Computer Networks Assignment 2: Implement three data link layer protocols, Stop and Wait, Go Back N Sliding Window and Selective Repeat Sliding Window for flow control

Name: Sayantan Biswas

Class: BCSE 3rd year 1st sem

Roll: 001910501057

Group: A2

Problem Statement: Sender, Receiver and Channel all are independent processes. There may be multiple Transmitter and Receiver processes, but only one Channel process. The channel process introduces random delay and/or bit error while transferring frames. Define your own frame format or you may use IEEE 802.3 Ethernet frame format.

Hints: Some points you may consider in your design.

Following functions may be required in Sender.

Send: This function, invoked every time slot at the sender, decides if the sender should (1) do nothing, (2) retransmit the previous data frame due to a timeout, or (3) send a new data frame. Also, you have to consider current network time measure in time slots.

Recv_Ack: This function is invoked whenever an ACK packet is received. Need to consider network time when the ACK was received, ack_num and timestamp are the sender's sequence number and timestamp that were echoed in the ACK. This function must call the timeout function.

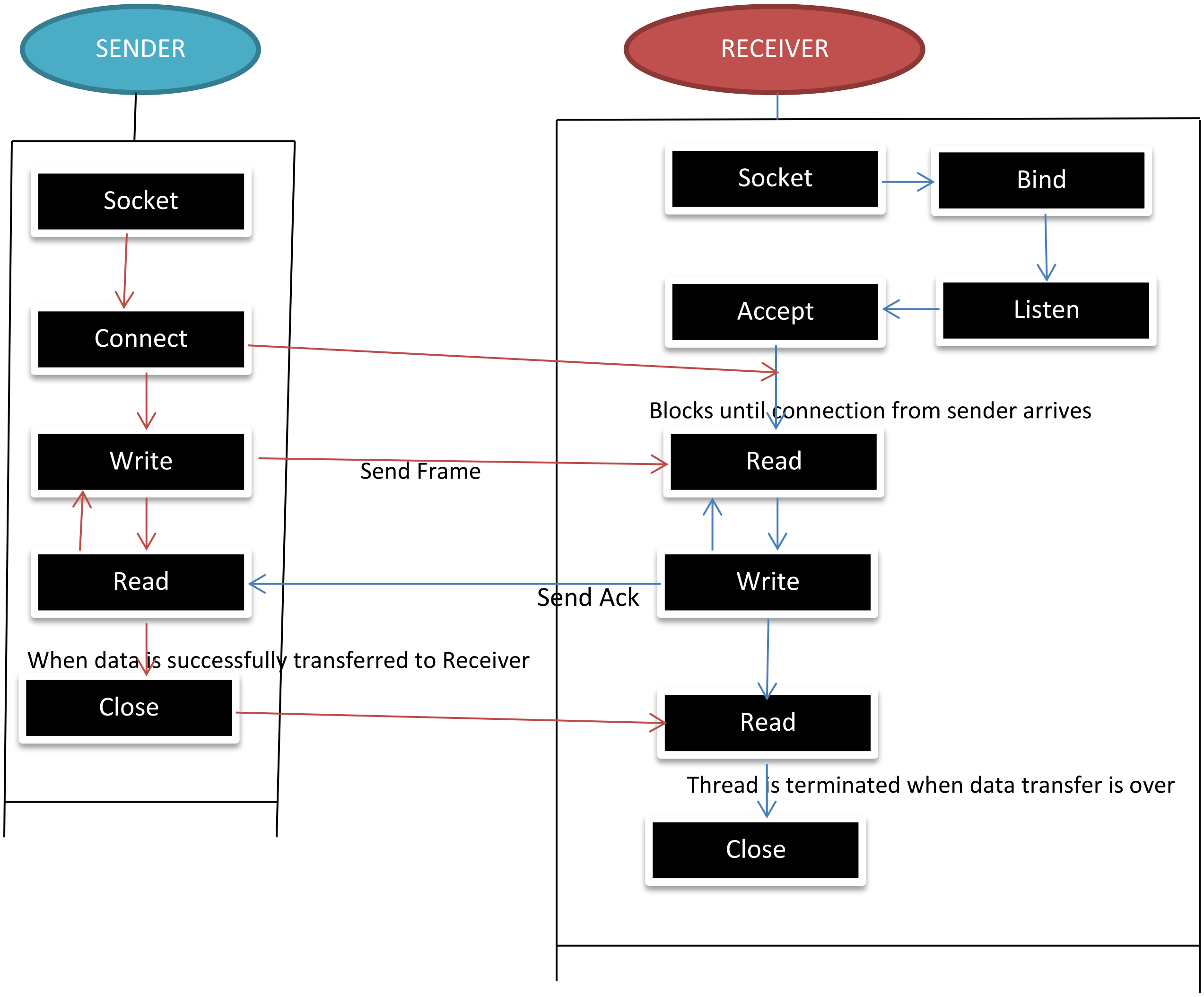
Timeout: This function should be called by ACK method to compute the most recent data packet's round-trip time and then re-compute the value of timeout.

