

PRACTICE - 8

**Stop and Wait Protocol and Sliding Window Protocol using
Client-Server Communication Model of TCP**

**VIKNESH RAJARAMON
COE18B060**

STOP AND WAIT PROTOCOL

Working

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

How Code Works

Sender Side

Client acts as Sender. First the Client connects with the Server. We get the message to be sent from the user. We send the message character by character in every iteration. Also in every iteration, we check if the end of the input from the user is reached. A character is sent and an acknowledgement is received. If ACK is 0, the same character is sent again. Otherwise, the next character is sent.

Receiver Side

Server acts as Receiver. First it accepts the Client connection request. We receive the message character by character in every iteration. Also in every iteration, we check if the end of the input from the user is reached. For understanding this protocol, we generate random ACK value to be sent to the Client. If ACK is 0, it requests the Client to resend the data packet. Otherwise, it requests the Client to send the next data packet.

Sender Code (client.c)

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

#define PORT 9000
#define MAXSIZE 1024

int main()
{
    int sock_fd;
    char data[MAXSIZE];
```

```

memset(data,0,MAXSIZE);
struct sockaddr_in Client;

sock_fd=socket(AF_INET,SOCK_STREAM,0);
if(sock_fd<0)
{
    printf("Socket Creation failed...");
    exit(0);
}

memset(&Client,0,sizeof(Client));
Client.sin_family=AF_INET;
Client.sin_port=htons(PORT);
Client.sin_addr.s_addr=INADDR_ANY;

if(connect(sock_fd,(struct sockaddr *) &Client,sizeof(Client))== -1)
{
    printf("Connection failed...");
    exit(0);
}

printf("Enter data to send : ");
fgets(data,MAXSIZE,stdin);

int length=strlen(data)-1;
int ack=1;
int exit_status=0;
int i=0;
printf("Sending Data...\n");

while(i<length)
{
    send(sock_fd,&exit_status,sizeof(exit_status),0);
    char buf=data[i];
    ack=0;
    while(ack==0)
    {
        send(sock_fd,&buf,sizeof(buf),0);
        printf("Sending : %c\n",buf);
        recv(sock_fd,&ack,sizeof(ack),0);
        printf("ACK : %d\n",ack);
    }
    ++i;
}

exit_status=1;
send(sock_fd,&exit_status,sizeof(exit_status),0);

```

```

        printf("Data Sent...\n");

        close(sock_fd);
        return 0;
}

```

Receiver Code (server.c)

```

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

#define PORT 9000
#define MAXSIZE 1024

int main()
{
    srand(time(0));
    int c_sockfd,s_sockfd;
    char data[MAXSIZE];
    memset(data,0,MAXSIZE);
    struct sockaddr_in Server,Other;

    memset(&Other,0,sizeof(Other));
    socklen_t size=sizeof(Other);

    s_sockfd=socket(AF_INET,SOCK_STREAM,0);
    if(s_sockfd<0)
    {
        printf("Socket Creation failed...");
        exit(0);
    }

    memset(&Server,0,sizeof(Server));
    Server.sin_family=AF_INET;
    Server.sin_port=htons(PORT);
    Server.sin_addr.s_addr=INADDR_ANY;

    if(bind(s_sockfd,(struct sockaddr *) &Server,sizeof(Server))== -1)
    {
        printf("Socket Bind failed...");
        exit(0);
    }
}

```

```

}

listen(s_sockfd,5);

c_sockfd=accept(s_sockfd,(struct sockaddr *)&Other,&size);
if(c_sockfd==-1)
{
    printf("Socket Accept failed...\n");
    exit(0);
}

int ack=0;
int exit_status=0;
int i=0;
char buf;
int receive=0;
printf("Receiving Data...\n");

while(1)
{
    recv(c_sockfd,&exit_status,sizeof(exit_status),0);
    if(exit_status==1)
    {
        break;
    }
    receive=0;
    do
    {
        receive=recv(c_sockfd,&buf,sizeof(buf),0);
        printf("Receiving : %c\n",buf);
        ack=rand()%2;
        printf("ACK : %d\n",ack);
        send(c_sockfd,&ack,sizeof(ack),0);
    }while(ack==0);
    data[i]=buf;
    ++i;
}

printf("Data Received...\n");

printf("\nReceived Data : %s\n",data);

