**EXERCISE: 3 SLIDING WINDOW PROTOCOL**

**23-08-2019 & GO\_BACK N PROTOCOL**

**SLIDING WINDOW PROTOCOL**

**AIM:** To implement sliding window protocol using C.

**ALGORITHM:**

1. Create a socket for server and start listening for client requests.

2. Create a socket for every client and create a thread for sending messages to the server.

3. Server will wait for client to send a message and on receiving one, lock the mutex so that no other client can send.

4. Server creates a thread for sending the message from this client to all other clients.

5. After sending the message to other clients mutex is unlocked.

6. Each client receives the message from the server and displays.

**CODE:**

**Server:**

#include <sys/types.h>

#include <netinet/in.h>

#include <netdb.h>

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <unistd.h>

#include <errno.h>

int main()

{

int sock,size,connect;

char senddata[50],data[50];

int val,count,i,port;

struct sockaddr\_in ser,cli;

printf("\n\n Server Running ...... ");

if ((sock = socket(AF\_INET, SOCK\_STREAM, 0)) == -1)

{

perror("\n Socket Creation Error");

exit(-1);

}

printf("\nEnter the port number : ");

scanf("%d",&port);

ser.sin\_family = AF\_INET;

ser.sin\_port = htons(port);

ser.sin\_addr.s\_addr=INADDR\_ANY;

bzero(&(ser.sin\_zero),8);

if(bind(sock,(struct sockaddr\*)&ser,sizeof(struct sockaddr)==-1)

{

perror("\n\t Error in Bind");

exit(-1);

}

if (listen(sock,2)==-1)

{

perror("\n\t Error in Listen");

exit(-1);

}

printf("\n\t Waiting for connection ");

size=sizeof(struct sockaddr);

connect=accept(sock,(struct sockaddr \*)&cli,&size);

if(connect==-1)

{

perror("\n\t Connection Failed :");

exit(-1);

}

printf("\n\t Connected Successfully");

printf("\n");

recv(connect,&val,sizeof(val),0);

count=val;

while(1)

{

i=recv(connect,&data,sizeof(data),0);

data[i]='\0';

if (strcmp(data,"end")==0)

{

printf("\n\t Finished");

break;

}

if(count!=val)

{

strcpy(senddata,"packet missing");

send(connect,&count,sizeof(count),0);

send(connect,senddata,strlen(senddata),0);

}

else

{

printf("\n The packet Number is : %d",val);

printf("\n The data is :%s",data);

count++;

strcpy(senddata,"send nextdata");

send(connect,&count,sizeof(count),0);

send(connect,senddata,strlen(senddata),0);

}

printf("\n The Expected Packet now is: %d \n",count);

recv(connect,&val,sizeof(val),0);

}

close(connect);

close(sock);

return 0;

}

**Client:**

#include <sys/types.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <netdb.h>

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <unistd.h>

#include <errno.h>

int main()

{

int sock,val,i,count,port;

char recvdata[50],sentdata[50];

struct sockaddr\_in server\_addr;

printf("\n\n Client Running ...... ");

if ((sock = socket(AF\_INET, SOCK\_STREAM, 0)) == -1)

{

perror("Socket");

exit(1);

}

printf("\nEnter the port number");

scanf("%d",&port);

server\_addr.sin\_family = AF\_INET;

server\_addr.sin\_port = htons(port);

server\_addr.sin\_addr.s\_addr= htonl(INADDR\_ANY);

bzero(&(server\_addr.sin\_zero),8);

if(connect(sock,(struct sockaddr\*)&server\_addr,sizeof(struct sockaddr))==-1)

{

perror("Connect");

exit(1);

}

while(1)

{

printf("\n Enter packet number ");

scanf("%d",&val);

send(sock,&val,sizeof(val),0);

printf("\n Enter data ");

scanf("%s",sentdata);

send(sock,sentdata,strlen(sentdata),0);

if(strcmp(sentdata,"end")==0)

break;

recv(sock,&count,sizeof(count),0);

i=recv(sock,recvdata,50,0);

recvdata[i]='\0';

printf("\n %s %d",recvdata,count);

}

close(sock);

return 0;

}

**OUTPUT:**

**Server:**

[106117107@localhost network]$ gcc SlidingServer.c

[106117107@localhost network]$ ./a.out

Server Running ......

Enter the port number : 8080

Waiting for connection

Connected Successfully

The packet Number is : 12

The data is :hello

The Expected Packet now is: 13

The packet Number is : 13

The data is :hi

The Expected Packet now is: 14

The Expected Packet now is: 14

Finished

**Client :**

[106117107@localhost network]$ gcc SlidingClient.c

[106117107@localhost network]$ ./a.out

Client Running ......

Enter the port number8080

Enter packet number 12

Enter data hello

send nextdata 13

Enter packet number 13

Enter data hi

send nextdata 14

Enter packet number 20

Enter data kitty

packet missing 14

Enter packet number 14

Enter data end

**INFERENCE:** Sliding window protocol is implemented using C.

**GO\_BACK N PROTOCOL**

**AIM:** To implement Go – Back N protocol using C.

**ALGORITHM:**

1. Create a socket for server and start listening for client requests.

2. Create a socket for every client and create a thread for sending messages to the server.

3. Server will wait for client to send a message and on receiving one, lock the mutex so that no other client can send.

4. Server creates a thread for sending the message from this client to all other clients.

5. After sending the message to other clients mutex is unlocked.

6. Each client receives the message from the server and displays.

**CODE:**

**Receiver:**

#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';

}

**Sender:**

#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 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(strcmp(a,b)==0)

{

e=c-wl;

if(e<f)

{

printf("\nTime Out, Resent Frame %d onwards",e);

}

break;

}

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';

}

**OUTPUT:**

**Receiver:**

[106117107@localhost network]$ gcc GoBackNReceiver.c

[106117107@localhost network]$ ./a.out

TCP Connection Established.

