Go-Back-N Protocol

Consider a 50-kbps satellite channel with a 500-msec round-trip propagation delay. Let us imagine trying to use protocol 4 to send 1000-bit frames via the satellite. At t = 0 the sender starts sending the first frame. At t = 20 msec the frame has been completely sent. Not until t = 270 msec has the frame fully arrived at the receiver, and not until t = 520 msec has the acknowledgment arrived back at the sender, under the best of circumstances (of no waiting in the receiver and a short acknowledgment frame).

The problem described here can be viewed as a consequence of the rule requiring a sender to wait for an acknowledgement before sending another frame. If we relax that restriction, much better efficiency can be achieved. Basically, the solution lies in allowing the sender to transmit up to w frames before blocking, instead of just 1.

To find an appropriate value for w we need to know how many frames can fit inside the channel as they propagate from sender to receiver. This capacity is determined by the bandwidth in bits/sec multiplied by the one-way transit time, or the bandwidth-delay product of the link.

Two basic approaches are available for dealing with errors in the presence of pipelining : One option, called **go-back-n**, is for the receiver simply to discard all subsequent frames, sending no acknowledgments for the discarded frames. This strategy corresponds to a receive window of size 1. In other words, the data link layer refuses to accept any frame except the next one it must give to the network layer. If the sender's window fills up before the timer runs out, the pipeline will begin to empty.

The other general strategy for handling errors when frames are pipelined is called **selective repeat.**

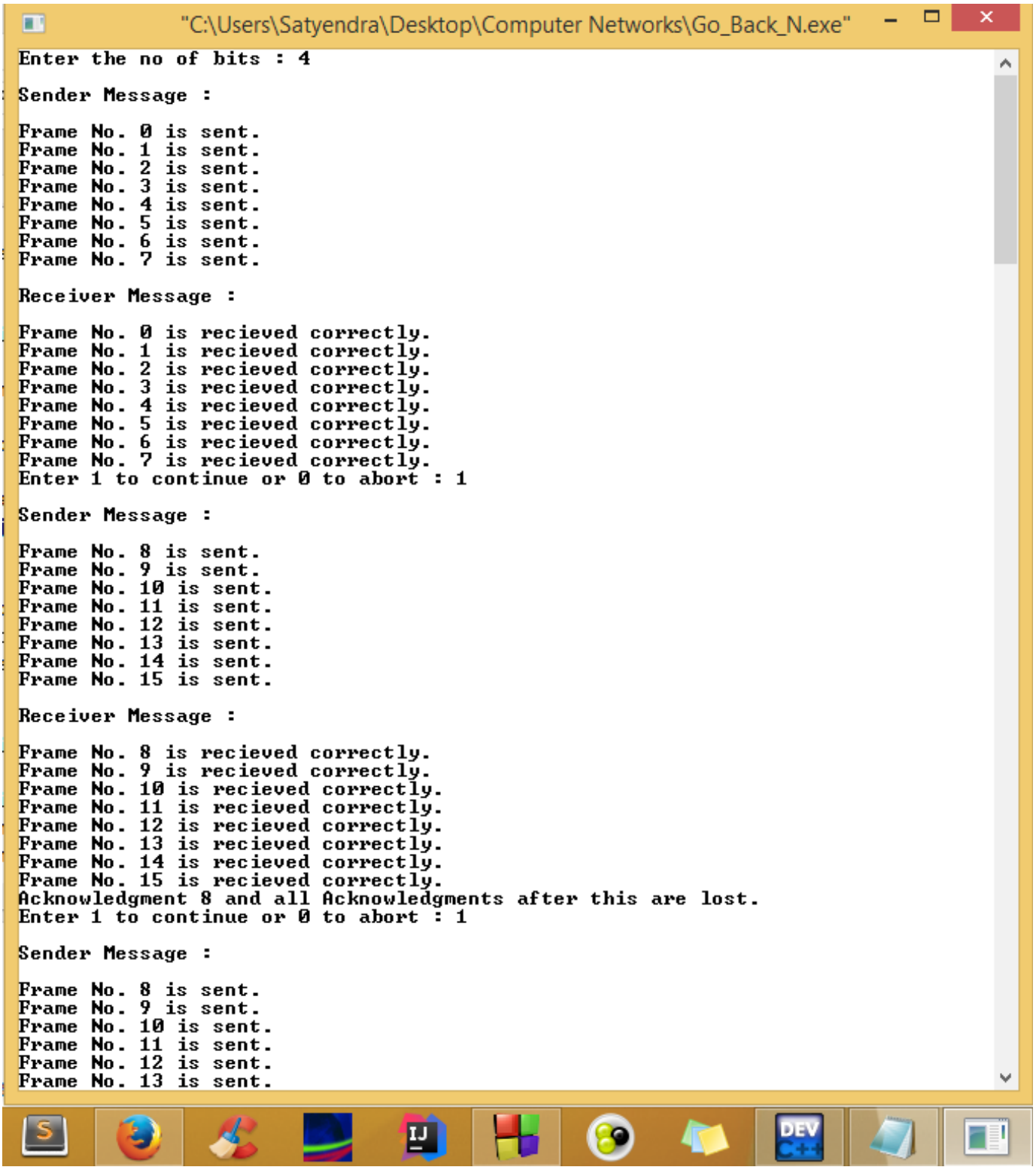
Code of Go-Back-N Protocol

```
#include<cstdlib>
#include<iostream>
#define Pipeline_Size 16
#define Max_No_of_Frames 1000
using namespace std;
class Head
{
   private:

   int Data[Max_No_of_Frames], Pipeline[Pipeline_Size], Bits_Sent,
      Bits_Received, No_of_bits_to_send;

   public:

   Head()
   {
      Bits_Sent=Bits_Received=0;
   }
   void Scan()
   {
      int No_of_Bits,No_of_Frames;
      cout<<"Enter the no of bits : ";
      cin>>No_of_Bits;
      No_of_Frames = 2<<(No_of_Bits-1);
      No_of_bits_to_send = (No_of_Frames/2);
      for(int j=0, Frame_No=0 ; j<Max_No_of_Frames ; Frame_No%=No_of_Frames)
         Data[j++]=Frame_No++;
      Event_Send(No_of_Frames);
   }

   void Event_Send(int Frames_Send)
   {
      cout <<"\nSender Message :\n";
      for (int k=0,i=Bits_Sent ; i<Bits_Sent+No_of_bits_to_send ; )
      {
         Pipeline[k]=Data[i++];
         cout<<"\nFrame No. "<<Pipeline[k++]<<" is sent.";
      }
      Event_Receive(Frames_Send);
   }
```

```
void  Event_Receive(int Frames_Received)
{
    int Trans_Error,Lost_Index,Ack_Lost_Prob,Damage;
    bool cont;
    cout <<"\n\nReceiver Message :\n";
    Trans_Error = rand() % 5;

    /* We have assumed if Damage_Prob=0 then transmission error happened
    otherwise we received data correctly. */

    if(Trans_Error!=0)
    {
      for (int i=0 ; i<No_of_bits_to_send ; i++)
      {
        if (Bits_Received == Pipeline[i])
        {
          cout <<"\nFrame No. "<<Pipeline[i]<<" is recieved correctly.";
          ++Bits_Received;
          Bits_Received%=Frames_Received;
        }
        else
          cout << "\nDuplicate Frame No. "<<Pipeline[i]<<" is discarded.";
      }
      Ack_Lost_Prob = rand() % 20;

            /* We have assumed if Ack_Lost_Prob is between 0 and 5 then
acknowledgments
      are lost otherwise we received all acknowledgments correctly. */

      if (Ack_Lost_Prob>=0 && Ack_Lost_Prob<5)
      {
            cout << "\nAcknowledgment "<<Pipeline[Ack_Lost_Prob]<<" and all
Acknowledgments after this are lost.";
        Bits_Sent = Pipeline[Ack_Lost_Prob];
      }
      else
        Bits_Sent=(Bits_Sent+No_of_bits_to_send)%Frames_Received;
    }
    else
    {
      Lost_Index=rand()%No_of_bits_to_send;
```

```
        // Lost_Index is the index of the frame being lost.

        for (int i = 0 ; i < Lost_Index ; i++)
        {
           if (Bits_Received == Pipeline[i])
           {
              cout<<"\nFrame No. "<<Pipeline[i]<<" is recieved correctly.";
              ++Bits_Received;
              Bits_Received%=Frames_Received;
           }
           else
              cout<<"\nDuplicate frame "<<Pipeline[i]<<" discarded.";
        }

        Damage = rand() % 2;
        // If Damage == 0 Frame damaged otherwise Frame lost.
        if(!Damage)
           cout <<"\nFrame No. "<<Pipeline[Lost_Index]<<" is damaged.";
        else
           cout <<"\nFrame No. "<<Pipeline[Lost_Index]<<" is lost.";
        for (int i=Lost_Index+1;i<No_of_bits_to_send; i++)
           cout << "\nFrame  No. "<<Pipeline[i]<<" is discarded.";
        cout<<"\nSender's Timeout thus Resend the Frame.";
        Bits_Sent = Pipeline[Lost_Index];
      }
      cout << "\nEnter 1 to continue or 0 to abort : ";
      cin >> cont;
      if (cont == 1)
        Event_Send(Frames_Received);
      else
        exit(0);
   }
};
int  main()
{
   Head Go_Back_N;
   Go_Back_N.Scan();
}
```

