HARISH K M
COE18B066

Computer Networks Practice
Exercise 8

Stop and Wait Protocol:

Stop and Wait Protocol is the simplest flow control protocol.

It works under the following assumptions-

- Communication channel is perfect.
- No error occurs during transmission.

Working:

The working of a stop and wait protocol may be explained as-

- Sender sends a data packet to the receiver.
- Sender stops and waits for the acknowledgment for the sent packet from the receiver.
- Receiver receives and processes the data packet.
- Receiver sends an acknowledgment to the sender.
- After receiving the acknowledgment, sender sends the next data packet to the receiver.

How code works:

Sender side:

Client acts as sender.So client first connects with the server.Sender code gets the message to be sent from the user.At every iteration (an iteration per character of the string),the sender sends a variable called stringend to notify to the receiver whether or not the string has ended.A character is sent and acknowledgement is received.If ack is 0 the character is sent again.This happens until string end is reached.

Receiver side:

Sercer acts as receiver.So it accepts clients connection request.At every iteration it checks if the string has ended using the stringend variable sent by sender.If string has not ended the character is received and an acknowledgement is sent (1 for success and 0 for failure).The received characters are stored in a character array and printed at last.

Sender code:

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

int main()
{
int c_sock;
c_sock=socket(AF_INET,SOCK_STREAM,0);
struct sockaddr_in client;
memset(&client,0,sizeof(client));
client.sin_family=AF_INET;
client.sin_port=htons(9009);
client.sin_addr.s_addr=INADDR_ANY;

if(connect(c_sock,(struct sockaddr*)&client,sizeof(client))==-1)
{
printf("server is busy\nconnection failure");
return 0;
}

char buf[2];
buf[1]='\0';
char msg[50];
printf("\nEnter message to be sent:\n");
gets(msg);
char stringend[2];
stringend[1]='\0';
for(int i=0;i<strlen(msg);++i)
{
buf[0]=msg[i];
```

```
char ack[2];
stringend[0]='0';
send(c_sock,stringend,sizeof(stringend),0);
send(c_sock,buf,sizeof(buf),0);
recv(c_sock,ack,sizeof(ack),0);
printf("%c ",ack[0]);
if(ack[0]=='0')i-=1;
}
stringend[0]='1';
send(c_sock,stringend,sizeof(stringend),0);
printf("\nmessage sent\n");
close(c_sock);
return 0;
}
```

Receiver code:

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

int main()
{
int s_sock,c_client;
s_sock=socket(AF_INET,SOCK_STREAM,0);
struct sockaddr_in server,other;
memset(&server,0,sizeof(server));
memset(&other,0,sizeof(other));
server.sin_family=AF_INET;
server.sin_port=htons(9009);
server.sin_addr.s_addr=INADDR_ANY;
socklen_t addr_size;
bind(s_sock,(struct sockaddr*)&server,sizeof(server));
```

```
listen(s_sock,5);
addr_size=sizeof(other);
c_client=accept(s_sock,(struct sockaddr*)&other,&addr_size);

char msg[50];
char stringend[2],buf[2],ack[2];
int msglen=0;
recv(c_client,stringend,sizeof(stringend),0);
while(stringend[0]=='0')
{
recv(c_client,buf,sizeof(buf),0);
printf("\n%c",buf[0]);
char ack1[2];
printf("\nEnter Ack:(1/0):");
gets(ack1);
ack[0]=ack1[0];
send(c_client,ack,sizeof(ack),0);
if(ack[0]=='1')
{
msg[msglen++]=buf[0];
}
recv(c_client,stringend,sizeof(stringend),0);
}
msg[msglen++]='\0';
printf("\nReceived msg!\n");
for(int i=0;i<msglen;++i)
printf("%c",msg[i]);
printf("\n");
close(s_sock);
close(c_client);
return 0;
}
```

Output:

```
harish@harish-ubuntu:~/Desktop/comp_networks/lab8/stopnwait$ ./sender

Enter message to be sent:
message
1 1 1 0 1 1 0 1 1
message sent
```

```
harish@harish-ubuntu:~/Desktop/comp_networks/lab8/stopnwait$ ./receiver

m
Enter Ack:(1/0):1

e
Enter Ack:(1/0):1

s
Enter Ack:(1/0):1

s
Enter Ack:(1/0):0

s
Enter Ack:(1/0):1

a
Enter Ack:(1/0):1

g
Enter Ack:(1/0):0

g
Enter Ack:(1/0):1

e
Enter Ack:(1/0):1

Received msg!
message
```

Sliding Window protocol:

The two well known implementations of sliding window protocol are-

- Go back N Protocol
- Selective Repeat Protocol

Selective Repeat is implemented here.

In SR protocol, sender window size is always same as receiver window size.If a corrupted message is received Acknowledgement 0 is sent.Upon which the packet is sent in next frame.

