MP2 Report

Overview

This project implements a connection-oriented, reliable, pipelined transport protocol similar to TCP. The protocol supports reliable data transfer between a sender and receiver using **flow control**, **congestion control**, **packet sequencing**, and **error handling**.

Specification of protocol

The protocol includes key features of TCP, such as **reliability**, **flow control**, **congestion control**, and **connection management**. We use **UDP sockets** to build these features from scratch, as UDP is a lightweight, connectionless protocol.

Connection-oriented

- Three-Way Handshake: The exchange of `SYN`, `SYN-ACK`, and `ACK` in the sender (`Sender.py`) and receiver (`Reveive.py`) ensures a connection is established before data transfer begins.

Reliable

- ACKs and Sequence ID: For every packet received, teh receievr sends an `ACK:<>seqence_number`. This confirms the succesful receipt pf the package.
- Timeout: If the sender doesn't receive an ACK within a specified timeout (`TIMEOUT_INTERVAL`), it will retransmits the unacknowledaged packet.

Pipline (Flow / Congestion Control):

- Using Go-back-N: We have a dynamic window size in this case. We use `cwnd`, `sshthresh` to control our window size.
- Receiver Advertised Window: The receiver advertises its available buffer space, and the sender adjusts the number of packets accordingly.
- Using Congestion Window `cwnd`: Starts with a value of 1 and grows exponetially during the slow start phase.
- Using `sshthresh`: Reduece the congestion window size upon detecting a timeout, resetting to a slower start phase for better network adaptability.

Performance:

- The sender leverages multi-threading for packet transmission to parallelize operations.

- Our sender will wait for a specified period to receive all potential ACKs after transmitting a batch of packages.

Connection Teardown

- The use of `FIN` and `FIN-ACK` ensures that the connection is closed, indicating that all data has been delivered reliably.

Test procedures

Simulation of loss and errors

- Set `LOSS_PROBABILITY = 0.2`: We use random function to drop packets randomly with 20% probability.

Packets Set Up:

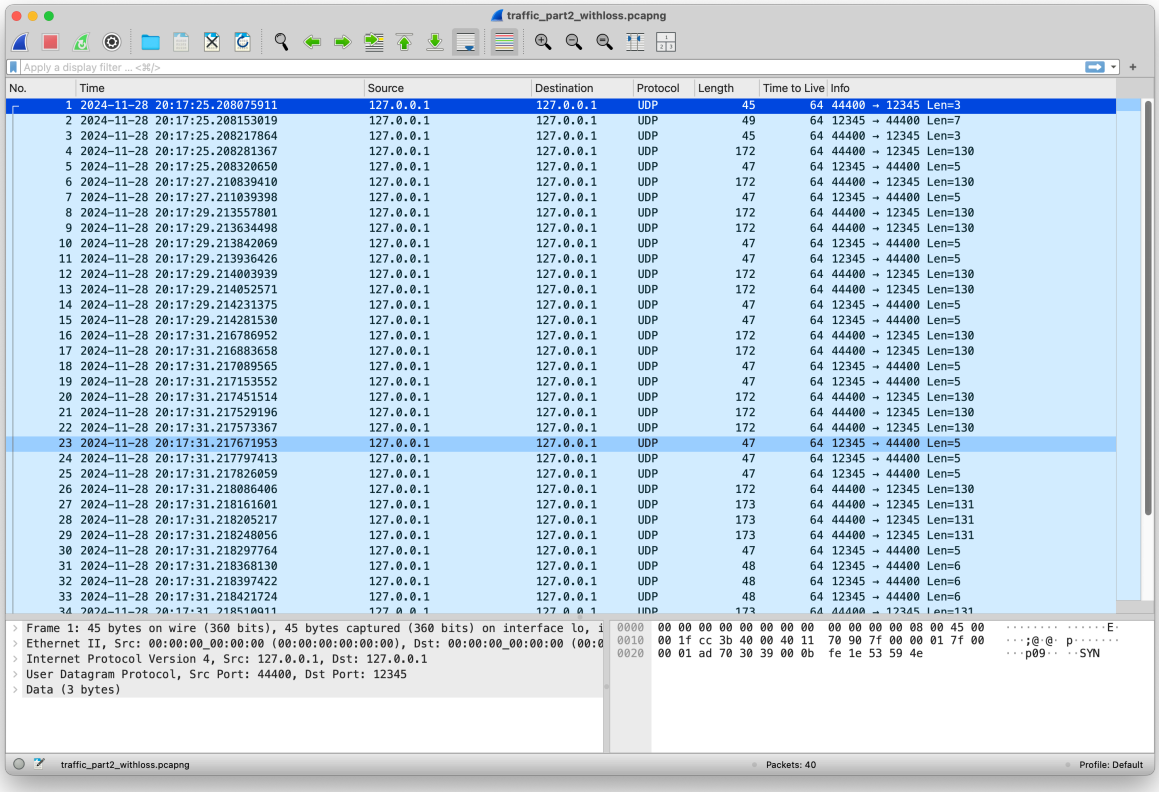
- Prepare a long string message.
- Divide the message into 14 packages. Each packet has a length of 128 bytes.

