Homework 3: Data Transfer Protocol

Cedric Bone

1 Introduction

This report documents a reliable data transfer protocol. The protocol has a sliding window, error detection, packet loss recovery, and handling of out-of-order packets.

2 Protocol

The protocol has the following components:

- **Sender:** Sliding window protocol with timeout-based retransmission
- **Receiver:** Buffering of out-of-order packets and verifies packet integrity
- **Network Simulator:** Simulates an unreliable channel
- **Packet:** Defines the data structure

To use the protocol you can open 3 terminals, and start the network simulator with "python network_simulator.py", start main in receive mode with an output file with "python main.py receive output_file.txt" and start main in send mode with a file you want to send with "python main.py send input_file.txt".

Figure 1: Network simulator showing packet forwarding, loss, and corruption. The simulator is an intermediary between sender and receiver, simulating network conditions.

2.1 Packet Corruption

The protocol uses a sliding window with cumulative acknowledgments and uses 16-bit checksums for error detection. When packets become corrupted during transmission, the protocol detects the corruption using checksums and triggers retransmission. As shown in Figures 1, 2, and 3, when a packet is corrupted, the receiver detects the checksum mismatch and requests retransmission by sending an ACK for the last correctly received packet.

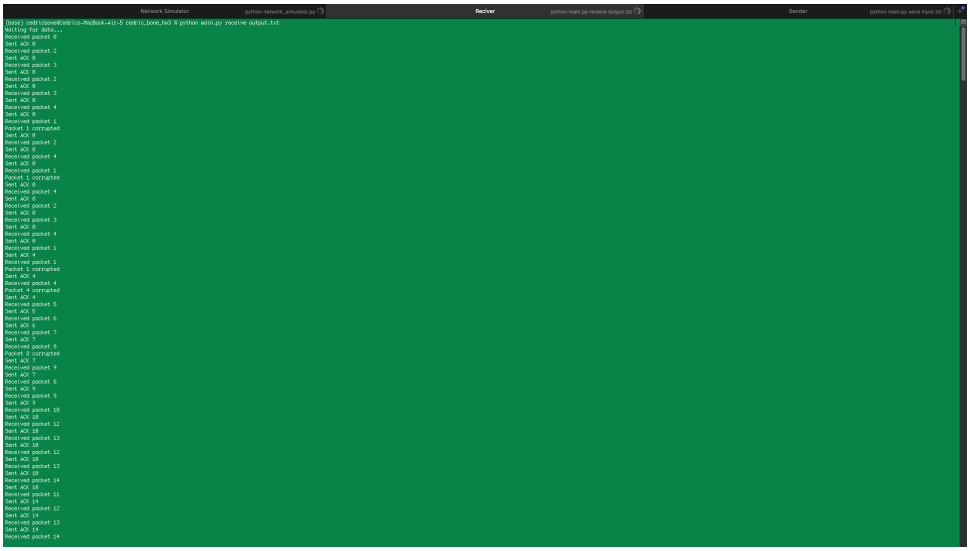

Figure 2: Receiver terminal output showing packet reception and acknowledgment. The receiver handles out-of-order packets and verifies packet integrity with checksums.

