

Simulation of Go-Back-N Protocol

Aarya Gupta

Adarsh Jha

CSE 232 Section B - Computer Networks

November 10, 2024

Abstract

This report details the implementation of the Go-Back-N ARQ protocol using Python and UDP sockets. The simulation involves data link layer protocol design, packet generation, encapsulation, acknowledgment, and retransmission mechanisms to simulate realistic network behavior. The protocol is tested under various conditions, including packet drop probability and network delay, and analyzed for performance metrics such as average delay and retransmission count.

1 Introduction

The Go-Back-N protocol is a widely-used automatic repeat request (ARQ) protocol for error control in data link layer communication. It supports reliable data transmission over an unreliable network by retransmitting packets upon detection of loss or corruption. This project implements Go-Back-N in a client-server model, where two machines simulate data link entities using UDP sockets for packet transfer. The objectives include:

- Understanding data link protocols through Go-Back-N.
- Simulating bidirectional packet transmission with acknowledgment.
- Analyzing protocol performance under varying network conditions.

2 Overview of Go-Back-N Protocol

The Go-Back-N protocol uses a sliding window mechanism where the sender can transmit up to a specified number of frames (window size) before needing

an acknowledgment. If a frame is lost, the protocol retransmits all subsequent frames in the window, allowing for efficient error correction. Key elements of the Go-Back-N protocol include:

- **Sliding Window:** The sender's window allows up to $N = 8$ frames, with a window size of 7, ensuring controlled transmission.
- **Acknowledgment and Retransmission:** The receiver acknowledges frames in order, and any unacknowledged frame causes retransmission of all subsequent frames in the sender's window.
- **Sequence Numbering:** Modulo-8 numbering is used for frame sequence numbers, ensuring continuity.

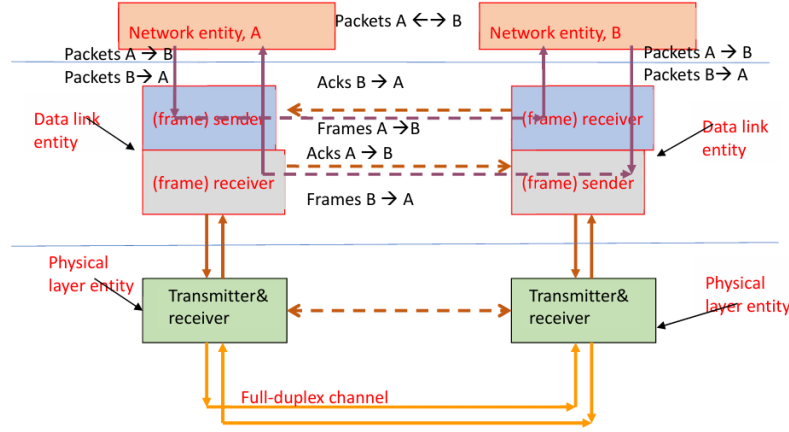


Figure 1: Workflow of Go-Back-N Protocol Implementation

3 Implementation Details

3.1 System Design and Multi-Machine Concept

The system design is based on a client-server model where:

- **Machine 1 (Client)** hosts DL_Entity_1, acting as the sender and receiver.
- **Machine 2 (Server)** hosts DL_Entity_2, also acting as both sender and receiver.

- Each machine creates UDP sockets to simulate full-duplex communication, enabling bidirectional data transfer between data link entities.

Each machine's data link entity encapsulates packets, maintains a transmission window, and manages acknowledgment and retransmission in accordance with the Go-Back-N protocol.

3.2 Packet Generation and Frame Encapsulation

The 'Network_Entity' on each machine generates packets at random intervals between $T1$ and $T2$, which are encapsulated as frames with sequence and acknowledgment numbers. These frames are added to an outgoing queue for transmission.