Step:

- Start receiver first `python Receive.py`
- Then Start the Sender `python Sender.py`

Explaination of results

Traffic Capture:

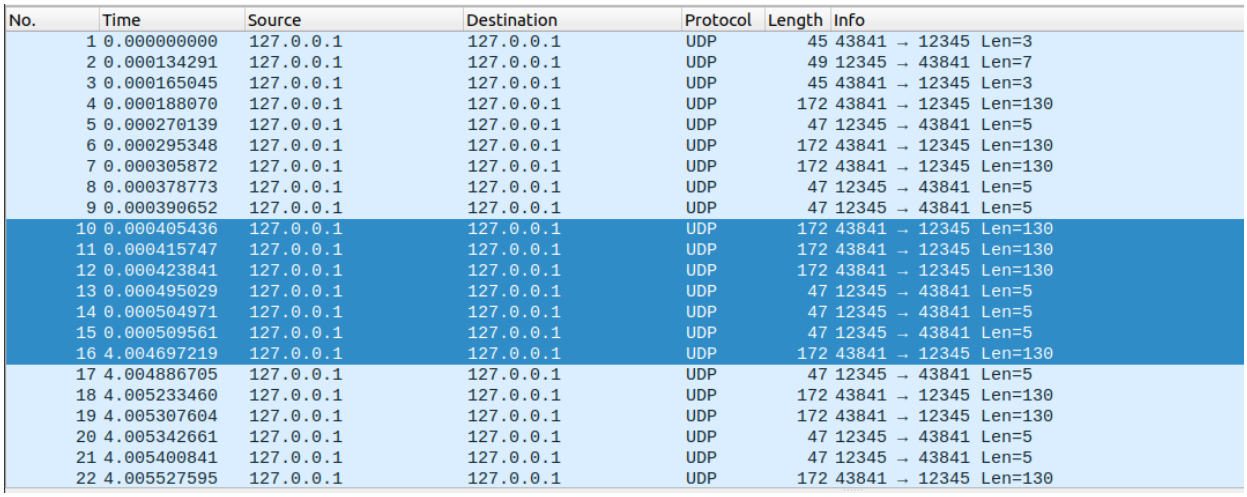


The first three packets are used for the Three-Way Handshake.

After completing the Three-Way Handshake, the sender begins transmitting the first message.

If no errors or losses occur, the Congestion Window (`cwnd`) will increase with each iteration (e.g., 20–22, 26–29).