close(s_sockfd);
close(c_sockfd);
return 0;
}

```

Output

```
viknesh@viknesh-ubuntu: ~/Documents/CN/Lab_8/Stop_and_Wait_Protocol
viknesh@viknesh-ubuntu:~/Documents/CN/Lab_8/Stop_and_Wait_Protocol$ gcc client.c -o client
viknesh@viknesh-ubuntu:~/Documents/CN/Lab_8/Stop_and_Wait_Protocol$ ./client
Enter data to send : hello
Sending Data...
Sending : h
ACK : 0
Sending : h
ACK : 1
Sending : e
ACK : 0
Sending : e
ACK : 0
Sending : e
ACK : 1
Sending : l
ACK : 1
Sending : l
ACK : 0
Sending : l
ACK : 0
Sending : l
ACK : 0
Sending : l
ACK : 0
Sending : l
ACK : 0
Sending : l
ACK : 0
Sending : l
ACK : 1
Sending : o
ACK : 0
Sending : o
ACK : 0
Sending : o
ACK : 1
Data Sent...
viknesh@viknesh-ubuntu:~/Documents/CN/Lab_8/Stop_and_Wait_Protocol$
```

```
viknesh@viknesh-ubuntu: ~/Documents/CN/Lab_8/Stop_and_Wait_Protocol
viknesh@viknesh-ubuntu:~/Documents/CN/Lab_8/Stop_and_Wait_Protocol$ gcc server.c -o server
viknesh@viknesh-ubuntu:~/Documents/CN/Lab_8/Stop_and_Wait_Protocol$ ./server
Receiving Data...
Receiving : h
ACK : 0
Receiving : h
ACK : 1
Receiving : e
ACK : 0
Receiving : e
ACK : 0
Receiving : e
ACK : 1
Receiving : l
ACK : 1
Receiving : l
ACK : 0
Receiving : l
ACK : 0
Receiving : l
ACK : 0
Receiving : l
ACK : 0
Receiving : l
ACK : 0
Receiving : l
ACK : 1
Receiving : o
ACK : 0
Receiving : o
ACK : 0
Receiving : o
ACK : 1
Data Received...

Received Data : hello
viknesh@viknesh-ubuntu:~/Documents/CN/Lab_8/Stop_and_Wait_Protocol$
```

SLIDING WINDOW PROTOCOL (SELECTIVE REPEAT ARQ)

Working

The two implementations of Sliding Window Protocol are:

- Go-Back-N ARQ
- Selective Repeat ARQ

Selective Repeat ARQ is implemented here.

In Selective Repeat ARQ, Sender's window size is the same as Receiver's window size. If a message received in the Receiver's end is corrupted, an ACK of 0 is sent following which the same message is sent in the next frame.

How Code Works

Sender Side

Client acts as Sender. First the Client connects with the Server. We get the message to be sent from the user. We send the message character by character in every iteration. Also in every iteration, **window_size** number of characters is sent. An acknowledgement is received for each character one after the other. If any ACK is 0, then it stops receiving further acknowledgements and starting from that character, the next frame is sent.

Receiver Side

Server acts as Receiver. First it accepts the Client connection request. We receive the message character by character in every iteration. Also in every iteration, **window_size** number of characters is sent. For understanding this protocol, we generate random ACK value to be sent to the Client. The characters with ACK 1 are stored in the array.

Sender Code (client.c)

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

```
#define PORT 9000
#define MAXSIZE 1024
```

```
int main()
{
```

```

int sock_fd;
char data[MAXSIZE];
memset(data,0,MAXSIZE);
struct sockaddr_in Client;
int window_size;

sock_fd=socket(AF_INET,SOCK_STREAM,0);
if(sock_fd<0)
{
    printf("Socket Creation failed...");
    exit(0);
}

memset(&Client,0,sizeof(Client));
Client.sin_family=AF_INET;
Client.sin_port=htons(PORT);
Client.sin_addr.s_addr=INADDR_ANY;

if(connect(sock_fd,(struct sockaddr *) &Client,sizeof(Client))== -1)
{
    printf("Connection failed...");
    exit(0);
}

printf("Enter the window size : ");
scanf("%d",&window_size);
getchar();

printf("Enter data to send : ");
fgets(data,MAXSIZE,stdin);

int length=strlen(data)-1;
int ack=0;
int exit_status=0;
int i=0;

send(sock_fd,&window_size,sizeof(window_size),0);

printf("Sending Data...\n");

while(i<length)
{
    send(sock_fd,&exit_status,sizeof(exit_status),0);
    char buf;
    int j=0;
    for(j=0;j<window_size;++j)
    {
        if(j==(length-i))

```



