Experiment – 2

AIM- To implement stop and wait protocol.

Description -

The Data link layer is used for error and flow control in computer networking. Stop and wait protocol is a data link layer protocol that helps to implement both error control and flow control.

In this protocol, the sender sends one frame at a time and waits for an acknowledgment before sending the next one. To detect corrupted frames, we need to add a CRC to each data frame. When a frame arrives at the receiver site, it is checked. If its CRC is incorrect, the frame is corrupted and silently discarded. The silence of the receiver is a signal for the sender that a frame was either corrupted or lost. Every time the sender sends a frame, it starts a timer. If an acknowledgment arrives before the timer expires, the timer is stopped and the sender sends the next frame (if it has one to send). If the timer expires, the sender resends the previous frame, assuming that the frame was either lost or corrupted.

This means that the sender needs to keep a copy of the frame until its acknowledgment arrives. When the corresponding acknowledgment arrives, the sender discards the copy and sends the next frame if it is ready.

Algorithms:

The stop and wait algorithm is based on rules for both sender and receiver.

Sender:
1) Send one data packet at a time.
2) Send next packet only after receiving acknowledgement for previous.

Receiver:
1) Send acknowledgement after receiving and consuming of data packet.
2) After consuming packet acknowledgement need to be sent (Flow Control).

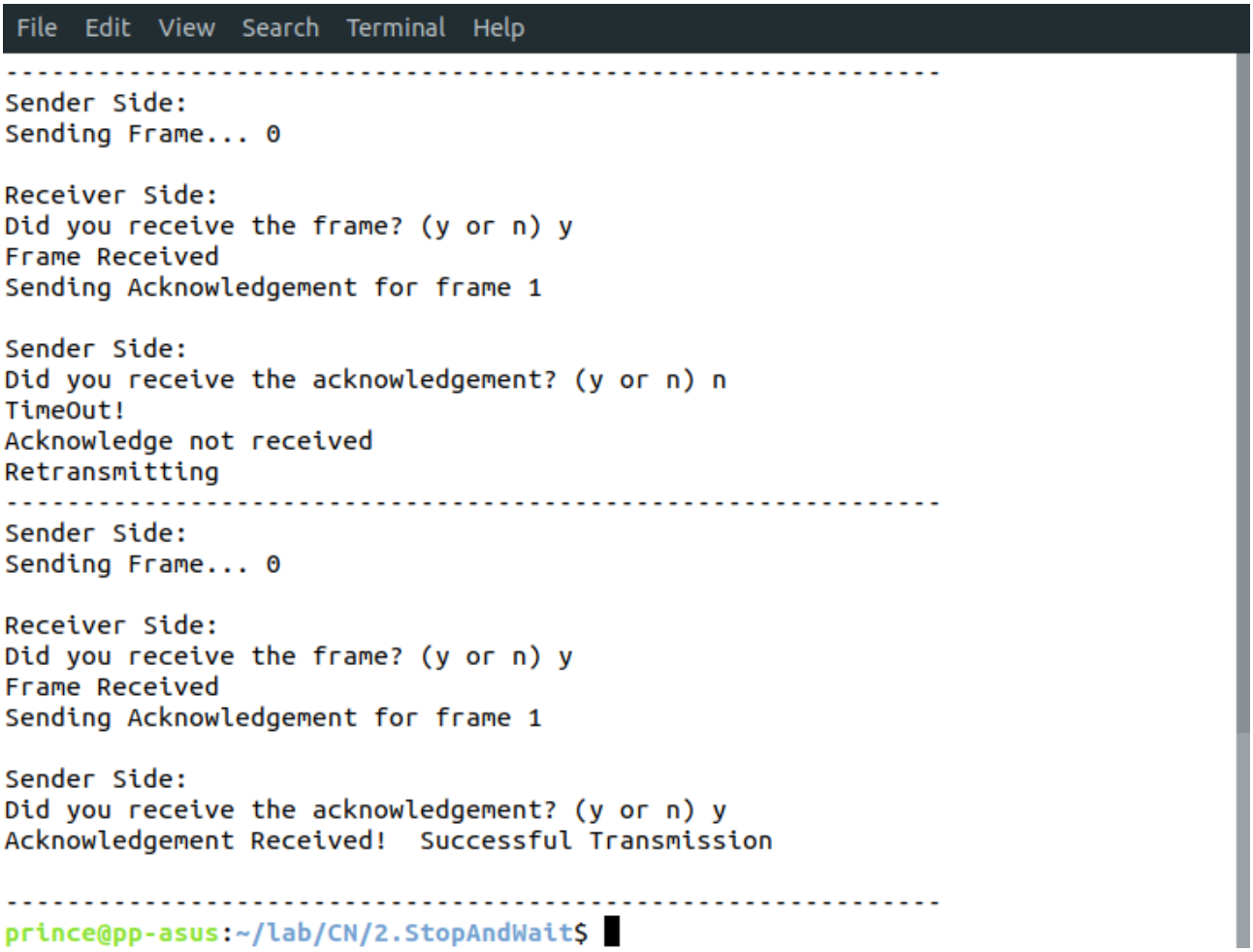
Code:

```
#include<bits/stdc++.h>

using namespace std;

int main(){
    int n;
    char ch;
    cout<<"Enter the number of frames: ";
    cin>>n;
    cout<<endl;

    for (int i=0; i<n; i++){
        cout<<"Sender Side:"<<endl;
        cout<<"Sending Frame... "<<(i%2)<<endl<<endl;

        cout<<"Receiver Side:"<<endl;
        cout<<"Did you receive the frame? (y or n) ";
        cin>>ch;
        if(ch == 'y'){
            cout<<"Frame Received" << endl;
            cout<<"Sending Acknowledgement for frame "
                << !(i%2) << endl<<endl;

            cout<<"Sender Side:" << endl;
            cout<<"Did you receive the acknowledgement? (y or n) ";
            cin>>ch;

            if(ch == 'y'){
                cout<<"Acknowledgement Received!  ";
                cout<<"Successful Transmission"<<endl<<endl;

            }else{
                cout<<"TimeOut!"<<endl;
                cout<<"Acknowledge not received"<<endl;
                cout << "Retransmitting"<<endl;
                i--;
            }
        }else{
            cout<<"Frame Not Received!"<<endl<<endl;
            cout<<"Sender Side:"<<endl;
            cout<<"TimeOut!"<<endl;
```

```
            cout<<"Acknowledge not received"<<endl;
            cout<<"Retransmitting"<<endl;
            i--;
        }
        cout << "----------------------------------------------------------" << endl;
    }

    return 0;
}
```