RESULT : -


"C:\Users\Satyendra\Desktop\Computer Networks\Go_Back_N.exe"
Enter the no of bits : 4
Sender Message :
Frame No. 0 is sent.
Frame No. 1 is sent.
Frame No. 2 is sent.
Frame No. 3 is sent.
Frame No. 4 is sent.
Frame No. 5 is sent.
Frame No. 6 is sent.
Frame No. 7 is sent.
Receiver Message :
Frame No. 0 is recieved correctly.
Frame No. 1 is recieved correctly.
Frame No. 2 is recieved correctly.
Frame No. 3 is recieved correctly.
Frame No. 4 is recieved correctly.
Frame No. 5 is recieved correctly.
Frame No. 6 is recieved correctly.
Frame No. 7 is recieved correctly.
Enter 1 to continue or 0 to abort : 1
Sender Message :
Frame No. 8 is sent.
Frame No. 9 is sent.
Frame No. 10 is sent.
Frame No. 11 is sent.
Frame No. 12 is sent.
Frame No. 13 is sent.
Frame No. 14 is sent.
Frame No. 15 is sent.
Receiver Message :
Frame No. 8 is recieved correctly.
Frame No. 9 is recieved correctly.
Frame No. 10 is recieved correctly.
Frame No. 11 is recieved correctly.
Frame No. 12 is recieved correctly.
Frame No. 13 is recieved correctly.
Frame No. 14 is recieved correctly.
Frame No. 15 is recieved correctly.
Acknowledgment 8 and all Acknowledgments after this are lost.
Enter 1 to continue or 0 to abort : 1
Sender Message :
Frame No. 8 is sent.
Frame No. 9 is sent.
Frame No. 10 is sent.
Frame No. 11 is sent.
Frame No. 12 is sent.
Frame No. 13 is sent.