3.3 Error Simulation and Network Delay

To simulate network conditions:

- **Frame Drop Probability (P):** A probability P is assigned for random frame drops during transmission.
- **Network Delay (T3, T4):** A random delay is introduced at both transmitter and receiver ends within the range $[T3, T4]$, simulating propagation and queuing delays.

3.4 Multithreading and Concurrent Tasks

Multithreading is utilized to manage various tasks:

- Packet generation, sending, receiving, and timeout handling operate in separate threads.
- This allows concurrent handling of transmission and retransmission, improving efficiency.

```
53096 Simulation Completed.
53097 Total Frames Sent: 10707
53098 Total Retransmissions: 707
53099 Average Delay per Packet: 0.1543 seconds
53100 Average Number of Times a Frame was Sent: 1.07
53101
```

Figure 2: Output Example from Machine 1 (DL_Entity_1)

```

89235 Simulation Completed.
89236 Total Frames Sent: 19463
89237 Total Retransmissions: 9463
89238 Average Delay per Packet: 0.3450 seconds
89239 Average Number of Times a Frame was Sent: 1.95
89240

```

Figure 3: Output Example from Machine 2 (DL_Entity_2)

4 Experimental Setup and Results

4.1 Parameters and Simulation Cases

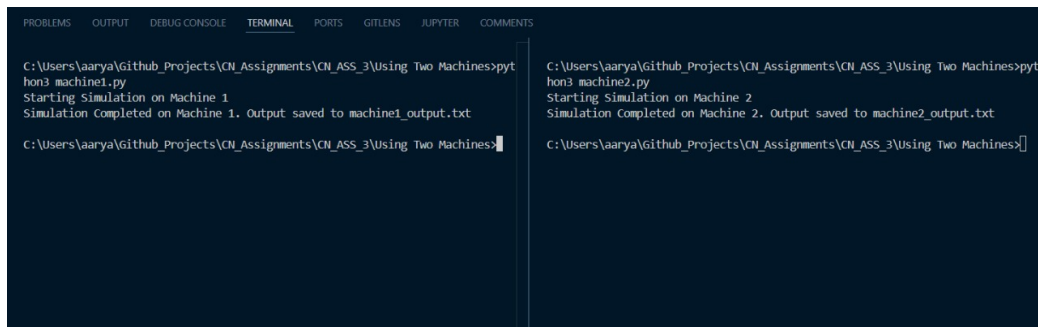
The simulation involves transferring 10,000 packets in each direction under the following conditions:

- Two different values of drop probability, P , to observe the impact on retransmissions.
- Two distinct pairs of network delay ranges, $T3$ and $T4$, to study average delay effects.

4.2 Performance Metrics

The simulation records and analyzes:

- **Average Delay:** Measured from the time of initial transmission to successful reception.
- **Retransmission Count:** The average number of times each frame is sent.



```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GIT LENS JUPYTER COMMENTS
C:\Users\aaarya\Github_Projects\CN_Assignments\CN_ASS_3\Using Two Machines>pyth
hon3 machine1.py
Starting Simulation on Machine 1
Simulation Completed on Machine 1. Output saved to machine1_output.txt
C:\Users\aaarya\Github_Projects\CN_Assignments\CN_ASS_3\Using Two Machines>
C:\Users\aaarya\Github_Projects\CN_Assignments\CN_ASS_3\Using Two Machines>pyth
hon3 machine2.py
Starting Simulation on Machine 2
Simulation Completed on Machine 2. Output saved to machine2_output.txt
C:\Users\aaarya\Github_Projects\CN_Assignments\CN_ASS_3\Using Two Machines>

```

Figure 4: Terminal Output Showing Simulation Metrics

5 Conclusion

The Go-Back-N protocol was successfully simulated, demonstrating key principles of data link layer communication. The analysis of average delay and retransmissions under varying network conditions highlights the effectiveness of Go-Back-N in handling errors. This project reinforces understanding of error correction and flow control, and future work could extend this simulation to other ARQ protocols or more complex network setups.