Emily Lakic
CS428 Programming Assignment 1

Design Document

Design Decisions Outline

This design document outlines the design decisions I made while implementing a UDP-based client ping C++ program. My client program sends a simple ping message to a server, receives a corresponding pong message back from the server, and determines the delay between when the client sent the ping message and received a pong message back. My client code sends 10 ping messages to the target server over UDP. For each of those messages, in the case that the packet was not lost, the client calculates the Round Trip Time (RTT) and prints it on the client side. The delay is calculated by the difference between the time the client receives the corresponding pong message back, and the time when the client sent the initial ping message to the server. Given UDP is an unreliable protocol, a packet sent by the client or server may be lost. A timeout system is implemented to combat the client waiting indefinitely for a reply. The client will wait 1 second for a reply, and if none is received, will assume that the packet was lost and print a message accordingly. This program demonstrates the unreliability of UDP and the subsequent need for timeouts in addition to a method to deal with lost data. Design decisions I made were as follows:

1) Configured the client to access the same port number the server was running on.
2) Tracked the number of pings through a for loop and printed each using an id counter to distinguish between pings.
3) Determined sending time and receiving time of each message on the client side.
4) Implemented an associated 1 second timeout system.
5) Printed either 1: The calculated RTT delay or 2: “Packet Was Lost” on the client side. If a response was received by the client from the server, the client determined the RTT delay and printed it. Otherwise, the packet was lost and the client printed a message accordingly.