Frame 1 Received

Frame 2 Received

Frame 3 Received

Frame 4 Received

Frame 5 Received

Frame 6 Received

Frame 7 Received

Frame 8 Received

Frame 9 Received

Frame 10 Received

Frame 11 Received

Frame 12 Received

Frame 13 Received

Frame 14 Received

Frame 15 Received

Frame 16 Received

Frame 17 Received

Frame 18 Received

Frame 19 Received

Frame 20 Received

Frame 21 Received

Frame 22 Received

Frame 23 Received

Frame 24 Received

Frame 25 Received

Frame 26 Received

Frame 27 Received

Frame 28 Received

Frame 29 Received

Frame 30 Received

Frame 31 Received

Frame 32 Received

Frame 33 Received

Frame 34 Received

Frame 35 Received

Frame 36 Received

Frame 37 Received

Frame 38 Received

Frame 39 Received

Frame 40 Received

Frame 41 Received

Frame 42 Received

Frame 43 Received

Frame 44 Received

Frame 45 Received

Frame 46 Received

Frame 47 Received

Frame 48 Received

Frame 49 Received

**Client :**

[106117107@localhost network]$ gcc GoBackNSender.c

[106117107@localhost network]$ ./a.out

TCP Connection Established.

Enter the number of Frames: 50

Frame 1 Sent

Frame 2 Sent

Frame 3 Sent

Frame 4 Sent

Frame 5 Sent

Frame 1 Acknowledged

Frame 2 Acknowledged

Frame 3 Acknowledged

Frame 4 Acknowledged

Frame 5 Acknowledged

Frame 6 Sent

Frame 7 Sent

Frame 8 Sent

Frame 9 Sent

Frame 10 Sent

Frame 6 Acknowledged

Frame 7 Acknowledged

Frame 8 Acknowledged

Frame 9 Acknowledged

Frame 10 Acknowledged

Frame 11 Sent

Frame 12 Sent

Frame 13 Sent

Frame 14 Sent

Frame 15 Sent

Time Out, Resent Frame 11 onwards

Frame 11 Sent

Frame 12 Sent

Frame 13 Sent

Frame 14 Sent

Frame 15 Sent

Time Out, Resent Frame 11 onwards

Frame 11 Sent

Frame 12 Sent

Frame 13 Sent

Frame 14 Sent

Frame 15 Sent

Frame 11 Acknowledged

Frame 12 Acknowledged

Frame 13 Acknowledged

Time Out, Resent Frame 14 onwards

Frame 14 Sent

Frame 15 Sent

Frame 16 Sent

Frame 17 Sent

Frame 18 Sent

Frame 14 Acknowledged

Frame 15 Acknowledged

Frame 16 Acknowledged

Time Out, Resent Frame 17 onwards

Frame 17 Sent

Frame 18 Sent

Frame 19 Sent

Frame 20 Sent

Frame 21 Sent

Frame 17 Acknowledged

Frame 18 Acknowledged

Frame 19 Acknowledged

Frame 20 Acknowledged

Frame 21 Acknowledged

Frame 22 Sent

Frame 23 Sent

Frame 24 Sent

Frame 25 Sent

Frame 26 Sent

Frame 22 Acknowledged

Frame 23 Acknowledged

Frame 24 Acknowledged

Time Out, Resent Frame 25 onwards

Frame 25 Sent

Frame 26 Sent

Frame 27 Sent

Frame 28 Sent

Frame 29 Sent

Frame 25 Acknowledged

Frame 26 Acknowledged

Frame 27 Acknowledged

Frame 28 Acknowledged

Frame 29 Acknowledged

Frame 30 Sent

Frame 31 Sent

Frame 32 Sent

Frame 33 Sent

Frame 34 Sent

Frame 30 Acknowledged

Time Out, Resent Frame 31 onwards

Frame 31 Sent

Frame 32 Sent

Frame 33 Sent

Frame 34 Sent

Frame 35 Sent

Time Out, Resent Frame 31 onwards

Frame 31 Sent

Frame 32 Sent

Frame 33 Sent

Frame 34 Sent

Frame 35 Sent

Frame 31 Acknowledged

Frame 32 Acknowledged

Frame 33 Acknowledged

Frame 34 Acknowledged

Frame 35 Acknowledged

Frame 36 Sent

Frame 37 Sent

Frame 38 Sent

Frame 39 Sent

Frame 40 Sent

Frame 36 Acknowledged

Frame 37 Acknowledged

Frame 38 Acknowledged

Frame 39 Acknowledged

Frame 40 Acknowledged

Frame 41 Sent

Frame 42 Sent

Frame 43 Sent

Frame 44 Sent

Frame 45 Sent

Frame 41 Acknowledged

Frame 42 Acknowledged

Frame 43 Acknowledged

Frame 44 Acknowledged

Frame 45 Acknowledged

Frame 46 Sent

Frame 47 Sent

Frame 48 Sent

Frame 49 Sent

Frame 50 Sent

Frame 46 Acknowledged

Frame 47 Acknowledged

Frame 48 Acknowledged

Frame 49 Acknowledged

Frame 50 Acknowledged

**INFERENCE:** Sliding window protocol is implemented using C.