EE450 Term Programming Part 3 (Final)

Fall 2019

Due Date: Wednesday December 11th, 2019 11:59 PM (Midnight)

(Strictly Enforced: the submission website will automatically close on the deadline)

The objective of this assignment is to familiarize you with UNIX socket programming. This assignment is worth 3% of your overall grade in this course. It is an individual assignment and no collaborations are allowed. **Any cheating will result in an automatic F in the course (not just in the assignment).**

If you have any doubts/questions, email TA your questions or come by TA's office hours. **You can ask TA any** question **about the content of the project, but TA has the right to reject your request for** debugging**.**

Problem Statement:

In this part of term project, you will implement both TCP and UDP communication between client and servers based on the work in the previous two parts.

A client boots up and establish a TCP connection with Main Server. Then it sends requests to the Main Server. The Main Server will take the request and do the similar **Search** and **Calculation** operation with Database Server and Calculation Server as required in Term Project Part 2. After receiving the results, it will send back to the client.

As bonus points(**extra 20%**), a monitor will boot up before the client. It will printout every events happen in the main server.

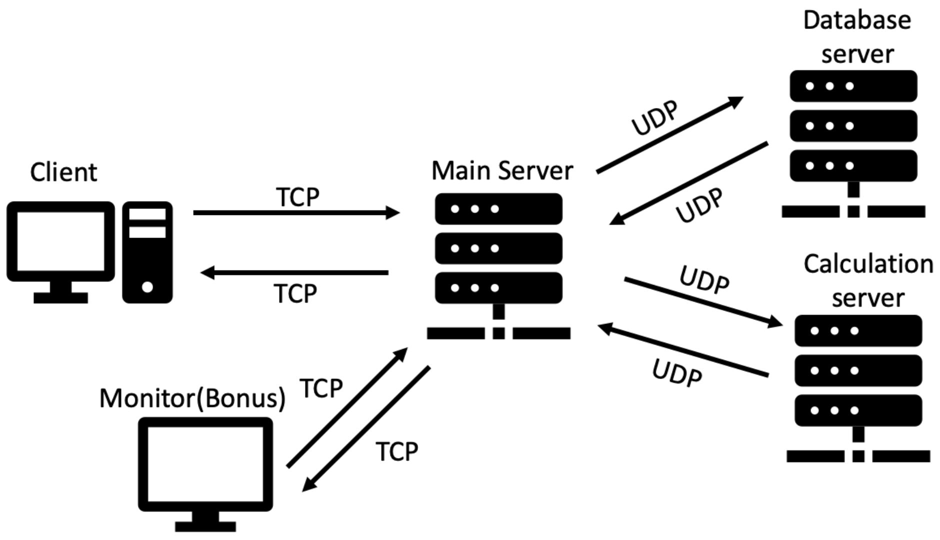

Figure 1. Illustration of the network

For three servers and monitor, they should be capable of handling multiple requests from the clients. They should be always running. For client, once the all the requests got replied and all the required information is printed out, it should terminate itself.

The input to the client should be given as command-line argument of main function, like what is required in *Term Project Part 1*.

Database file provided is only for testing. TA will change the database file for grading. So DO NOT Hardcode any data from database.
The order of boot up should follow in this way: Main Server, Database Server, Calculation Server, Monitor(optional), client.

Source Code Files

Your implementation should include the source code files described below, for each component of the system.

1. Client: You must name your code file: **`client.c`** or **`client.cpp`**. Also you must call the corresponding header file (if you have one; it is not mandatory) **`client.h`**.
2. Monitor: You must name your code file: **`monitor.c`** or **`monitor.cpp`**. Also you must call the corresponding header file (if you have one; it is not mandatory) **`monitor.h`**.
3. Main Server: You must name your code file: **`mainServer.c`** or **`mainServer.cpp`**. Also you must call the corresponding header file (if you have one; it is not mandatory) **`mainServer.h`**.
4. Database Server: The name of this piece of code must be **`dbServer.c`** or **`dbServer.cpp`** and the header file (if you have one; it is not mandatory) must be called **`dbServer.h`**.
5. Calculation Server: he name of this piece of code must be **`calcServer.c`** or **`calcServer.cpp`** and the header file (if you have one; it is not mandatory) must be called **`calcServer.h`**.

Required Port Number Allocation

The ports to be used by the servers for the exercise are specified in the following table:

Table 1. Static and Dynamic assignments for UDP ports

Process	**Dynamic Ports**	**Static Ports**
mainServer	-	21000 + xxx (TCP with client) 22000 + xxx (UDP) 25000 + xxx (TCP with monitor)
dbServer	-	23000 + xxx (UDP)
calcServer	-	24000 + xxx (UDP)
Client	TCP	-
Monitor	TCP	-

NOTE: xxx is the last 3 digits of your USC ID. For example, if the last 3 digits of your USC ID are “319”, you should use the port: **21000+319 = 21319** for the server. **It is NOT going to be 21000319.**

ON SCREEN MESSAGES: Table 2. Client on screen messages	
Event	**On Screen Message (inside quotes)**
Booting up (Only while starting):	"The Client is up and running."
Upon sending to Main Server:	"Send Link <ID> and file size <size>MB to main server."
Upon receiving from the Main Server:	"Receive transmission delay <Tt>ms, propagation delay < Tp>ms and total delay <Tt+Tp>ms" Or "No match found."

