Title: Write a program to simulate Go back N Sliding Window Protocol in peer-to-peer mode.

Sliding Window Protocol

The sliding window is a technique for sending multiple frames at a time. It controls the data packets between the two devices where reliable and gradual delivery of data frames is needed. It is also used in TCP (Transmission Control Protocol).

In this technique, each frame has sent from the sequence number. The sequence numbers are used to find the missing data in the receiver end. The purpose of the sliding window technique is to avoid duplicate data, so it uses the sequence number.

Types of Sliding Window Protocol Sliding window protocol has two types:

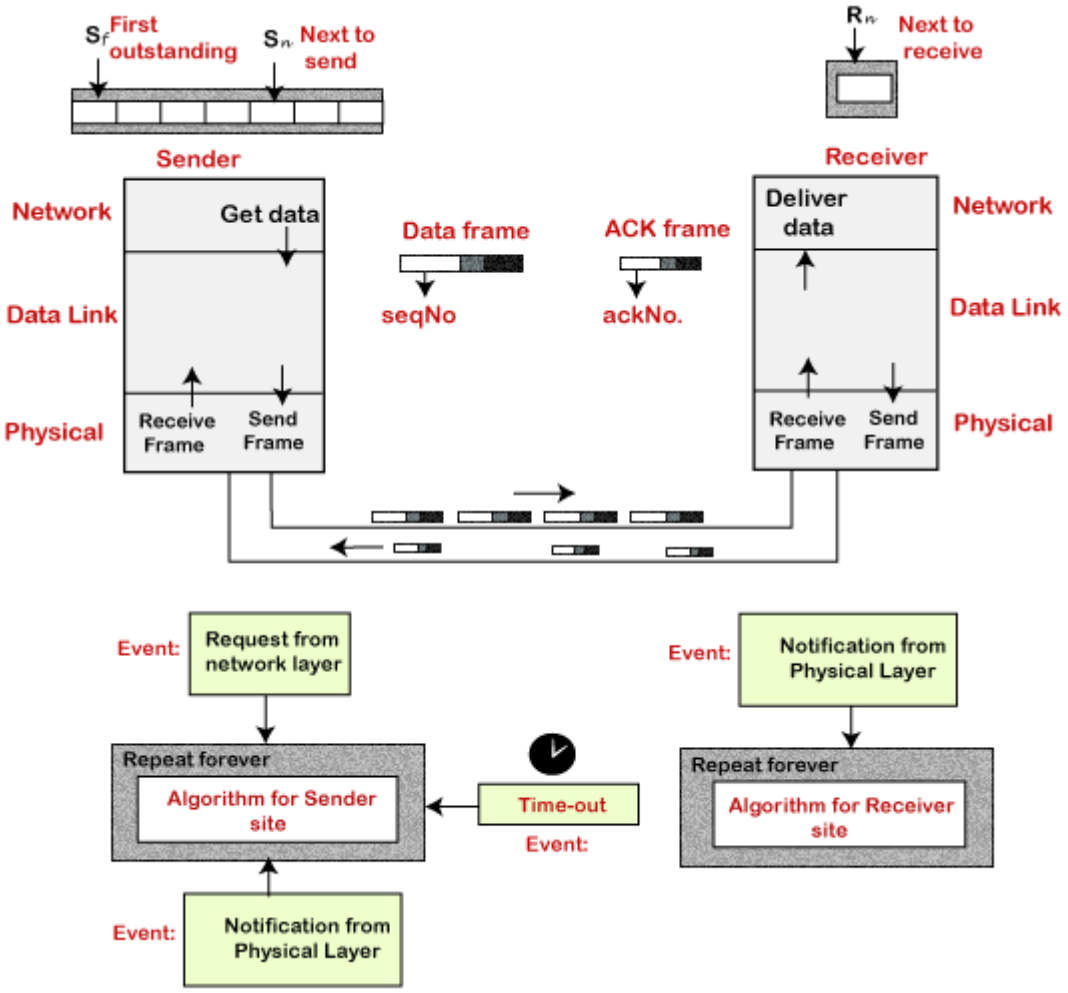
- Go-Back-N ARQ
- Selective Repeat ARQ

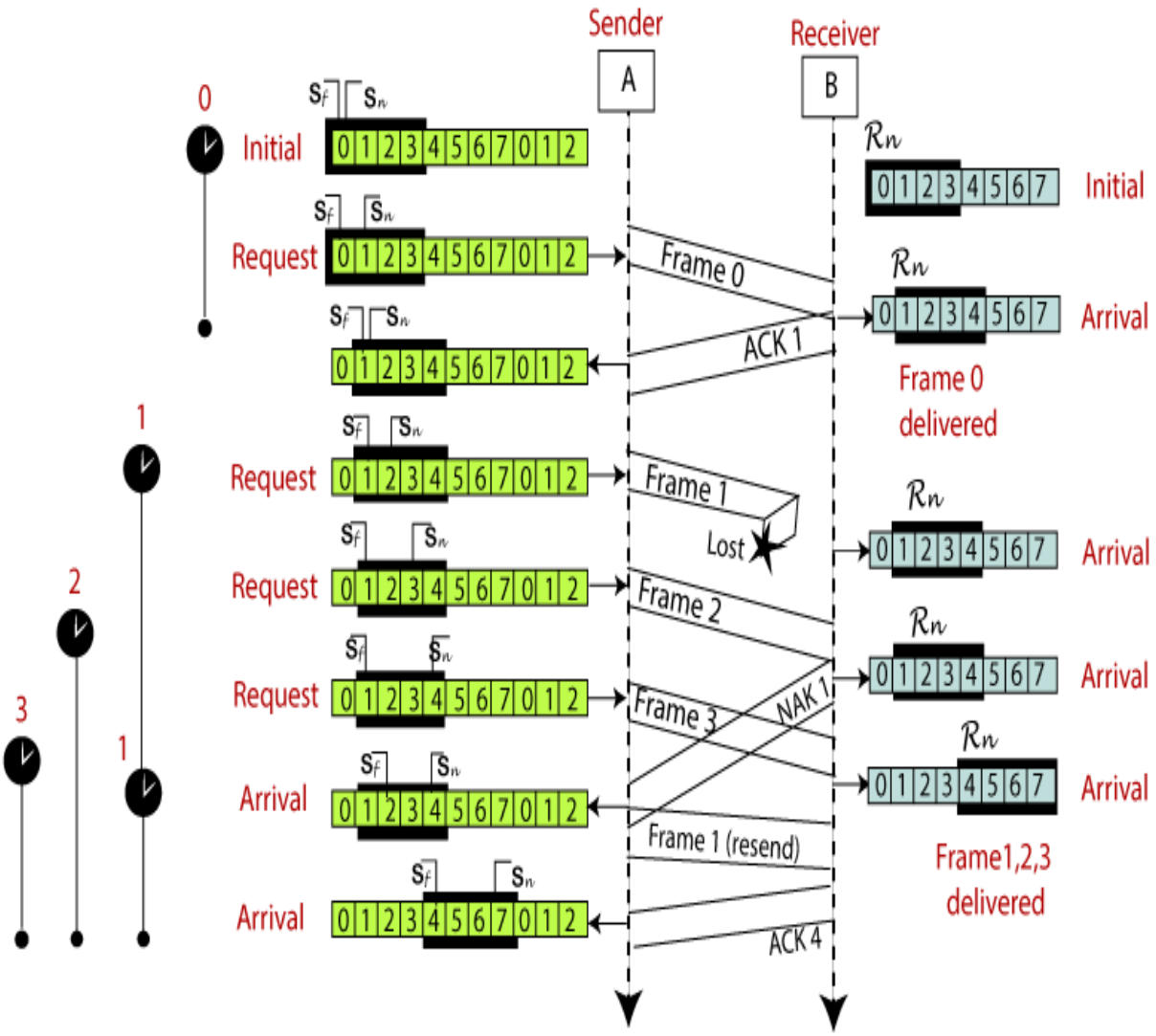
Go-Back-N Sliding

Go-Back-N ARQ protocol is also known as Go-Back-N Automatic Repeat Request. It is a data link layer protocol that uses a sliding window method. In this, if any frame is corrupted or lost, all subsequent frames have to be sent again.

