CS3205: Introduction to Computer Networks

Report: Assignment 4
Selective Repeat and Go-Back-N Protcols

Anirudh S
CS17B003

Problem Statement

The aim of this assignment is to implement two reliable packet transmission protocols: **Selective repeat** and **Go-Back-N**. Exact details of the assignment for each of these protocols are discussed in section 2

1 Introduction

1.1 Selective Repeat

Selective repeat is a sliding window protocol used for efficient and reliable packet transmission. In this protocol, the sender can send up to **N** packets before it must wait for an acknowledgement from the receiver. The receiver accepts and acknowledges any packet in the receiver window. If the first packet is acknowledged, the window slides and the new packet in the window is transmitted and so on. In case a packet is corrupted, a negative acknowledgement is sent back to the sender. Whenever retransmission occurs, due to timeout or negative acknowledgement, only the required packet is retransmitted. The protocol is summarised in figure 1.

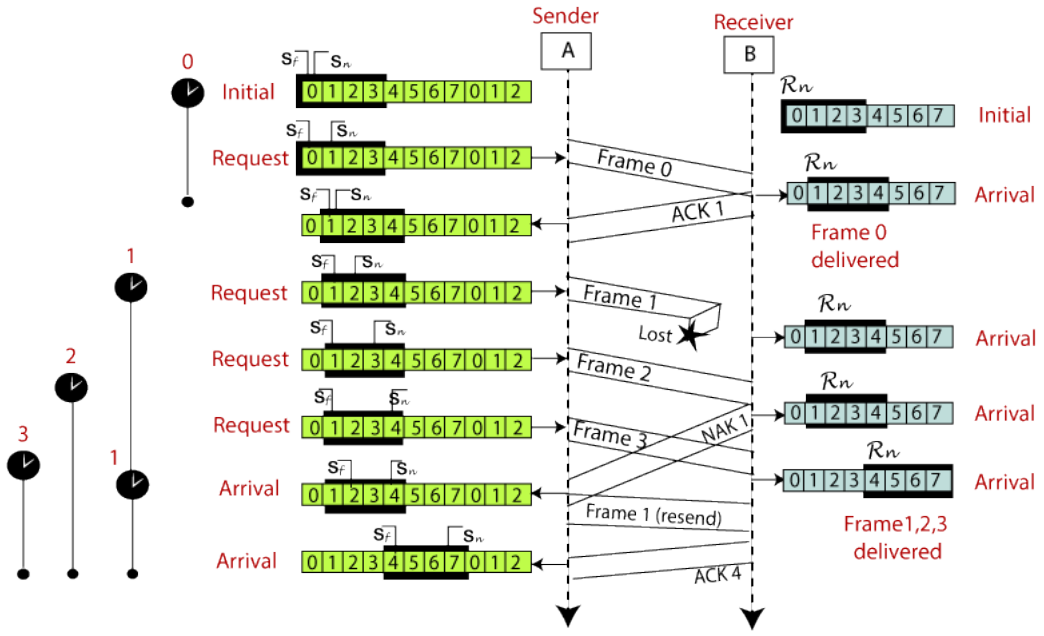


Figure 1: Selective repeat protocol[1]

1.2 Go-Back-N

Go-back-N is a sliding window protocol used for efficient and reliable packet transmission. In this protocol, the sender can send up to **N** packets before it must wait for an acknowledgement from the receiver. If the first packet is acknowledged, the window slides and the new packet in the window is transmitted and so on. The receiver accepts packets in-order only. In case a packet times out before it is acknowledged, all packets in the current window are retransmitted. The protocol is summarised in figure 2.