Error Handling



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	127.0.0.1	127.0.0.1	UDP	45	43841 → 12345 Len=3
2	0.000134291	127.0.0.1	127.0.0.1	UDP	49	12345 → 43841 Len=7
3	0.000165045	127.0.0.1	127.0.0.1	UDP	45	43841 → 12345 Len=3
4	0.000188070	127.0.0.1	127.0.0.1	UDP	172	43841 → 12345 Len=130
5	0.000270139	127.0.0.1	127.0.0.1	UDP	47	12345 → 43841 Len=5
6	0.000295348	127.0.0.1	127.0.0.1	UDP	172	43841 → 12345 Len=130
7	0.000305872	127.0.0.1	127.0.0.1	UDP	172	43841 → 12345 Len=130
8	0.000378773	127.0.0.1	127.0.0.1	UDP	47	12345 → 43841 Len=5
9	0.000390652	127.0.0.1	127.0.0.1	UDP	47	12345 → 43841 Len=5
10	0.000405436	127.0.0.1	127.0.0.1	UDP	172	43841 → 12345 Len=130
11	0.000415747	127.0.0.1	127.0.0.1	UDP	172	43841 → 12345 Len=130
12	0.000423841	127.0.0.1	127.0.0.1	UDP	172	43841 → 12345 Len=130
13	0.000495029	127.0.0.1	127.0.0.1	UDP	47	12345 → 43841 Len=5
14	0.000504971	127.0.0.1	127.0.0.1	UDP	47	12345 → 43841 Len=5
15	0.000509561	127.0.0.1	127.0.0.1	UDP	47	12345 → 43841 Len=5
16	4.004697219	127.0.0.1	127.0.0.1	UDP	172	43841 → 12345 Len=130
17	4.004886705	127.0.0.1	127.0.0.1	UDP	47	12345 → 43841 Len=5
18	4.005233460	127.0.0.1	127.0.0.1	UDP	172	43841 → 12345 Len=130
19	4.005307604	127.0.0.1	127.0.0.1	UDP	172	43841 → 12345 Len=130
20	4.005342661	127.0.0.1	127.0.0.1	UDP	47	12345 → 43841 Len=5
21	4.005400841	127.0.0.1	127.0.0.1	UDP	47	12345 → 43841 Len=5
22	4.005527595	127.0.0.1	127.0.0.1	UDP	172	43841 → 12345 Len=130

In this case (e.g., 10–16), the receiver expected to receive 4 packets, but only 3 packets were sent. As a result, the Congestion Window (`cwnd`) is reset to 1, and the sender will retransmit the missing packet based on the ACKs received.

Sender output:

```
Sender: Setting up connection. Sending SYN
Sender: Connection established. Send ACK
Sender: Sent packet: 0

Sender: Received ACK: 0

Sender: Sent packet: 1
Sender: Sent packet: 2

Sender: Received ACK: 1
Sender: Received ACK: 2

Sender: Sent packet: 3
Sender: Packet 4 lost
Sender: Sent packet: 5
Sender: Sent packet: 6

Sender: Received ACK: 3
Sender: Received ACK: 3
Sender: Received ACK: 3
Sender: Timeout occurred, retransmitting...

```

```
Sender: Packet 4 lost

Sender: Timeout occurred, retransmitting...

Sender: Sent packet: 4

Sender: Received ACK: 4

Sender: Sent packet: 5
Sender: Sent packet: 6

Sender: Received ACK: 5
Sender: Received ACK: 6

Sender: Sent packet: 7
Sender: Sent packet: 8
Sender: Sent packet: 9

Sender: Received ACK: 7
Sender: Received ACK: 8
Sender: Received ACK: 9

Sender: Packet 10 lost
Sender: Packet 11 lost
Sender: Sent packet: 12
Sender: Sent packet: 13

Sender: Received ACK: 9
Sender: Received ACK: 9
Sender: Timeout occurred, retransmitting...

Sender: Sent packet: 10

Sender: Received ACK: 10

Sender: Sent packet: 11
Sender: Sent packet: 12

Sender: Received ACK: 11
Sender: Received ACK: 12

Sender: Sent packet: 13
Sender: Sent packet: 14

Sender: Received ACK: 13
```

```
Sender: Received ACK: 14

All packets acknowledged.
Sender: Tearing down connection...
Sender: Connection closed.
```

Receiver Output

```
Receiver: Waiting for connection...
Receiver: Received SYN, sending SYN-ACK...
Receiver: Connection established.
Receiver: Received packet: 0
Receiver: Received packet: 1
Receiver: Received packet: 2
Receiver: Received packet: 3
Receiver: Received packet: 4
Receiver: Received packet: 5
Receiver: Received packet: 6
Receiver: Received packet: 7
Receiver: Received packet: 8
Receiver: Received packet: 9
Receiver: Received packet: 10
Receiver: Received packet: 11
Receiver: Received packet: 12
Receiver: Received packet: 13
Receiver: Received packet: 14
Receiver: connection teardown initiated by sender.
Receiver: connection closed.
```