission of multiple frames
- Assigns each frame a k-bit sequence number
- Range of sequence number is [0…2k -1], i.e., frames are counted modulo 2k.

The link utilization in case of Sliding Window Protocol is:

U = 1, for N > 2a + 1

U = $\frac{N}{(1+2a)}$, for N < 2a + 1

Where N = the window size, and a = Propagation time / transmission time

ALGORITHM: Server program

1. Start
2. Create an unnamed TCP socket and bind its address
3. Accept connection request from a client process
4. Prompt the user to enter the data and the window size (say N)

5. Create N frames and send to the frames to the receiver
6. Wait until the acknowledgement arrives
7. If positive ACK arrives (with the available window size M) then send M more frames and continue
8. If a negative ACK (error ACK with sequence number **X** and window M) arrives, then go back and retransmit M frames from sequence number 'X'
9. Repeat steps 6 through and 8 until all the packets are transmitted.
10. Stop

ALGORITHM: Client program

1. Start
2. Create an unnamed TCP socket
3. Establish a connection specifying the server address
4. Read frames/packets from the server and display
5. Check whether a packet is received correctly or not.
6. If received correctly then send a positive ACK specifying the next packet sequence number and window size. Otherwise, send a negative ACK specifying the error packet sequence number and the window size.
7. Repeat steps 4 through 6 until all the packets received.
8. Stop

Sample code:

//Simulation of Sliding Window Flow Control Protocol

//SlideServer.c

```
#include<stdio.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<string.h>
#include<stdlib.h>
#include<arpa/inet.h>
#define SIZE 4                                  //window size
int main(int argc, char *argv[])
{
   int sfd,s,nsfd,len,i,j,status;
   char str[100],frame[100],temp[100],ack[20];
   struct sockaddr_in sa,ca;
   sfd=socket(AF_INET,SOCK_STREAM,0);
```

```
if(sfd<0)
{
   perror("Error");
   exit(-1);
}
bzero(&sa,sizeof(sa));
sa.sin_family=AF_INET;
sa.sin_addr.s_addr=htonl(INADDR_ANY);
sa.sin_port=htons(atoi(argv[1]));

s=bind(sfd,(struct sockaddr*)&sa,sizeof(sa));  //assign a name to  the server
if(s<0)
{
  perror("Bind Error");
  exit(-1);
}

listen(sfd,5);
len=sizeof(&ca);
nsfd=accept(sfd,(struct sockaddr*)&ca,&len); //take a connection request
printf(" Enter the text : ");
scanf("%s",&str);                          //read a string to be transmitted
i=0;
while(i<strlen(str))
{
    memset(frame,0,100);
    strncpy(frame,str+i,SIZE);             //generate a frame
    write(1," Transmitting Frames: ",23);
    len=strlen(frame);
    for(j=0;j<len;j++)                     //copy the  sequence numbers into the frame
    {
      printf(" %d ",i+j);
      sprintf(temp," %d",i+j);
      strcat(frame,temp);
    }

    printf("\n");
    write(nsfd,&frame,sizeof(frame));   //send frames
    //printf("\nSent message: %s\n",frame);
    read(nsfd,ack,20);
    sscanf(ack,"%d",&status);
    if(status==-1)
      printf(" Transmission is successful. \n");
    else
```

```
        {
          printf(" Received error in frame: %d ",status);
          printf("\n Retransmitting Frames:  ");
          for(j=0;;)
          {
             frame[j]=str[j+status];
             printf("%d ",j+status);
             j++;
             if((j+status)%4==0)
                break;
          }
          printf("\n");
          frame[j]='\0';
          len=strlen(frame);
          for(j=0;j<len;j++)
          {
           sprintf(temp," %d ",j+status);
           strcat(frame,temp);
          }
          write(nsfd,&frame,sizeof(frame));   //Retransmit the error frame
        }
        i+=SIZE;
      }
          write(nsfd,"exit",sizeof("exit"));     //End of transmission
          printf("\nExiting.....\n");
          sleep(2);
          close(nsfd);
          close(sfd);
    return 0;
}
```

//SlideClient.c

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<netinet/in.h>
int main(int argc, char *argv[])
{
      int sfd,lfd,len,choice,s,n;
      char str[100],str1[100],err[100];
      struct sockaddr_in saddr,caddr;
```

```
    sfd=socket(AF_INET,SOCK_STREAM,0);     //create an unnamed TCP socket
    if(sfd<0)
    {
      perror("FdError");
      exit(-1);
    }
    bzero(&saddr,sizeof(saddr));
    saddr.sin_family=AF_INET;                    //initialize the server address buffer
    saddr.sin_addr.s_addr=INADDR_ANY;
    saddr.sin_port=htons(atoi(argv[1]));
    s=connect(sfd,(struct sockaddr*)&saddr,sizeof(saddr));//connect to the sender
    if(s<0)
     {
         perror("connect error");
      exit(-1);
     }

    for(;;)
    {
          n=recv(sfd,&str,100,0);              //read the frames from the sender
          if(!strncmp(str,"exit",4))
          {
                printf("Exiting.....\n");
                break;
          }
          str[n]='\0';
          printf("\nReceived message is: %s\n Are there any errors?(1-Yes 0-No): ",str);
          scanf("%d",&choice);
          if(!choice)
                write(sfd,"-1",sizeof("-1"));
          else
          {
                printf("Enter the sequence no of the frame where error has occured: ");
                scanf("%s",&err);
                write(sfd,&err,sizeof(err));
                n=read(sfd,&str,20);
                str[n]='\0';
                printf("\n\nReceived the re-transmitted frames: %s\n\n",str);
          }
    }
}
```

RESULTS AND DISCUSSIONS:

- Test the code with different combinations of input and write your observations for each input

CONCLUSIONS:

- Understood and simulated the sliding window flow control protocol using socket API and C compiler
- Successfully test with all possible test cases

References:

1. https://www.youtube.com/watch?v=elq4gEeFmg4
2. https://www.youtube.com/watch?v=zY3Sxvj8kZA
3. https://www.youtube.com/watch?v=ZLtkhsgQp8U&ebc=ANyPxKoUxYhRILUQErzLNBynIVSq2KyRIMubjFQDJHTrVBNbmdx5sEoCJtxMs32lwdSOOz_dpwNFkb9yc8Br7wBQRNfvlJa2Ww
4. http://nptel.ac.in/courses/106105080/pdf/M3L3.pdf