Exercise 1 Report

By Omer Ben Moshe and Yuvali Leibovich

This report details the implementation and analysis of a network throughput measurement system using two C programs: a client and a server. The goal is to evaluate the data transmission rate between the client and server for different message sizes, providing insights into network performance.

The client program is designed to send data of varying sizes to the server while measuring the time taken for these transmissions. The main workflow of the client program begins with configuring the server IP address, which can be passed as a command-line argument or defaults to a predefined IP. A socket is created, then the client attempts to connect to the server. The program iterates over different message sizes, starting from 1 byte and doubling each time up to 1 MB. Before measuring the throughput, the client undergoes a warm-up phase, sending a number of messages to stabilize the connection before the main data transmission phase, depending on the message size. For messages up to size 1024 B, the warmup phase consists of 500 messages, and for bigger messages the warmup consists of 200 messages. After this phase, the time taken to send a specified number of messages, depending on the message size, is measured, and throughput is calculated in Mbps. for messages of size 1 B up to 1024 B we send 1,000,000 messages. For the next 5 sizes of messages (1024 B up to 32,768 B), we send 25,000 messages. Next for messages of sizes 32,768 B to 524,288 B, we send 1,000 messages, and finally, for the MB messages, we send 800 messages. Lastly, the socket is closed, and the buffer is freed to clean up resources.

The server program is designed to receive data from the client and acknowledge the connection. The main workflow of the server involves creating a server socket and binding the socket to the specified port and address, and then waiting for connection with the client. Upon accepting a connection, the server enters an infinite loop to read data from the client. If a read error occurs, it is handled with an error message, and the program exits. Finally, the server socket is closed, and the buffer is freed to clean up resources.

In summary, the client program connects to the server and sends messages of increasing sizes, measuring the throughput for each message size. The server program listens for connections, receives data from the client, and handles any read errors, ensuring robust communication between the client and server.

Message Size	Throughput	Units
2^0	11.911028	Mbps
2^1	23.896406	Mbps
2^2	45.802111	Mbps
2^3	94.980679	Mbps
2^4	194.852989	Mbps
2^5	384.231499	Mbps
2^6	756.900993	Mbps
2^7	906.496722	Mbps
2^8	898.72237	Mbps
2^9	897.991183	Mbps
2^{10}	900.805799	Mbps
2^{11}	893.573947	Mbps
2^{12}	898.284766	Mbps
2^{13}	898.433034	Mbps
2^{14}	897.051758	Mbps
2^{15}	890.71994	Mbps
2^{16}	896.804409	Mbps
2^{17}	898.117516	Mbps
2^{18}	897.891702	Mbps
2^{19}	897.558284	Mbps
2^{20}	898.170838	Mbps