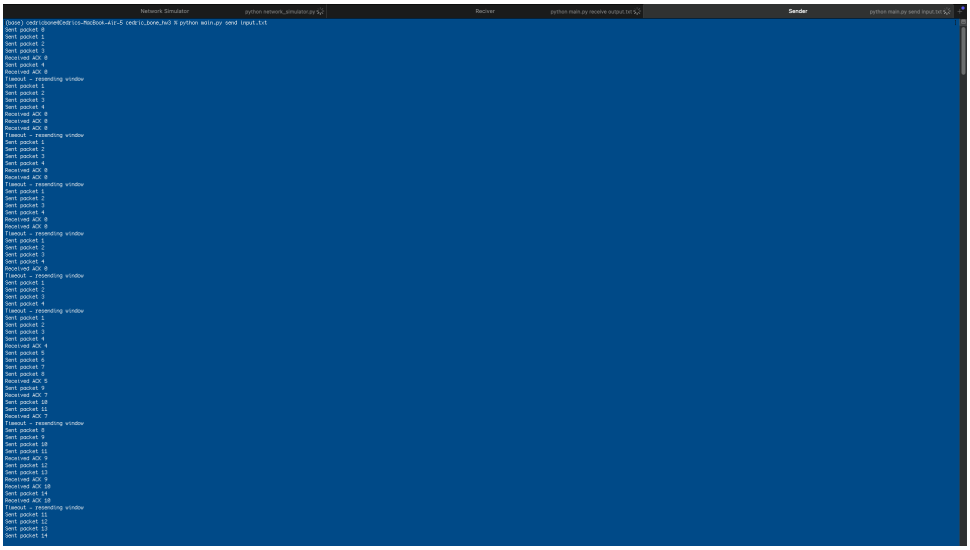

Figure 3: Sender output showing the sliding window with packet transmission and acknowledgment. The sender handles timeout retransmissions.

2.2 Packet Loss

The protocol uses timeout to detect and recover from packet loss. When acknowledgments are not received within the timeout period, packets are automatically retransmitted. Figures 1, 2, and 3 show how the sender detects packet loss through timeout and resends all unacknowledged packets in the current window.

2.3 Packet Reordering

The protocol can handle packets arriving out of order by buffering them at the receiver until the missing packets arrive. As shown in Figures 1, 2, and 3, when packets arrive out of order, the receiver buffers them and processes them in the correct sequence once the missing packets are received.

3 Example

The protocol successfully transferred a file from sender to receiver through the unreliable network simulator. Figure 4 shows a complete file transfer. The end shows the original input file (the start of a literature review I did for a class) and the output file matches the original file.

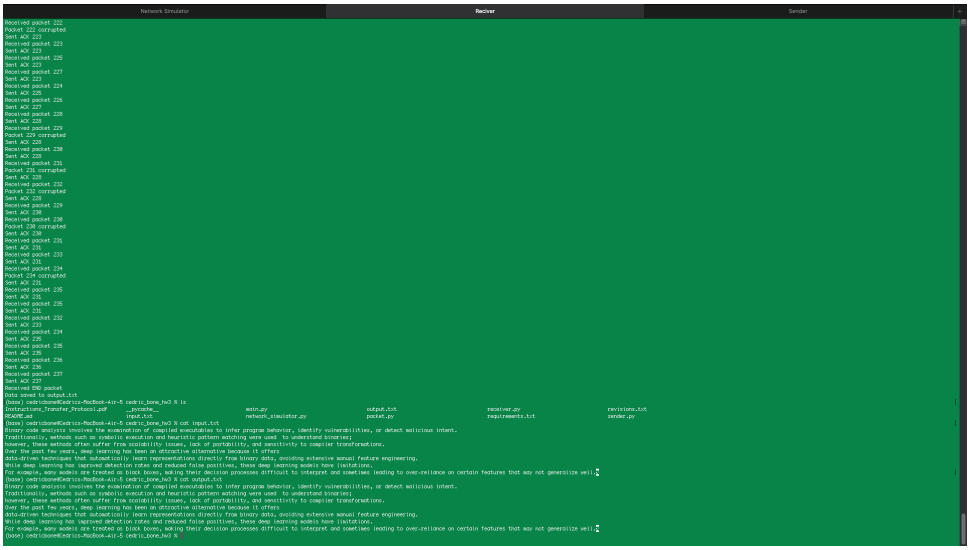

Figure 4: Successful file transfer through the protocol

4 Conclusion

The implemented reliable data transfer protocol successfully provides data integrity over an unreliable channel. The protocol shows the ability to deal with packet loss, corruption, and reordering.