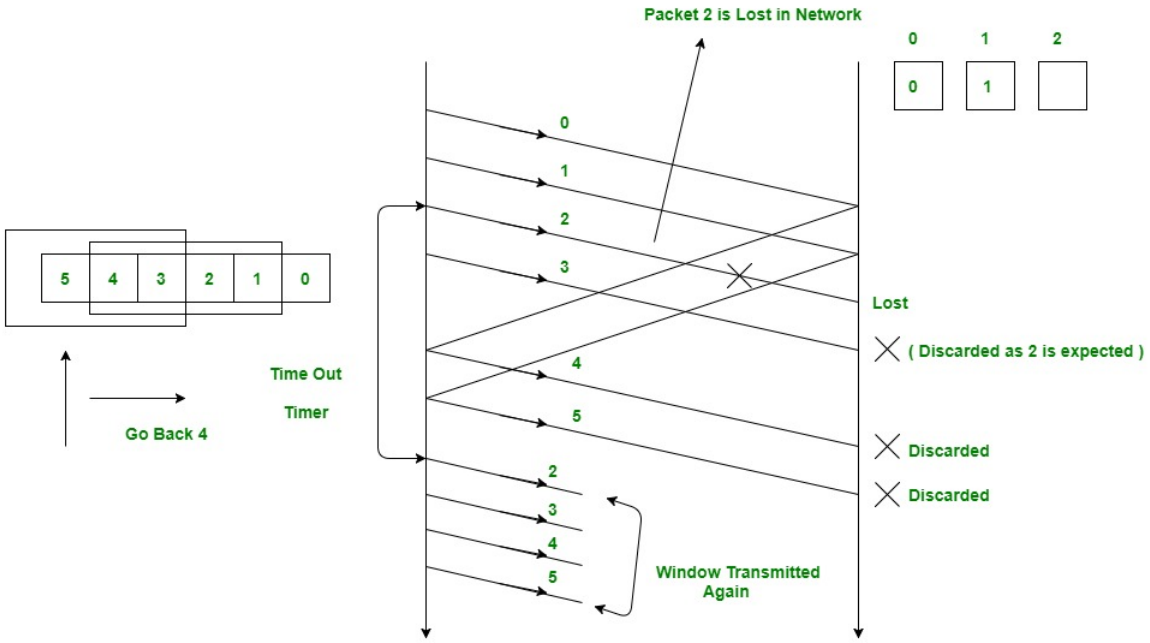


Figure 2: Go-back-N protocol[2]

2 Problem description

2.1 Selective repeat

2.1.1 Sender

- Sender takes the following command line arguments
  1. -d : Turn ON Debug Mode (OFF if flag not present)
  2. -s string : Receiver Name or IP address.
  3. -p integer : Receiver's Port Number
  4. -n integer : Sequence Number Field Length (in bits)
  5. -L integer : MAX_PACKET_LENGTH, in bytes
  6. -R integer : PACKET_GEN_RATE, in packets per second
  7. -N integer : MAX_PACKETS
  8. -W integer : WINDOW_SIZE
  9. -B integer : BUFFER_SIZE
- Sender generates packets of length Uniform(40, MAX_PACKET_LENGTH) at the rate specified by PACKET_GEN_RATE.

- Sender transmits packets based on window conditions. The first 10 packets have a timeout of 300ms and $10 * RTT_{avg}$ for the subsequent packets.
- Sender processes an ACK packet upon arrival by updating the local state variables and cancelling timers corresponding to acknowledged packets. RTT for the same is calculated and RTT_{avg} is updated.
- If a packet times out, only that packet is retransmitted by the sender.
- The sender terminates after MAX_PACKETS have been successfully acknowledged or if the maximum retransmission attempts for any sequence number exceeds 10.

2.1.2 Receiver

- Receiver takes the following command line arguments
  1. -d – Turn ON Debug Mode (OFF if flag not present)
  2. -p integer – Receiver's Port Number
  3. -N integer – MAX_PACKETS
  4. -n integer – Sequence Number Field Length (in bits)
  5. -W integer – WINDOW_SIZE
  6. -B integer – BUFFER_SIZE
  7. -e double – PACKET_ERROR_RATE
- Upon reception of a packet, the receiver randomly decides that packet is corrupted, depending on PACKET_ERROR_RATE, and decides to drop the packet.
- If the packet is decided as not corrupted, the receiver reads the packet and extracts the sequence number. If the receiver buffer is full, then the received packets are discarded even if they were correctly received. Otherwise, it follows the Selective Repeat protocol for generating ACKs, and buffering out-of-order packets.

2.2 Go-Back-N

2.2.1 Sender

- Sender takes the following command line arguments
  1. -d – Turn ON Debug Mode (OFF if flag not present)
  2. -s string – Receiver Name or IP address.
  3. -p integer – Receiver's Port Number
  4. -l integer – PACKET_LENGTH, in bytes
  5. -r integer – PACKET_GEN_RATE, in packets per second
  6. -n integer – MAX_PACKETS
  7. -w integer – WINDOW_SIZE
  8. -b integer – MAX_BUFFER_SIZE
- Sender generates packets of length PACKET_LENGTH at the rate specified by PACKET_GEN_RATE.
- Sender transmits packets based on window conditions. The first 10 packets have a timeout of 100ms and $10 * RTT_{avg}$ for the subsequent packets.

