Experiment-6

**Date**-May 20, 2021.

**AIM-** Implement and study of Stop and Wait Protocol.

**Stop and Wait Protocol**

Before understanding the stop and Wait protocol, we first know about the error control mechanism. The error control mechanism is used so that the received data should be exactly same whatever sender has sent the data. The error control mechanism is divided into two categories, i.e., Stop and Wait ARQ and sliding window. The sliding window is further divided into two categories, i.e., Go Back N, and Selective Repeat. Based on the usage, the people select the error control mechanism whether it is **stop and wait** or **sliding window**.

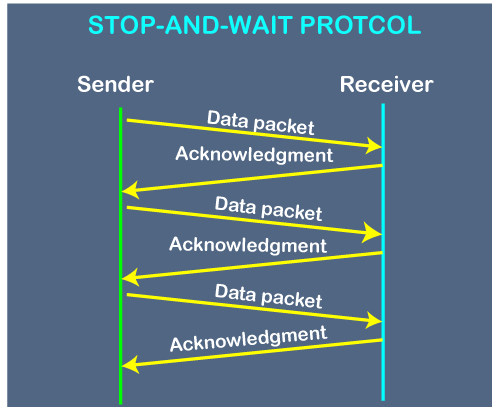
Stop and wait means, whatever the data that sender wants to send, he sends the data to the receiver. After sending the data, he stops and waits until he receives the acknowledgment from the receiver. The stop and wait protocol is a flow control protocol where flow control is one of the services of the data link layer.

It is a data-link layer protocol which is used for transmitting the data over the noiseless channels. It provides unidirectional data transmission which means that either sending or receiving of data will take place at a time. It provides flow-control mechanism but does not provide any error control mechanism.

The idea behind the usage of this frame is that when the sender sends the frame then he waits for the acknowledgment before sending the next frame.

### **Working of Stop and Wait protocol**

The figure below shows the working of the stop and wait protocol. If there is a sender and receiver, then sender sends the packet and that packet is known as a data packet. The sender will not send the second packet without receiving the acknowledgment of the first packet. The receiver sends the acknowledgment for the data packet that it has received. Once the acknowledgment is received, the sender sends the next packet. This process continues until all the packet are not sent. The main advantage of this protocol is its simplicity but it has some disadvantages also. For example, if there are 1000 data packets to be sent, then all the 1000 packets cannot be sent at a time as in Stop and Wait protocol, one packet is sent at a time.



### **Primitives of Stop and Wait Protocol**

**The primitives of stop and wait protocol are:**

**Sender side:**

**Rule 1:** Sender sends one data packet at a time.

**Rule 2:** Sender sends the next packet only when it receives the acknowledgment of the previous packet.

Therefore, the idea of stop and wait protocol in the sender's side is very simple, i.e., send one packet at a time, and do not send another packet before receiving the acknowledgment.

### **Receiver side:**

**Rule 1:** Receive and then consume the data packet.

**Rule 2:** When the data packet is consumed, receiver sends the acknowledgment to the sender.

Therefore, the idea of stop and wait protocol in the receiver's side is also very simple, i.e., consume the packet, and once the packet is consumed, the acknowledgment is sent. This is known as a flow control mechanism.

**PROGRAM (C++)**

#include<iostream>

using namespace std;

int main() {

cout<<"Enter number of frames : ";

int n;

cin>>n;

int max\_time = 5;

cout<<"Max Time for receving ack : "<<max\_time << endl;

int totaltime = 0;

int frame\_number = 1;

while(frame\_number <= n){

cout<<"\nFrame number : "<<frame\_number<<endl;

cout<<"Enter time to receive the Ack : ";

int time\_taken;

cin>>time\_taken;

cout<<endl;

if(time\_taken <= max\_time){

cout<<"Acknowledgement Received\n"<<endl;

totaltime += time\_taken;

frame\_number++;

}

else{

cout<<"No Acknowledgement Received\n"<<endl;

totaltime += time\_taken - max\_time;

}

}

cout<<"Total time : "<<totaltime;

return 0;

}

**OUTPUT-**