How code works:

Sender side:

Client acts as sender.So client first connects with the server.Sender code gets the window size from user and transmits it to the receiver.Sender code gets the message to be sent from the user.At every iteration (an iteration per per window size number of character of the string),the sender sends a variable called stringend to notify to the receiver whether or not the string has ended.Window size number of characters are sent.Acknowledgement is received one by one.If any ack is 0 then it stops receiving further acknowledgements and starting from this character the next frame is sent.This happens until string end is reached.

Receiver side:
Server acts as receiver.So it accepts clients connection request.At every iteration it checks if the string has ended using the stringend variable sent by sender.If string has not ended, the window size number of characters are received and for every character acknowledgement is sent (1 for success and 0 for failure).The characters with ack 1 are stored in a character array and printed at last.

Sender code:

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

int main()
{
int c_sock;
//char buf[100];
//char bufr[100];
c_sock=socket(AF_INET,SOCK_STREAM,0);
struct sockaddr_in client;
memset(&client,0,sizeof(client));
client.sin_family=AF_INET;
client.sin_port=htons(9009);
client.sin_addr.s_addr=INADDR_ANY;
```

```
if(connect(c_sock,(struct sockaddr*)&client,sizeof(client))==-1)
{
printf("server is busy\nconnection failure");
return 0;
}

char buf[2];
buf[1]='\0';
char msg[50];
int window_size;
printf("\nEnter window size:\n");
scanf("%d",&window_size);
char ch;
scanf("%c",&ch);
int ws[]={window_size};
send(c_sock, ws, sizeof(ws), 0);
printf("\nEnter message to be sent:\n");
gets(msg);
char stringend[2];
stringend[0]='0';
stringend[1]='\0';
for(int i=0;i<strlen(msg);++i)
{
send(c_sock, stringend, sizeof(stringend), 0);
for(int j=0;j<window_size;j++)
{
printf("Sending: %c\n", msg[i+j]);
buf[0] = msg[i+j];
send(c_sock, buf, sizeof(buf), 0);
}
int ack_check = -1;
char ack[2];
for(int j=0;j<window_size;j++)
{
recv(c_sock, ack, sizeof(ack), 0);
printf("ACK: %s\n", ack);
if(ack[0] == '0')
```

```
{
ack_check = j;
j=window_size;
}
}
if(ack_check == -1) i = i + window_size-1;
else i = i + ack_check-1;
}
stringend[0]='1';
send(c_sock,stringend,sizeof(stringend),0);
printf("\nmessage sent\n");
close(c_sock);
return 0;
}
```

Receiver code:

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

int main()
{
int s_sock,c_client;
//char buf[100];
//char buf2[100];
s_sock=socket(AF_INET,SOCK_STREAM,0);
struct sockaddr_in server,other;
memset(&server,0,sizeof(server));
memset(&other,0,sizeof(other));
server.sin_family=AF_INET;
server.sin_port=htons(9009);
server.sin_addr.s_addr=INADDR_ANY;
```

```
socklen_t addr_size;
bind(s_sock,(struct sockaddr*)&server,sizeof(server));
listen(s_sock,5);
addr_size=sizeof(other);
c_client=accept(s_sock,(struct sockaddr*)&other,&addr_size);

char msg[50];
char stringend[2],buf[2],ack[2];
int msglen=0;
recv(c_client,stringend,sizeof(stringend),0);
while(stringend[0]=='0')
{
recv(c_client,buf,sizeof(buf),0);
printf("\n%c",buf[0]);
char ack1[2];
printf("\nEnter Ack:(1/0):");
//scanf("%s",ack1);
gets(ack1);
//printf("%c",ack1[0]);
ack[0]=ack1[0];
send(c_client,ack,sizeof(ack),0);
if(ack[0]=='1')
{
msg[msglen++]=buf[0];
}
recv(c_client,stringend,sizeof(stringend),0);
}
msg[msglen++]='\0';
printf("\nReceived msg!\n");
for(int i=0;i<msglen;++i)
printf("%c",msg[i]);
printf("\n");
close(s_sock);
close(c_client);
return 0;
}
```

Output:

```
harish@harish-ubuntu:~/Desktop/comp_networks/lab8/slidingwindow$ ./sender

Enter window size:
3

Enter message to be sent:
message
Sending: m
Sending: e
Sending: s
ACK: 1
ACK: 1
ACK: 1
Sending: s
Sending: a
Sending: g
ACK: 1
ACK: 0
Sending: a
Sending: g
Sending: e
ACK: 1
ACK: 0
Sending: g
Sending: e
Sending:
ACK: 1
ACK: 1
ACK: 1

message sent
```

```
harish@harish-ubuntu:~/Desktop/comp_networks/lab8/slidingwindow$ ./receiver
Buf: m
Buf: e
Buf: s
Ack: 1
Ack: 1
Ack: 1
Buf: s
Buf: a
Buf: g
Ack: 1
Ack: 0
Buf: a
Buf: g
Buf: e
Ack: 1
Ack: 0
Buf: g
Buf: e
Buf:
Ack: 1
Ack: 1
Ack: 1

Received msg!
message
```