- Sender processes an ACK packet upon arrival by updating the local state variables and cancelling timers corresponding to acknowledged packets. RTT for the same is calculated and RTT_{avg} is updated.
- If a packet times out, all the packets in the current window are retransmitted by the sender.
- The sender terminates after MAX_PACKETS have been successfully acknowledged or if the maximum retransmission attempts for any sequence number exceeds 5.

2.2.2 Receiver

- Receiver takes the following command line arguments
  1. -d – Turn ON Debug Mode (OFF if flag not present)
  2. -p integer – Receiver's Port Number
  3. -n integer – MAX_PACKETS
  4. -e double – Packet Error Rate (RANDOM_DROP_PROB)
- Upon reception of a packet, the receiver randomly decides that packet is corrupted, depending on RANDOM_DROP_PROB, and decides to drop the packet.
- If the packet is decided as not corrupted, the receiver reads the packet and extracts the sequence number. If sequence number matches the NEXT EXPECTED sequence number, it transmits an ACK to the sender, and updates local state variables.

3 Experiments

3.1 Selective Repeat

Fixed parameters:

- IP address: 127.0.0.1
- Port: 12345
- Sequence number field length: 5
- MAX_PACKETS: 100
- WINDOW_SIZE: 4
- BUFFER_SIZE (Sender): 20
- BUFFER_SIZE (Receiver): 25

Variable parameters:

- PACKET_GEN_RATE (PGR) $\in \{20, 300\}$
- PACKET_LENGTH (PL) $\in \{256, 1500\}$
- RANDOM_DROP_PROB (RDP) $\in \{0.001, 0.01, 0.1\}$

3.1.1 Results

PL \ RDP	PGR = 20			PGR = 300		
	0.001	0.01	0.1	0.001	0.01	0.1
256	0.404	0.459	44.617	0.537	1.185	5.391
1500	0.408	0.497	7.367	0.4724	0.4542	4.618

Table 1: Average RTT

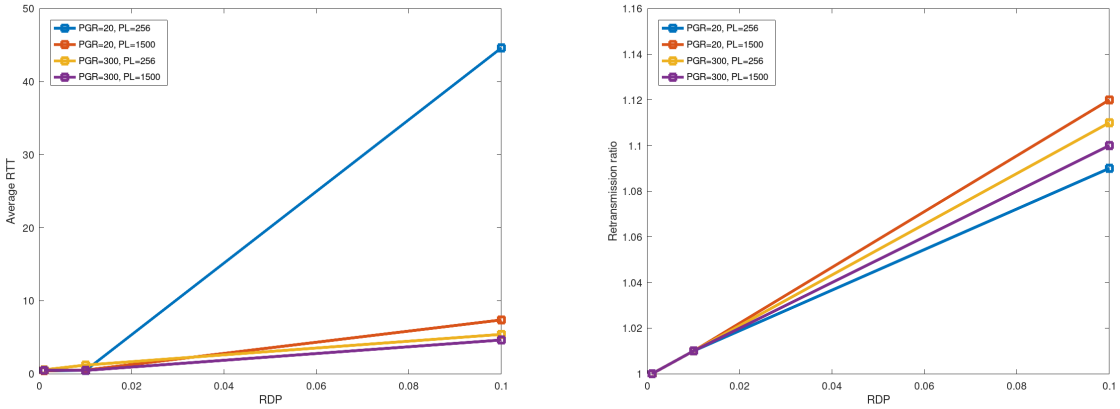
PL \ RDP	PGR = 20			PGR = 300		
	0.001	0.01	0.1	0.001	0.01	0.1
256	1.0	1.01	1.09	1.0	1.01	1.11
1500	1.0	1.01	1.12	1.0	1.01	1.10

Table 2: Retransmission ratio

PL \ RDP	PGR = 20			PGR = 300		
	0.001	0.01	0.1	0.001	0.01	0.1
256	163.63	141.99	146.75	153.98	137.26	154.20
1500	795.06	779.52	768.78	801.29	813.07	702.91