Following functions may be required in Receiver.

Recv: This function at the receiver is invoked upon receiving a data frame from the sender.
Send_Ack: This function is required to build the ACK and transmit.
Sliding window:
The sliding window protocols (Go-Back-N and Selective Repeat) extend the stop-and-wait protocol by allowing the sender to have multiple frames outstanding (i.e., unacknowledged) at any given time. The maximum number of unacknowledged frames at the sender cannot exceed its "window size". Upon receiving a frame, the receiver sends an ACK for the frame's sequence number. The receiver then buffers the received frames and delivers them in sequence number order to the application.
Performance metrics: Receiver Throughput (packets per time slot), RTT, bandwidth-delay product, utilization percentage.

DESIGN: The code has been written in Java. There are two principal components for each of the three protocols:
(1) Sender
(2) Receiver
Input Data is given to the program through a text file to the sender. Sender calculates CRC of the data and constructs frames, which is transmitted to the receiver. While transmitting, data is corrupted by the transmission media, which is done in the sender’s side in this implementation. After reaching the receiver, the recipient system runs the error detection algorithm; if error is detected no acknowledgement is sent to the sender and if error is not found acknowledgements are sent to sender according to different protocols.
The implementation has been done with the help of ***socket programming*** and ***objected oriented programming of java.***

Input Format: A text file containing 1000 frames of binary digits as input (64bit.txt). It takes the CRC polynomial as input, in binary format. Errors are being injected randomly.
Output Format: Received frames, avg RTT time.


SENDER
RECEIVER
Socket
Connect
Write
Read
When data is successfully transferred to Receiver
Close
Socket
Bind
Accept
Listen
Blocks until connection from sender arrives
Send Frame
Read
Write
Send Ack
Read
Thread is terminated when data transfer is over
Close

Structure diagram of the program

IMPLEMENTATION:

For each protocols there are sender and receiver(two .java files).

(1) Sender.java

- **Static public String modify(String s):** To inject error in frames
- **Static public class CRC:**
  - Public String CRC(String s):constructor
  - Public String(char a, char b):performs xor between two characters which is used in CRC calculation
  - Public String get CRC Code(String data):Find CRC code of the data and return a data with appended CRC code.

(2) Receiver.java

- **Public class Receiver:** It creates a server socket and waits for senders to connect. When a sender connects, it starts a thread to communicate with the sender.
- **Class SenderHandler extends Thread:** It actually communicates with particular sender and sends acknowledgement.

Result:

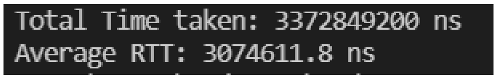
A test has been performed over 1000 frames each containing 64 bit data; total time taken and average RTT time is observed.
Frame format: DATA(64 bit) + CRC(7 bit) + Sequence Number(1 bit) = 72 bit(total)
CRC polynomial used: 10001001
Number of erroneous frames: 100
n = 3 (here 'n' is number of bits to represent window size)

PROTOCOL	Total time taken(ns)	Average RTT time(ns)
Stop and Wait	3372849200	3074611.80
Go Back N	465466700	464537.62
Selective Repeat	383028400	382263.88

OUTPUT:

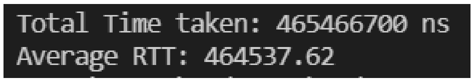
Output screenshots->

Stop and Wait:



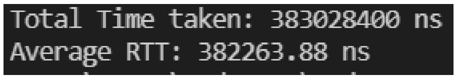
```
Total Time taken: 3372849200 ns
Average RTT: 3074611.8 ns
```

Go Back N:



```
Total Time taken: 465466700 ns
Average RTT: 464537.62
```

Selective Repeat:



```
Total Time taken: 383028400 ns
Average RTT: 382263.88 ns
```
