COMP 5320/6320/6320-D1 - Design and Analysis of Computer Networks
Programming Lab Assignment 1 Report

Group Members:
Spencer Purdy, scp0069@auburn.edu
Sam Chitty, slc0089@auburn.edu

Submission Date: September 20, 2024

---

## 1. Lab Overview

The lab assignment consisted of two main parts:

Lab 1.1: UDP "PING" Tester
The goal was to develop a UDP echo server and two clients. One client sends a single message, while the other sends 10,000 messages, recording statistics such as round-trip time (RTT), missing messages, and more.

Lab 1.2: TCP Network Calculator
The task involved developing a TCP server and client to perform basic arithmetic operations: addition, subtraction, multiplication, and division. The client sends two unsigned integers and an operation code to the server, and the server sends back the result.

---

## 2. Protocol Adherence

Lab 1.1 (UDP):
The message format was implemented as per the protocol defined in the RFC document. Each message contained a sequence number, a timestamp in milliseconds since the epoch, and a UTF-8 encoded string message of up to 1024 bytes.

Message Structure:
2 bytes: Message length.
4 bytes: Sequence number.
8 bytes: Timestamp (milliseconds since January 1, 1970).
Up to 1024 bytes: Message string.

Lab 1.2 (TCP):
The TCP message format was also followed strictly as per the protocol. The client bundled two operands and an operation code into a 9-byte message, and the server responded with a 14-byte message containing the operands, operation, result, and a validity flag.

---

## 3. Compilation and Execution Instructions

Compilation Commands:

To compile the server and client programs, the following commands were used:

Compile server11.c (UDP server)
```
gcc -o server11 server11.c -lpthread
```

Compile client11b.c (UDP client 1)
```
gcc -o client11b client11b.c
```

Compile client11c.c (UDP client 2)
```
gcc -o client11c client11c.c
```

Compile server12.c (TCP server)
```
gcc -o server12 server12.c
```

Compile client12.c (TCP client)
```
gcc -o client12 client12.c
```

Execution Commands:

Lab 1.1 (UDP):
1. Start the UDP server:
```
./server11
```

2. Run the first UDP client to send a single message:
```
./client11b 127.0.0.1
```

Example input:
```
Enter a message to send (up to 1024 characters): This is a message.
```

3. Run the second UDP client to send 10,000 messages:
```
./client11c localhost
```

Lab 1.2 (TCP):
1. Start the TCP server:
```
./server12
```

2. Run the TCP client with operands and an operator as the ip address is hardcoded to be 127.0.0.1:
```
./client12 15 5 +
```

---

4. Output and Results

Lab 1.1 (UDP)

Client 11b Output:

```
Enter a message to send (up to 1024 characters): This is a message.
Received from server: This is a message.
Round-trip time: 0 ms
```

Client 11c Output:

```
Summary Report:
Total messages received: 10000
Missing messages: 0
Min RTT: 0 ms
Max RTT: 2 ms
Average RTT: 0 ms
Sender finished sending messages
```

Lab 1.2 (TCP)

TCP Client Output:

```
./client12 15 5 +
Result: 15 + 5 = 20

./client12 15 5 -
Result: 15 - 5 = 10

./client12 15 5 x
Result: 15 x 5 = 75

./client12 15 5 /
Result: 15 / 5 = 3
```

---

## 5. Code Functionality and Observations

All programs compiled successfully and ran without any issues. The following is a summary of the program functionality:

Lab 1.1:
The server successfully handled both UDP clients. Client 1 sent a single message, which the server echoed back. Client 2 sent 10,000 messages, and all messages were successfully received by the server with accurate RTT statistics and no missing messages.

Lab 1.2:
The TCP calculator worked as expected, correctly handling addition, subtraction, multiplication, and division. All responses from the server were accurate, and the results were returned in the correct format.

There were no errors, bugs, or missing functionality in the code.

---

## 6. Conclusion

In this assignment, we successfully implemented both the UDP and TCP components following the protocol definitions provided in the RFC documents. The server-client interaction in both cases was handled smoothly, and the outputs were correct. We completed the assignment as specified and tested the programs thoroughly.

---

## 7. Bugs/Problems

No bugs or issues were encountered during the implementation and testing of this lab.

---

## 8. How to Compile and Run the Code

1. Compile the programs using the `gcc` commands provided in Section 3.
For programs that involve threading (such as `server11.c`), ensure to include the `-lpthread` flag during compilation. This allows for proper multithreading support in handling concurrent operations.

Example:

```
gcc -o server11 server11.c -lpthread
```

2. Run the servers and clients using the specified commands.
Make sure to start the server before executing the corresponding clients.

3. Make sure the server is running before starting the corresponding clients.

---

## 9. Submission Information

1. The code and report are provided in a folder named `lab1_lastname1_lastname2`, where `lastname1` and `lastname2` correspond to the last names of the group members.
2. The folder has been zipped and uploaded as per the submission instructions.

---

## 10. Group Members

Spencer Purdy, scp0069@auburn.edu
Sam Chitty, slc0089@auburn.edu