ON SCREEN MESSAGES: Table 3. Main Server on screen messages	
Event	**On Screen Message (inside quotes)**
Booting up (Only while starting):	"The Main Server is up and running."
Upon receiving the input:	"Receive Link <ID>, file size <size>MB."
Upon sending to Database Server:	"Send Link <ID> to database server."
Upon receiving from Database Server:	"Receive link capacity <C>Mbps, link length <L>km, and propagation velocity <Vp>km/s." OR "Receive no match found"
Upon sending to Calculation Server:	"Send information to calculation server."
Upon receiving from the Calculation Server:	"Receive transmission delay <Tt>ms, propagation delay < Tp>ms and total delay <Tt+Tp>ms"

ON SCREEN MESSAGES: Table 4. Database Server on screen messages	
Event	**On Screen Message (inside quotes)**
Booting up (Only while starting):	“The Database Server is up and running.”
Upon receiving from main server:	“Receive request from Main Server.”
Upon finishing searching and sending to main server:	“Send link <ID>, capacity <C>Mbps, link length <L>km, propagation velocity <Vp>km/s.” OR “No match found.”

ON SCREEN MESSAGES: Table 5. Calculation Server on screen messages	
Event	**On Screen Message (inside quotes)**
Booting up (Only while starting):	“The Calculation Server is up and running.”
Upon receiving from main server:	“Receive request from Main Server.”
Upon finishing calculating and sending to main server:	“Send transmission delay <Tt>ms, propagation delay <Tp>ms, total delay <Tt+Tp>ms.”

ON SCREEN MESSAGES: Table 6. Monitor on screen messages	
Event	**On Screen Message (inside quotes)**
Booting up (Only while starting):	“The Monitor is up and running.”
Upon Main Server receives Request from client:	“Main server receives Link <ID> and file size <size>MB from client.”
Upon Main Server sends Request to the Database Server:	“Main server sends Link <ID> to database server.”
Upon Main Server receives	“Main server receives

message from Database server:	information from database server: link capacity <C>Mbps, link length <L>km, and propagation velocity <Vp>km/s.” OR “Main server receives information from database server: no match found”
Upon Main Server sends Request to the Calculation Server:	“Main Server sends information to calculation server.”
Upon Main Server receives information from calculation server:	“Main Server receives information from calculation server.”
Upon Main Server sends information to client:	“Main Server sends information to client: transmission delay <Tt>ms, propagation delay <Tp>ms, total delay <Tt+Tp>ms.”

Example to Illustrate Output Formatting:

Client Terminal:

```
The Client is up and running
Send Link 1 and file size 100MB to main server.
Receive transmission delay 800.00ms, propagation delay 0.33ms, and total delay
800.33ms
```

Main Server Terminal:

```
The Server is up and running
Receive link 1, file size 100MB.
Send Link 1 to database server.
Receive link capacity 1000Mbps, link length 100km, and propagation velocity
300000km/s.
Send information to calculation server.
Receive transmission delay 800.00ms, propagation delay 0.33ms, and total delay
800.33ms.
```

Database Server Terminal:

```
The Database Server is up and running.
```

```
Receive request from Main Server.
Send link 1, capacity 1000Mbps, link length 100km, propagation velocity 300000km
/s.
```

Calculation Server Terminal:

```
The Calculation Server is up and running.
Receive request from Main Server.
Send transmission delay 800.00ms, propagation delay 0.33ms, total delay
800.33ms.
```

Monitor Terminal:

```
The Monitor is up and running.
Main server receives Link 1 and file size 100MB from client.
Main server sends Link 1 to database server.
Main server receives information from database server: link capacity 1000 Mbps,
link length 100km, and propagation velocity 300000km/s.
Main Server sends information to calculation server.
Main Server receives information from calculation server.
Main Server sends information to client: transmission delay 800.00ms, propagation
delay 0.33ms, total delay 800.33ms.
```

Assumptions:

1. If you need to have more code files than the ones that are mentioned here, please use meaningful names and all small letters and mention them all in your README file.

2. You are allowed to use blocks of code from Beej's socket programming tutorial (Beej's guide to network programming) in your project. However, you need to mark the copied part in your code.

3. When you run your code, if you get the message "port already in use" or "address already in use", **please first check to see if you have a zombie process** (see following). If you do not have such zombie processes or if you still get this message after terminating all zombie processes. If you have to change the port number, please do mention it in your README file and provide reasons for it.

Requirements:

1. The host name must be hardcoded as **localhost (127.0.0.1)** in all codes.

2. The servers should keep running and be waiting for another request until the TAs terminate them by Ctrl+C. It they terminate before that, you will lose some points for it.