The size of the sender window is N in this protocol. For example, Go-Back-8, the size of the sender window, will be 8. The receiver window size is always 1.

If the receiver receives a corrupted frame, it cancels it. The receiver does not accept a corrupted frame. When the timer expires, the sender sends the correct data.

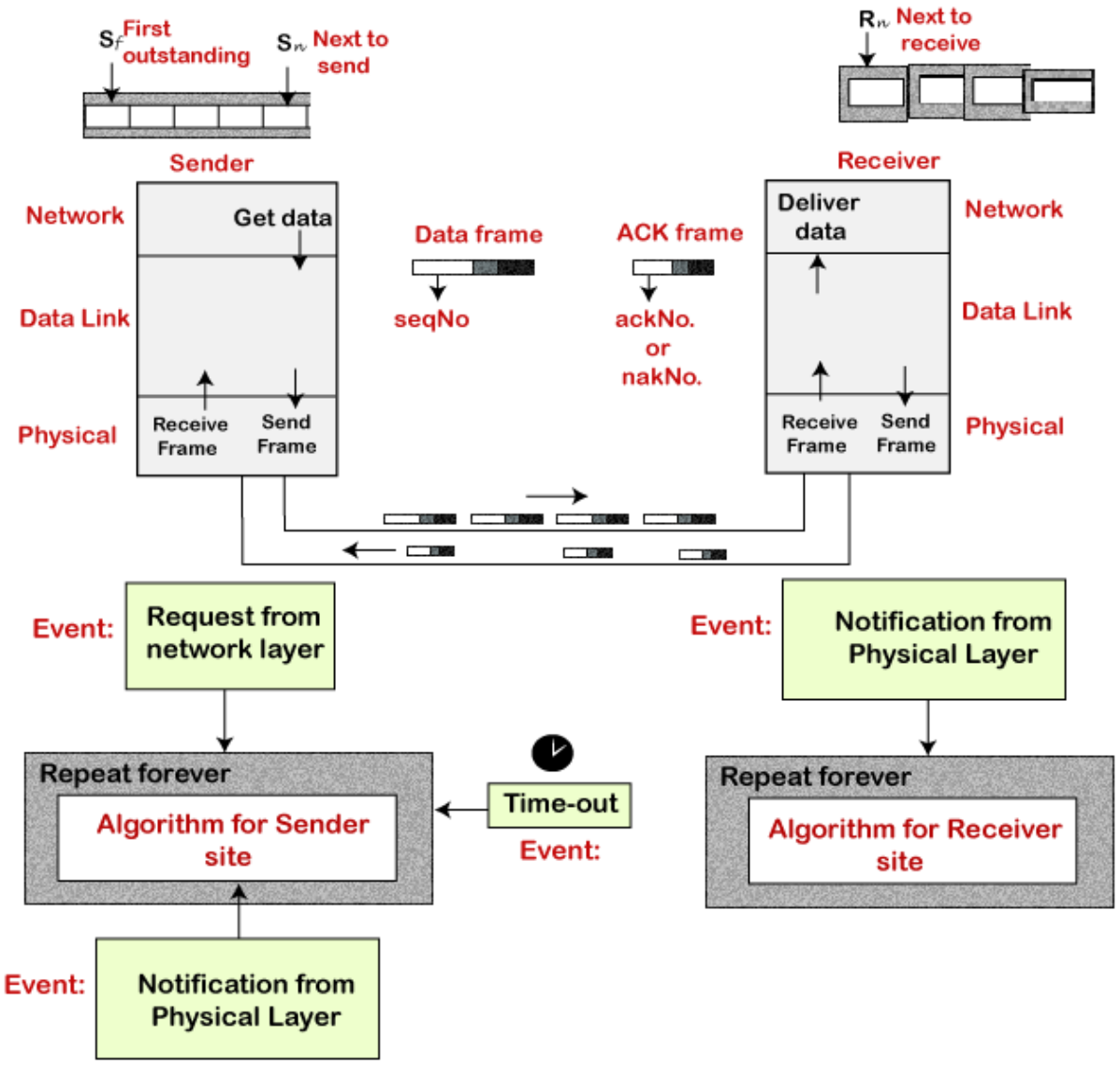

S_f First outstanding
S_n Next to send
R_n Next to receive
Sender
Receiver
Network
Get data
Data Link
Physical
Receive Frame
Send Frame
Data frame
seqNo
ACK frame
ackNo.
Deliver data
Network
Data Link
Physical
Receive Frame
Send Frame
Event:
Request from network layer
Repeat forever
Algorithm for Sender site
Time-out
Event:
Event:
Notification from Physical Layer
Event:
Notification from Physical Layer
Repeat forever
Algorithm for Receiver site