```

        {
            break;
        }
        printf("Sending : %c\n",data[i+j]);
        buf=data[i+j];
        send(sock_fd,&buf,sizeof(buf),0);
    }
    if((i+j)==length)
    {
        buf='\0';
        send(sock_fd,&buf,sizeof(buf),0);
    }
    int ack_count=-1;
    for(j=0;j<window_size;++j)
    {
        if(j==(length-i))
        {
            break;
        }
        recv(sock_fd,&ack,sizeof(ack),0);
        printf("ACK : %d\n",ack);
        if(ack==0)
        {
            ack_count=j;
            break;
        }
    }
    if(ack_count== -1)
    {
        i=i+window_size;
    }
    else
    {
        i=i+ack_count;
    }
}

exit_status=1;
send(sock_fd,&exit_status,sizeof(exit_status),0);

printf("Data Sent...\n");

close(sock_fd);
return 0;
}

```

Receiver Code (server.c)

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

#define PORT 9000
#define MAXSIZE 1024

int main()
{
    srand(time(0));
    int c_sockfd,s_sockfd;
    char data[MAXSIZE];
    memset(data,0,MAXSIZE);
    struct sockaddr_in Server,Other;

    memset(&Other,0,sizeof(Other));
    socklen_t size=sizeof(Other);

    s_sockfd=socket(AF_INET,SOCK_STREAM,0);
    if(s_sockfd<0)
    {
        printf("Socket Creation failed...");
        exit(0);
    }

    memset(&Server,0,sizeof(Server));
    Server.sin_family=AF_INET;
    Server.sin_port=htons(PORT);
    Server.sin_addr.s_addr=INADDR_ANY;

    if(bind(s_sockfd,(struct sockaddr *) &Server,sizeof(Server))== -1)
    {
        printf("Socket Bind failed...");
        exit(0);
    }

    listen(s_sockfd,5);

    c_sockfd=accept(s_sockfd,(struct sockaddr *)&Other,&size);
    if(c_sockfd== -1)
```

```

{
    printf("Socket Accept failed...\n");
    exit(0);
}

int exit_status=0;
int i=0;
char buf;
int receive=0;

int window_size;

recv(c_sockfd,&window_size,sizeof(window_size),0);

int ack;

printf("Receiving Data...\n");

while(1)
{
    recv(c_sockfd,&exit_status,sizeof(exit_status),0);
    if(exit_status==1)
    {
        break;
    }
    int count=window_size;
    for(int j=0;j<window_size;++j)
    {
        recv(c_sockfd,&buf,sizeof(buf),0);
        if(buf=='\0')
        {
            count=j;
            break;
        }
        printf("Received : %c\n",buf);
        data[i]=buf;
        ++i;
    }
    for(int j=0;j<count;++j)
    {
        ack=rand()%2;
        printf("ACK : %d\n",ack);
        send(c_sockfd,&ack,sizeof(ack),0);
        if(ack==0)
        {
            i=i-window_size+j;
            break;
        }
    }
}

