Sebastian Sotomayor

EEL4781

Programming Project Report

Chosen Protocol

For this project, I chose to simulate the Go-Back-N protocol using Java. Go-Back-N is a sliding window protocol that lets the sender transmit multiple packets before needing an acknowledgment. Although I initially considered Python, I ultimately went with Java due to my greater familiarity with it.

Implementation Approach

The simulation includes four core components: Packet, Sender, Receiver, and Channel. Each packet contains a sequence number, data, and ACK status. The channel manages transmission between sender and receiver and introduces packet loss (manual and probabilistic). The sender and receiver classes handle sending, receiving, and responding to packets and acknowledgments.

Timers, Sequence Numbers, and Buffering

Packets are assigned sequence numbers by the sender. The receiver tracks its expected sequence number, accepting only in-order packets and ignoring others. If an out-of-order packet arrives, the receiver resends the previous ACK. When an in-order packet is received, it sends an ACK for that packet, letting the sender move its window forward.

The sender sets a timer for the oldest unacknowledged packet. If no ACK is received before it expires, the sender retransmits the current window and resets the timer. Packet loss probability is set globally at 20%, and propagation delay at 1 second.

Buffering occurs at both ends. The sender stores all packets up front for easy retransmission. The receiver only buffers in-order packets—out-of-order ones are ignored until the missing packet arrives.

Observations Under Different Network Conditions

I tested the simulation using different window sizes and manual packet/ACK drops. Smaller windows slowed down transmission noticeably. With a window size of one, the protocol acted like Stop-and-Wait. Larger windows improved performance even with random losses.

Manual drops were handled using a HashSet, allowing the user to specify which packets or ACKs would be lost during their first transmission. These losses triggered retransmissions, demonstrating the protocol's reliability mechanism.

Protocol Behavior in the Simulation

At the start, the simulation displays the message being sent and total packet count. It then prompts the user for window size and which packets/ACKs to drop. The image below shows this in action:

```
Hello World! This is my simulation for the Go-Back-N Protocol.
There are 10 total packages in this simulation.
Enter window size: 4
Enter the number of packets you would like to drop: 2
Enter the sequence number (1 - 10) of the packet you would like to drop: 1
Enter the sequence number (1 - 10) of the packet you would like to drop: 2
Enter the number of acks you would like to drop: 3
Enter the sequence number (1 - 10) of the ack you would like to drop: 4
Enter the sequence number (1 - 10) of the ack you would like to drop: 5
Enter the sequence number (1 - 10) of the ack you would like to drop: 6
```

During the simulation, you'll see when packets are sent, received (in or out of order), acknowledged, dropped, or retransmitted due to timeouts. This sequence (shown below) showcases the Go-Back-N protocol in action:

```
Total packets to send: 10
Sender sending: PKT1
Sender sending: PKT2
Sender sending: PKT3
Sender sending: PKT4
Channel (manually) dropped: PKT1
Channel (manually) dropped: PKT2
Channel lost: PKT3
Channel lost: PKT4
Sender timeout. Retransmitting window...
Receiver got: PKT1
Receiver sending: ACK1
Receiver got: PKT2
Receiver sending: ACK2
Channel lost: PKT3
Receiver got out-of-order: PKT4
Receiver sending: ACK1
Sender got: ACK1
Sender sending: PKT5
Sender got: ACK2
Sender sending: PKT6
Receiver got out-of-order: PKT5
```

Once the packet is fully transmitted, it will display the data obtained by the receiver and showcase the stats regarding the transmission. An image of this is shown below:

```
Final Received Message: Hello World! This is my simulation for the Go-Back-N Protocol.

Stats for simulation:
Total Packages Sent: 48
Total Packages Received: 35
Number of Retransmission: 28
Total Packages Lost: 13

Thank you!!!
```

Conclusion

This project helped me strengthen both my understanding of the Go-Back-N protocol and my skills in Java programming. By building the simulation from scratch, I gained a clearer view of how reliable data transmission works over an unreliable network, including the role of timers, acknowledgments, and retransmissions. It also gave me valuable experience in structuring Java classes, working with threads, and handling real-time behavior through scheduling and delays. Overall, it was a rewarding way to apply both networking concepts and Java coding techniques in a practical, hands-on way.