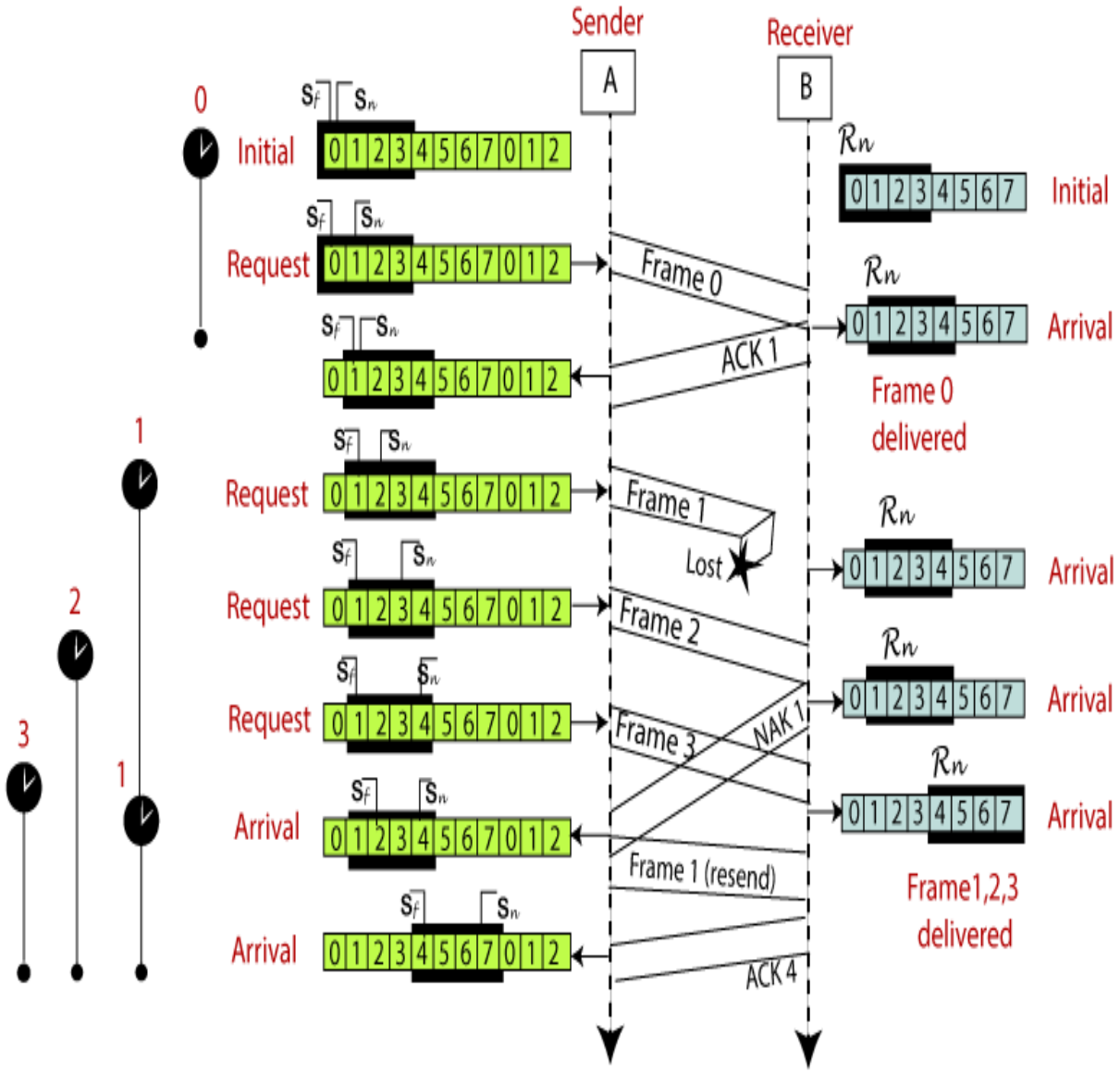

Sender
Receiver
A
B
0
Initial
Request
Frame 0
ACK 1
Frame 0
delivered
1
Request
Frame 1
Lost
2
Request
Frame 2
3
Request
Frame 3
NAK 1
1
Arrival
Frame 1 (resend)
Frame1,2,3
delivered
Arrival
ACK 4
Initial
Arrival
Arrival
Arrival
Arrival

Selective Repeat ARQ

Selective Repeat ARQ is also known as the Selective Repeat Automatic Repeat Request. It is a data link layer protocol that uses a sliding window method. The Go-back-N ARQ protocol works well if it has fewer errors. But if there is a lot of error in the frame, lots of bandwidth loss in sending the frames again. So, we use the Selective Repeat ARQ protocol. In this protocol, the size of the sender window is always equal to the size of the receiver window. The size of the sliding window is always greater than 1.

If the receiver receives a corrupt frame, it does not directly discard it. It sends a negative acknowledgment to the sender. The sender sends that frame again as soon as on the receiving negative acknowledgment. There is no waiting for any time-out to send that frame. The design of the Selective Repeat ARQ protocol is shown below.




Sender
Receiver
A
B
0
Initial
Request
Frame 0
ACK 1
Arrival
Frame 0 delivered
1
Request
Frame 1
Lost
Arrival
2
Request
Frame 2
Arrival
3
Request
Frame 3
NAK 1
Arrival
1
Arrival
Frame 1 (resend)
Arrival
Frame1,2,3 delivered
ACK 4