3. All the naming conventions and the on-screen messages must conform to the previously mentioned rules.

4. All the on-screen messages must conform exactly to the project description. You should not add anymore on-screen messages. If you need to do so for the debugging purposes, you must comment out all of the extra messages before you submit your project.

Programming platform and environment:

1. All your submitted code **MUST** work well on the provided virtual machine Ubuntu.

2. All submissions will only be graded on the provided Ubuntu. TAs won't make any updates or changes to the virtual machine. It's your responsibility to make sure your code working well on the provided Ubuntu. "It works well on my machine" is not an excuse and we don't care.

3. Your submission MUST have a Makefile. Please follow the requirements in the following "Submission Rules" section.

Programming languages and compilers:

You must use only C/C++ on UNIX as well as UNIX Socket programming commands and functions. Here are the pointers for Beej's Guide to C Programming and Network Programming (socket programming):

http://www.beej.us/guide/bgnet/

(If you are new to socket programming please do study this tutorial carefully as soon as possible and before starting the project)

http://www.beej.us/guide/bgc/

Also inside your code you need to include these header files in addition to any other header file you think you may need:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
```

```
#include <errno.h>
#include <string.h>
#include <netdb.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <sys/wait.h>
```

Submission Rules:

1. Along with your code files, include a **README file and a Makefile**. In the README file write
   a. Your **Full Name** as given in the class list
   b. Your Student ID
   c. What you have done in the assignment.
   d. What your code files are and what each one of them does. (Please do not repeat the project description, just name your code files and briefly mention what they do).
   e. The format of all the messages exchanged.
   g. Any idiosyncrasy of your project. It should say under what conditions the project fails, if any.
   h. Reused Code: Did you use code from anywhere for your project? If not, say so. If so, say what functions and where they're from. (Also identify this with a comment in the source code.)

Submissions WITHOUT README AND Makefile WILL BE SUBJECT TO A SERIOUS PENALTY

Makefile tutorial:
https://www.cs.swarthmore.edu/~newhall/unixhelp/howto_makefiles.html

About the Makefile: makefile here is both for compiling and running the codes.

`make all`	Compiles **all** your files and creates executables
`make mainServer`	Run Main Server
`make dbServer`	Run Database Server
`make calcServer`	Run Calculation Server
`make monitor`	Run Monitor (bonus points)
`./client <ID> <file size>`	Run client

2. Compress all your files including the README file into a single “tar ball” and call it: **ee450_TermPart3_yourUSCusername.tar.gz** e.g. my filename would be **ee450_TermPart3_songhao.tar.gz**. Please make sure that your name matches the one in the class list. Here are the instructions:

   a. On your VM, go to the directory which has all your project files. Remove all executable and other unnecessary files. **Only include the required source code files, Makefile and the README file**. Now run the following commands:

   b.
   ```
   >> tar cvf ee450_TermPart3_yourUSCusername.tar *
   >> gzip ee450_ TermPart3_yourUSCusername.tar
   ```
   Now, you will find a file named “ee450_TermPart3_yourUSCusername.tar.gz” in the same directory. Please notice there is a star(*) at the end of first command.
   Any compressed format other than .tar.gz will NOT be graded!

3. Please DO NOT wait till the last 5 minutes to upload and submit because some technical issues might happen and you will miss the deadline. And a kind suggestion, if you still get some bugs one hour before the deadline, please make a submission first to make sure you will get some points for your hard work!

4. After receiving the confirmation email, please confirm your submission by downloading and compiling it on your machine. If the outcome is not what you expected, try to resubmit and confirm again. We will only grade what you submitted even though it’s corrupted.

Grading Criteria:

Your project grade will depend on the following:

1. Correct functionality, i.e. how well your programs fulfill the requirements of the assignment, especially the communications through TCP sockets.

2. Complete README file: 10 out of 100;

3. Working Makefile: 10 out of 100;

4. Client implementation: 20 out of 100;

5. Main sever implementation: 20 out of 100;

6. Database server implementation: 20 out of 100;

7. Calculation server implementation: 20 out of 100;

8. Monitor implementation: 20 bonus points.

9. If your code does not correctly assign the TCP/UDP port numbers, you will lose 10 points each.

10. Your code will not be altered in any ways for grading purposes and however it will be tested with different inputs. Your TA/grader runs your project as is, according to the project description and your README file and then check whether it works correctly or not. If your README is not consistent with the description, we will follow the description.

Academic Integrity:

All students are expected to write all their code on their own.

Copying code from friends is called **plagiarism** not **collaboration** and will result in an F for the entire course. Any libraries or pieces of code that you use and you did not write must be listed in your README file. All programs will be compared with automated tools to detect similarities; examples of code copying will get an F for the course. **IF YOU HAVE ANY QUESTIONS ABOUT WHAT IS OR ISN'T ALLOWED ABOUT PLAGIARISM, TALK TO THE TA.** "I didn't know" is not an excuse.