Output:

```
File  Edit  View  Search  Terminal  Help
prince@pp-asus:~/lab/CN/2.StopAndWait$ g++ stopWait.cpp
prince@pp-asus:~/lab/CN/2.StopAndWait$ ./a.out
Enter the number of frames: 3

Sender Side:
Sending Frame... 0

Receiver Side:
Did you receive the frame? (y or n) y
Frame Received
Sending Acknowledgement for frame 1

Sender Side:
Did you receive the acknowledgement? (y or n) y
Acknowledgement Received!  Successful Transmission

----------------------------------------------------------
Sender Side:
Sending Frame... 1

Receiver Side:
Did you receive the frame? (y or n) n
Frame Not Received!

Sender Side:
TimeOut!
Acknowledge not received
Retransmitting
----------------------------------------------------------
Sender Side:
Sending Frame... 1

Receiver Side:
Did you receive the frame? (y or n) y
Frame Received
Sending Acknowledgement for frame 0

Sender Side:
Did you receive the acknowledgement? (y or n) y
Acknowledgement Received!  Successful Transmission
```



Learnings –

1. Stop and Wait ARQ mainly implements Sliding Window Protocol concept with Window Size 1.
2. If Bandwidth*Delay product is very high, then stop and wait protocol is not so useful. The sender has to keep waiting for acknowledgements before sending the processed next packet.
3. The Stop and Wait ARQ solves main three problems, but may cause big performance issues as sender always waits for acknowledgement even if it has next packet ready to send.
4. So Stop and Wait ARQ may work fine where propagation delay is very less for example LAN connections, but performs badly for distant connections like satellite connection.