Table 3: Average packet length

3.1.2 Plots



(a) Average RTT (b) Retransmission ratio

Figure 3: Plots for Selective Repeat

3.2 Go-Back-N

Fixed parameters:

- IP address: 127.0.0.1
- Port: 12345
- MAX_PACKETS: 100
- WINDOW_SIZE: 4
- BUFFER_SIZE: 20

Variable parameters:

- PACKET_GEN_RATE (PGR) $\in \{20, 300\}$
- PACKET_LENGTH (PL) $\in \{256, 1500\}$
- RANDOM_DROP_PROB (RDP) $\in \{0.001, 0.01, 0.05\}$

3.2.1 Results

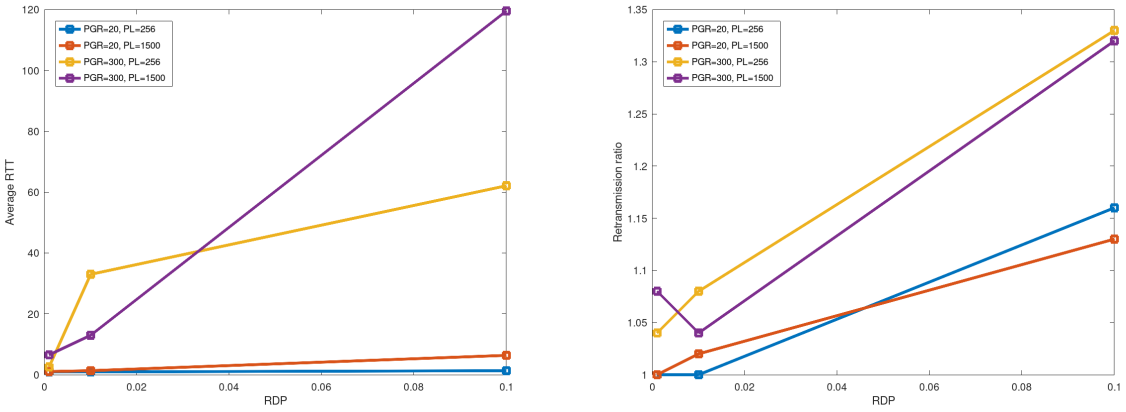
	PGR = 20			PGR = 300		
RDP / PL	0.001	0.01	0.05	0.001	0.01	0.05
256	0.917	0.925	11.317	2.556	32.995	62.106
1500	1.037	1.332	6.366	6.522	12.964	119.542

Table 4: Average RTT

	PGR = 20			PGR = 300		
RDP / PL	0.001	0.01	0.05	0.001	0.01	0.05
256	1.0	1.0	1.16	1.04	1.08	1.33
1500	1.0	1.02	1.13	1.08	1.04	1.32

Table 5: Retransmission ratio

3.2.2 Plots



(a) Average RTT (b) Retransmission ratio

Figure 4: Plots for Go-Back-N

3.2.3 Observations

- As drop probability increases, both retransmission ratio and average RTT increase as expected. This is because more timeouts occur for higher drop probabilities.
- The values for Go-Back-N are higher as compared to the corresponding values for Selective Repeat as expected. This is because in case of a timeout, the entire window is retransmitted as opposed to only the required packet in Selective Repeat.
- The packet length does not seem to affect the values.

4 Conclusion

This assignment was very helpful in understanding the Selective Repeat and Go-Back-N protocols thoroughly. It introduced us to the challenges that a person faces while implementing the protocol in a real setting. The assignment also proved to be an excellent exercise in concurrent programming. There were many concurrency and synchronizations issues to be handled and it was often very hard and time-consuming to locate and rectify the bugs. My solution to this assignment still has scope for improvement. Negative acknowledgements for Selective Repeat are not implemented in the current version and implementing them might lead to better results. Second, cumulative acknowledgements could have been implemented in Go-Back-N. Finally, concurrency issues could have been handled better but were left unrefined due to shortage of time. All in all, this assignment provided a good mix of theoretical understanding as well as practical implementation experience.

References

[1] Sliding Window Protocol URL: https://www.javatpoint.com/sliding-window-protocol

[2] Sliding Window Protocol — Set 2 (Receiver Side)
URL: https://www.geeksforgeeks.org/sliding-window-protocol-set-2-receiver-side/