Homework 1

1. Introduction

This document presents a comparative analysis of the Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) based on experimental data. The experiments measured the time required to transfer data sizes of approximately 500 MB and 1 GB using different message sizes and transfer methods, including continuous streaming and a stop-and-wait mechanism for UDP. The goal is to evaluate the performance, reliability, and efficiency of each protocol under varying conditions.

2. Usage Instructions for the Client and Server

This section provides detailed instructions on how to build and use the client and server programs for measuring data transfer performance using TCP and UDP.

2.1 Building the Client and Server

The client and server programs are implemented in Rust. To build them, ensure you have Rust installed on your system. Then, navigate to the project directory and build both the client and server.

2.2 Running the Server

Start the server before running the client. The server will listen for incoming connections and process data transmissions.

By default, the server listens on a predefined port for both TCP and UDP transmissions.

2.3 Running the Client

The client is used to send data to the server, measuring the transmission time and performance. The command-line syntax for running the client is:

Usage: hw1_client [OPTIONS] <PROTOCOL> <MESSAGE_SIZE> <TRANSFER_AMOUNT>

Arguments:

- `<PROTOCOL>`: Specifies the transport protocol to use. Possible values:

    - `tcp` – Uses TCP for data transmission.
    - `udp` – Uses UDP for data transmission.
- `<MESSAGE_SIZE>`: Defines the size of each message in bytes.
- `<TRANSFER_AMOUNT>`: Defines the total amount of data to transfer. Possible values:
    - `small` – Transfers approximately 500 MB.
    - `big` – Transfers approximately 1 GB.

Options:

- `-s, --stop-and-wait`: Enables the stop-and-wait mechanism (only applicable for UDP). In this mode, the client waits for an acknowledgment after each message before sending the next one.
- `-h, --help`: Displays help information.

Example Usage:

Send 500 MB using TCP with 1024-byte messages:

./target/release/hw1_client tcp 1024 small

Send 1 GB using UDP with 8192-byte messages (continuous streaming):

./target/release/hw1_client udp 8192 big

Send 500 MB using UDP with 100-byte messages and stop-and-wait mechanism:

./target/release/hw1_client udp 100 small -s

By using these commands, users can test different configurations and analyze network performance under various conditions.

3. Experimental Setup

- **Protocols Tested**: TCP and UDP
- **Data Sizes**: Approximately 500 MB ("small") and 1 GB ("big")
- **Message Sizes**: 100 bytes, 1024 bytes, 8192 bytes, 16384 bytes, and 65536 bytes. Note: For UDP, the maximum message size tested is 8192 bytes, because osx has a lower maximum datagram size.
- **Transfer Methods**:
    - **TCP**: Continuous streaming
    - **UDP**:
        - Continuous streaming ("no stop")
        - Stop-and-wait mechanism ("stop")

4. Results

The experimental results are summarized in the tables below:

Table 1: TCP Performance

Data Size	Message Size	Client Messages	Client Bytes	Time (s)	Server Messages	Server Bytes
500 MB	100	5,000,001	529,868,431	30.14	5,000,001	640,000,004
500 MB	1,024	488,283	503,775,455	5.22	488,283	513,672,668
500 MB	16,384	30,519	500,189,521	2.45	30,519	500,861,420
500 MB	65,536	7,631	500,100,221	2.00	7,631	500,253,324
1 GB	65,536	15,260	1,000,135,397	3.94	15,260	1,000,441,080
1 GB	16,384	61,037	1,000,379,541	4.87	61,037	1,001,722,836
1 GB	1,024	976,564	1,007,681,447	10.33	976,564	1,027,344,280
1 GB	100	10,000,001	1,059,868,431	60.02	10,000,001	1,280,000,004

Table 2: UDP Performance

Data Size	Message Size	Transfer Method	Client Messages	Client Bytes	Time (s)	Server Messages	Server Bytes
500 MB	100	Stop-and-wait	5,000,002	534,868,433	75.06	5,000,002	534,868,433
500 MB	100	No stop	5,000,002	534,868,433	20.68	4,726,990	505,656,149
500 MB	1,024	No stop	488,284	504,263,739	3.73	488,284	504,263,739
500 MB	1,024	Stop-and-wait	488,284	504,263,739	9.09	488,284	504,263,739

500 MB	8,192	No stop	61,038	500,433,667	2.38	61,038	500,433,667
500 MB	8,192	Stop-and-wait	61,038	500,433,667	3.09	61,038	500,433,667
1 GB	8,192	Stop-and-wait	122,073	1,000,972,704	6.16	122,073	1,000,972,704
1 GB	8,192	No stop	122,073	1,000,972,704	4.73	122,073	1,000,972,704
1 GB	1,024	No stop	976,565	1,008,658,012	7.36	976,565	1,008,658,012
1 GB	1,024	Stop-and-wait	976,565	1,008,658,012	18.14	976,565	1,008,658,012
1 GB	100	Stop-and-wait	10,000,002	1,069,868,433	149.48	10,000,002	1,069,868,433
1 GB	100	No stop	10,000,002	1,069,868,433	41.54	9,488,375	1,015,124,344

5. Analysis

5.1 Transmission Time

- TCP: Transmission time decreases as message size increases. For 500 MB transfers, the time reduced from 30.14 seconds (100-byte messages) to 2.00 seconds (65,536-byte messages). Similarly, for 1 GB transfers, the time decreased from 60.02 seconds (100-byte messages) to 3.94 seconds (65,536-byte messages). This confirms that larger message sizes reduce overhead, improving efficiency.
- UDP: Transmission time is significantly affected by the transfer method. In stop-and-wait mode, transmission takes longer due to the acknowledgment requirement. For example, in the 500 MB, 100-byte case, the stop-and-wait method took 75.06 seconds, while continuous streaming took only 20.68 seconds. However, streaming without acknowledgment results in higher packet loss.

5.2 Data Loss Rates

- TCP ensures reliability with no data loss, as all messages are retransmitted if lost. The byte count at the server consistently matches the client's output.

- UDP exhibits significant data loss when using the no-stop mode. On LAN, the data loss has been minimal, and increased for numerous small messages instead of the large ones. This may not be representative for real-world applications. For example, in the 1 GB transfer using 100-byte messages, the client sent 10,000,002 messages, but the server received only 9,488,375 messages, indicating a loss of approximately 5.1%.
- Impact of Stop-and-Wait: While stop-and-wait reduces loss rates to zero, it greatly increases transmission time.

5.3 Impact of Message Size on Performance

- TCP: Larger messages improve efficiency by reducing per-message overhead. The performance gain is evident from the sharp decrease in transfer time with increased message size.
- UDP: Larger messages reduce loss in no-stop mode by decreasing the number of packets transmitted, which improves delivery rates. The 8192-byte messages resulted in efficient transmission with minimal loss (in this LAN environment).

6. Conclusions

- TCP is ideal for reliable transmissions where data integrity is critical, as it ensures complete and ordered delivery. However, its performance overhead is higher for small message sizes.
- UDP is optimal for speed-sensitive applications, such as video streaming and gaming, where some data loss is acceptable. The no-stop method is much faster but prone to packet loss, while stop-and-wait ensures delivery at the cost of increased latency.
- Message size plays a crucial role in both protocols, with larger messages leading to better efficiency and reduced transmission overhead.