```

```

        }
        if((buf=='\0') && (ack==1))
        {
            break;
        }
    }

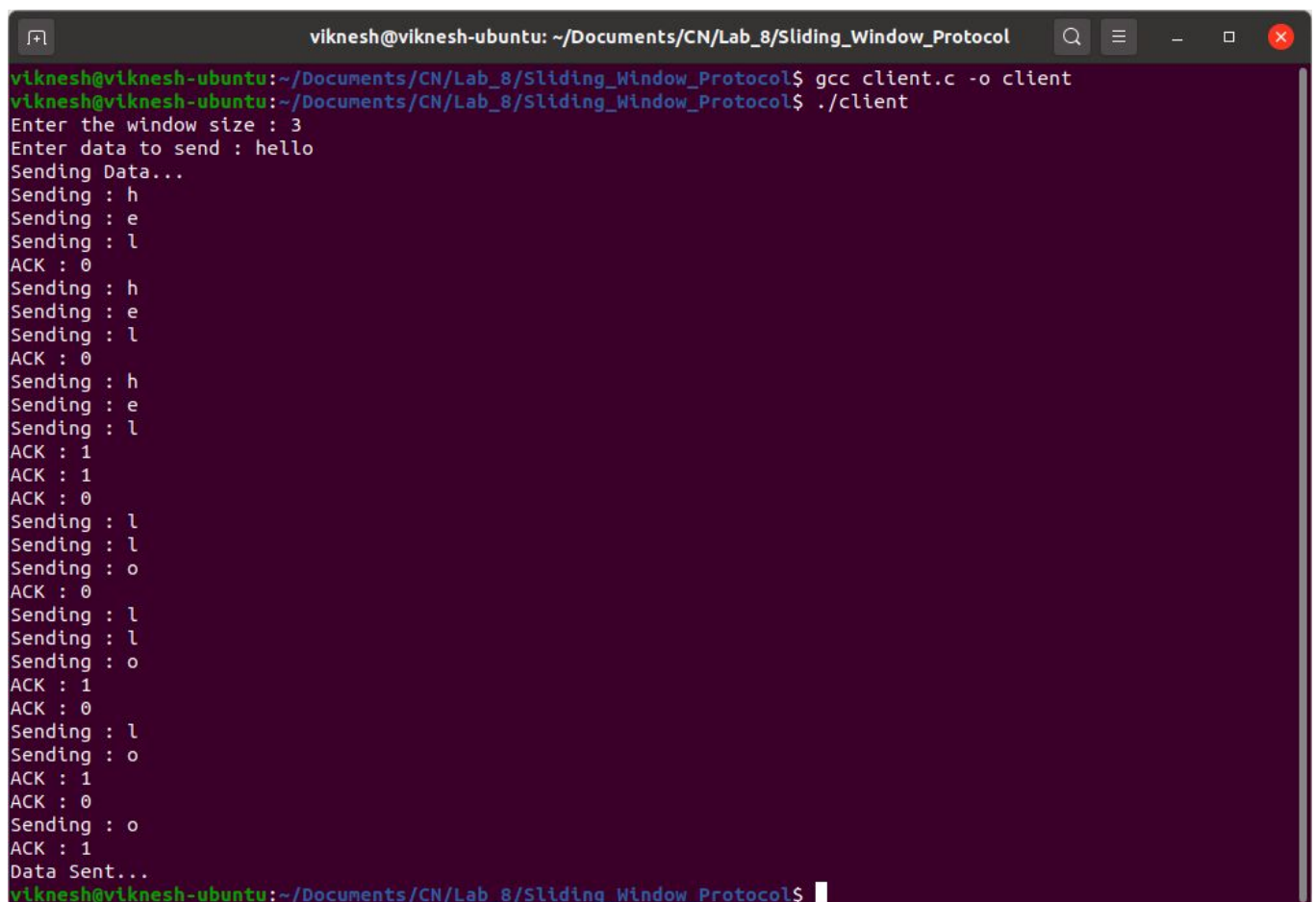
    printf("Data Received...\n");

    printf("\nReceived Data : %s\n",data);

    close(s_sockfd);
    close(c_sockfd);
    return 0;
}

```

Output



```

viknesh@viknesh-ubuntu: ~/Documents/CN/Lab_8/Sliding_Window_Protocol
viknesh@viknesh-ubuntu:~/Documents/CN/Lab_8/Sliding_Window_Protocol$ gcc client.c -o client
viknesh@viknesh-ubuntu:~/Documents/CN/Lab_8/Sliding_Window_Protocol$ ./client
Enter the window size : 3
Enter data to send : hello
Sending Data...
Sending : h
Sending : e
Sending : l
ACK : 0
Sending : h
Sending : e
Sending : l
ACK : 0
Sending : h
Sending : e
Sending : l
ACK : 1
ACK : 1
ACK : 0
Sending : l
Sending : l
Sending : o
ACK : 0
Sending : l
Sending : l
Sending : o
ACK : 1
ACK : 0
Sending : l
Sending : o
ACK : 1
ACK : 0
Sending : o
ACK : 1
Data Sent...
viknesh@viknesh-ubuntu:~/Documents/CN/Lab_8/Sliding_Window_Protocol$

```

```
viknesh@viknesh-ubuntu: ~/Documents/CN/Lab_8/Sliding_Window_Protocol
viknesh@viknesh-ubuntu:~/Documents/CN/Lab_8/Sliding_Window_Protocol$ gcc server.c -o server
viknesh@viknesh-ubuntu:~/Documents/CN/Lab_8/Sliding_Window_Protocol$ ./server
Receiving Data...
Received : h
Received : e
Received : l
ACK : 0
Received : h
Received : e
Received : l
ACK : 0
Received : h
Received : e
Received : l
ACK : 1
ACK : 1
ACK : 0
Received : l
Received : l
Received : o
ACK : 0
Received : l
Received : l
Received : o
ACK : 1
ACK : 0
Received : l
Received : o
ACK : 1
ACK : 0
Received : o
ACK : 1
Data Received...
Received Data : heloo
viknesh@viknesh-ubuntu:~/Documents/CN/Lab_8/Sliding_Window_Protocol$
```