Code:-

```
import java.util.Scanner;

public class GoBackNSimulation {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the number of frames to be sent: ");
        int numFrames = scanner.nextInt();

        System.out.print("Enter the size of the sender's window: ");
        int senderWindowSize = scanner.nextInt();

        System.out.println("\nSimulating Go-Back-N Sliding Window Protocol...");

        Sender sender = new Sender(numFrames, senderWindowSize);
        Receiver receiver = new Receiver(numFrames);

        while (!sender.isTransmissionComplete() ||
!receiver.isReceivedAllFrames()) {
            sender.sendFrames(receiver);
            receiver.receiveFrames(sender);
        }

        System.out.println("\nTransmission complete. All frames have been sent
and acknowledged.");
    }
}

class Sender {
    private int[] frames;
    private int windowSize;
    private int nextFrameToSend;
    private int base;
    private boolean[] acknowledged;

    public Sender(int numFrames, int windowSize) {
        this.frames = new int[numFrames];
        for (int i = 0; i < numFrames; i++) {
            this.frames[i] = i + 1;
        }
        this.windowSize = windowSize;
        this.nextFrameToSend = 0;
        this.base = 0;
        this.acknowledged = new boolean[numFrames];
    }

    public int getWindowSize() {
        return windowSize;
    }
```

```
    public int getBase() {
        return base;
    }

    public int getNumFrames() {
        return frames.length;
    }

    public int getFrame(int index) {
        return frames[index];
    }

    public boolean isTransmissionComplete() {
        return nextFrameToSend == frames.length;
    }

    public void sendFrames(Receiver receiver) {
        System.out.println("\nSender transmitting frames " + base + " to " +
(base + windowSize - 1));
        for (int i = base; i < Math.min(base + windowSize, frames.length); i++)
{
            if (i < nextFrameToSend && !acknowledged[i]) {
                System.out.println("Sending Frame " + frames[i]);
            }
        }

        System.out.print("Enter the last acknowledged frame by the receiver: ");
        int lastAcknowledgedFrame = receiver.getLastAcknowledgedFrame();


        for (int i = base; i <= lastAcknowledgedFrame; i++) {
            acknowledged[i] = true;
        }


        while (base < frames.length && acknowledged[base]) {
            base++;
        }


        if (nextFrameToSend < frames.length && nextFrameToSend < base +
windowSize) {
            nextFrameToSend++;
        }
    }
}


class Receiver {
    private int[] receivedFrames;
    private int lastAcknowledgedFrame;

    public Receiver(int numFrames) {
```

```
        this.receivedFrames = new int[numFrames];
        this.lastAcknowledgedFrame = -1;
    }

    public boolean isReceivedAllFrames() {
        return lastAcknowledgedFrame == receivedFrames.length - 1;
    }

    public void receiveFrames(Sender sender) {
        System.out.println("\nReceiver waiting for frames...");

        for (int i = 0; i < sender.getWindowSize(); i++) {
            if (sender.getBase() + i < sender.getNumFrames() &&
receivedFrames[sender.getBase() + i] == 0) {

                int frame = sender.getFrame(sender.getBase() + i);
                System.out.println("Receiver received Frame " + frame);
                receivedFrames[frame - 1] = 1;  // Mark the frame as received
                lastAcknowledgedFrame = frame - 1;
            }
        }

        System.out.println("Sending acknowledgment for Frame " +
(lastAcknowledgedFrame + 1));
    }

    public int getLastAcknowledgedFrame() {
        return lastAcknowledgedFrame;
    }
}
```

Output:

```
PS C:\Users\Admin\Documents\AIDS-B1-75-Rushikesh-Tanpure> c
Enter the number of frames to be sent: 8
Enter the size of the sender's window: 4

Simulating Go-Back-N Sliding Window Protocol...

Sender transmitting frames 0 to 3
Enter the last acknowledged frame by the receiver:
Receiver waiting for frames...
Receiver received Frame 1
Receiver received Frame 2
Receiver received Frame 3
Receiver received Frame 4
Sending acknowledgment for Frame 4

Sender transmitting frames 0 to 3
Sending Frame 1
Enter the last acknowledged frame by the receiver:
Receiver waiting for frames...
Receiver received Frame 5
Receiver received Frame 6
Receiver received Frame 7
Receiver received Frame 8
Sending acknowledgment for Frame 8
```

```
Sender transmitting frames 4 to 7
Enter the last acknowledged frame by the receiver:
Receiver waiting for frames...
Sending acknowledgment for Frame 8

Sender transmitting frames 8 to 11
Enter the last acknowledged frame by the receiver:
Receiver waiting for frames...
Sending acknowledgment for Frame 8

Sender transmitting frames 8 to 11
Enter the last acknowledged frame by the receiver:
Receiver waiting for frames...
Sending acknowledgment for Frame 8

Sender transmitting frames 8 to 11
Enter the last acknowledged frame by the receiver:
Receiver waiting for frames...
Sending acknowledgment for Frame 8

Sender transmitting frames 8 to 11
Enter the last acknowledged frame by the receiver:
Receiver waiting for frames...
Sending acknowledgment for Frame 8

Sender transmitting frames 8 to 11
Enter the last acknowledged frame by the receiver:
Receiver waiting for frames...
Sending acknowledgment for Frame 8
```