Sample Outputs

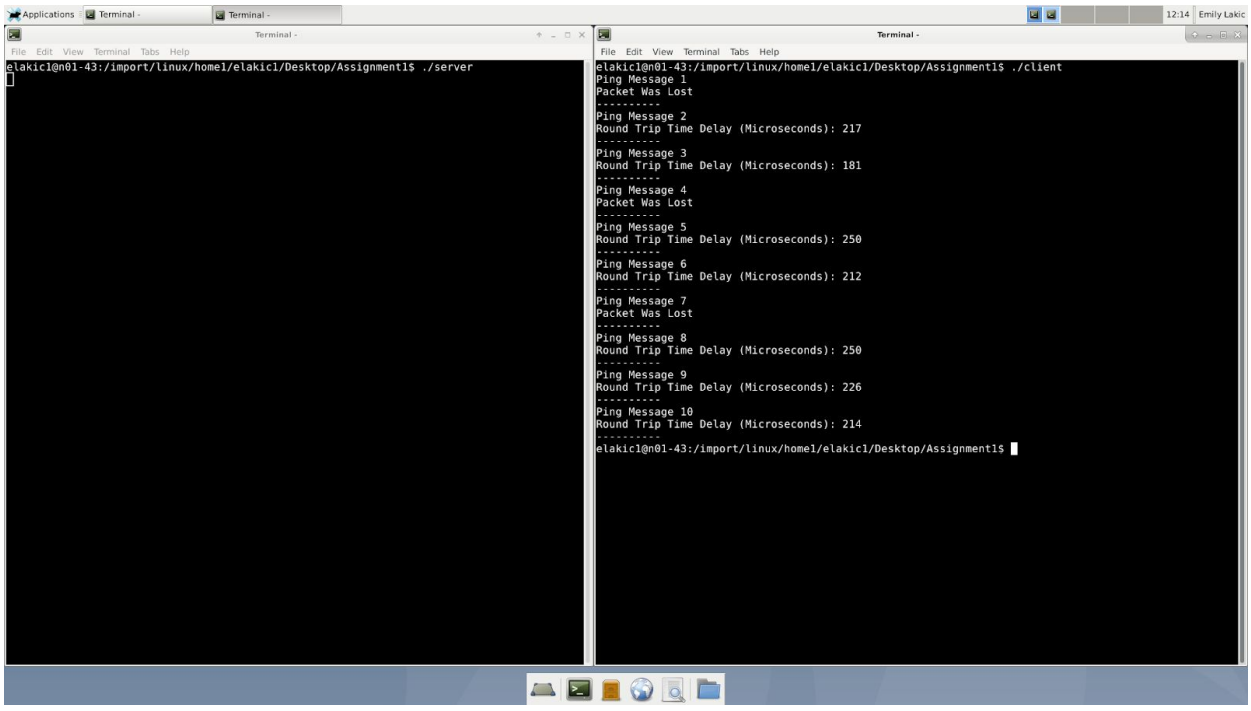

Figure 1. Client and Server Sample Output 1

Emily Lakic
CS428 Programming Assignment 1

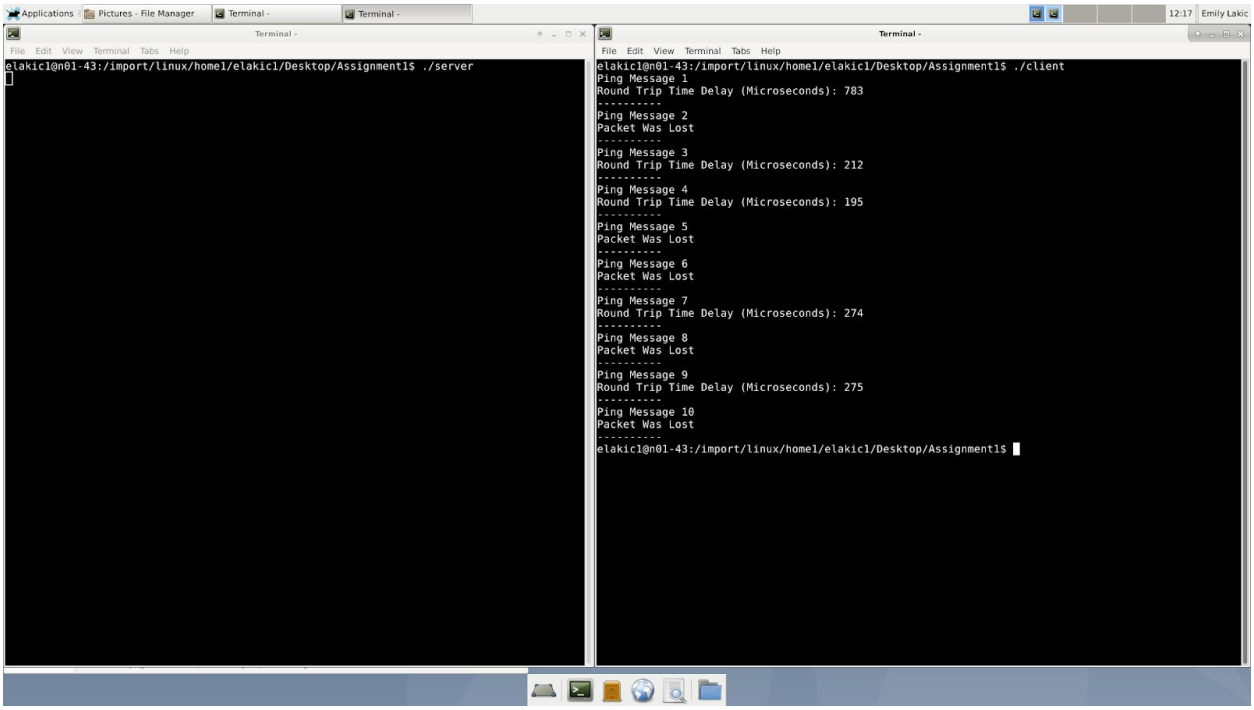

Figure 2. Client and Server Sample Output 2

In both Figures 1 and 2, the server program is running on the left and the client program is running on the right. Each figure shows a unique output, with a different number of packets lost and RTT delays calculated for each. In both outputs, the code runs correctly.

Running Code

This programming assignment was run on the Linux machines, which is in essence equivalent to running remotely from a local machine. I was not able to run the client and server programs on my Mac machine due to an error with the MSG_CONFIRM flag not being declared on Mac OS. Thus, all screenshots included are of the Linux environment in which this programming assignment was completed on. Figures 1, 2, and 3 are screenshots that show the code running correctly on the University's Linux machine.

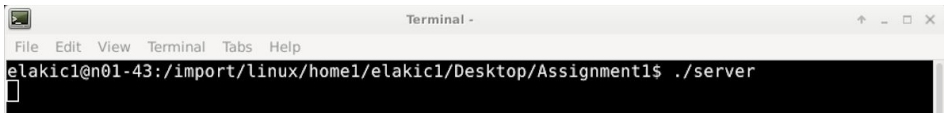

Figure 3. Server Terminal

Emily Lakic
CS428 Programming Assignment 1

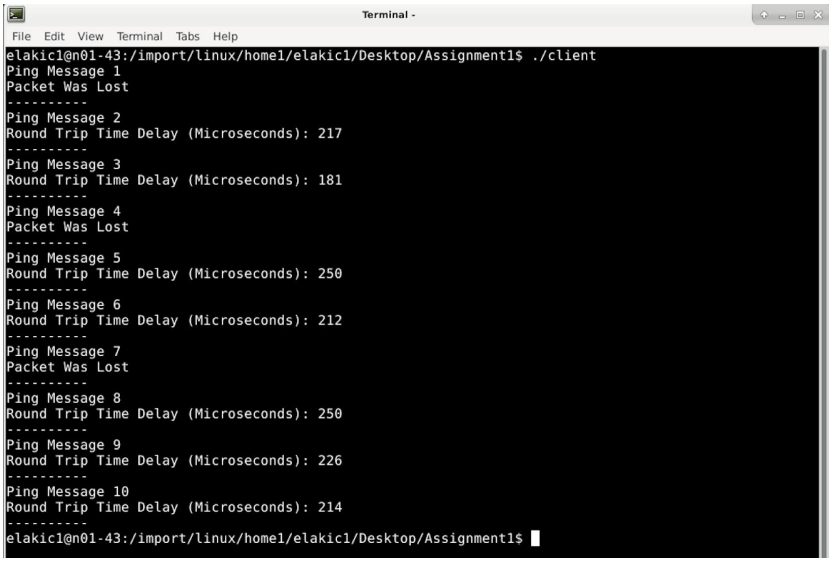

Figure 4. Example Client Terminal

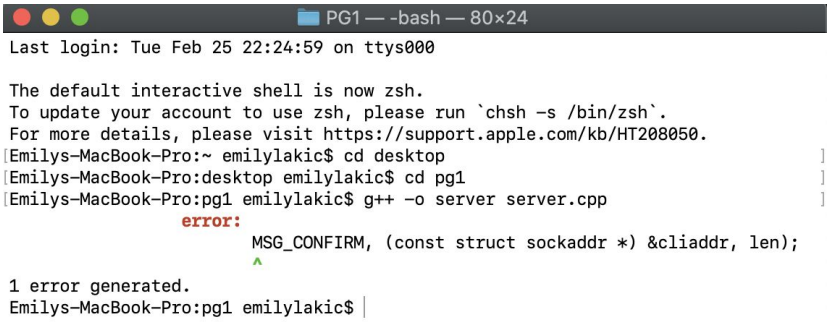

Figure 5. Error Running on Mac

Cases Code Might Fail and Possible Ways of Improving Program

One possible way of improving my program is to implement better error handling. Another possible addition is bug checking for timeout errors and possible false negatives on packet loss. Finally, greater accuracy would be sufficient on the Round Trip Time (RTT) calculation.