# Network Programming Project

Milestone 2

Anthony Alves, Sally Tan

CSE 4232 – E1 – Network Programming

Dr. Marius Silaghi

March 1, 2016

Table of Contents

# 1. Introduction

This program runs a server that stores projects with their respected tasks. The program stores this information into a SQLite database. A client can connect to the server with the port given to the server. Once the server has started, it loads the SQLite database and waits for clients. For TCP, once a client is connected, a new thread is made to handle the client's commands. The server can handle multiple amounts of clients. UDP uses one thread for all communication. Each thread handler then parses the input given by the client and executes the commands.

# 2. Architecture

The multi-client server handles each client in a separate thread. Each thread access the database through the same DB object. A full JavaDoc to the program can be found at http://my.fit.edu/~aalves2012/np/

## 2.1 Starting the Server

The server starts with the default database file in the root directory of the project. The default port is also 4232. To give custom arguments to the program, pass the port followed by the run.sh file. The Main class is called and binds the server to the localhost. After the server has been created, the DB object is then created. While the server is connected, it will wait to accept clients. When a client is accepted, a new thread is

created passing the client socket and database object. The program ends when the ^c character is given.

## 2.2 Handling the Client

Each client is handled in concurrent when connected to the Server. The class, TCPClientHandler, has access to the database and receives input from the client and sends output back. When the handler receives input, it parses the command and queries the database with the given command. If the client disconnects, all streams are closed and the thread is destroyed.

## 2.3 Program Commands

The server currently handles four commands given by the client. The commands access the database and send the current information back to the client. The commands are:

- ***PROJECT_DEFINITION:project_name;TASKS:n;task_name;start_date;end_date;project_start_date;project_end_date***

  To create a project, the client needs to supply the project definition tag, project name, the amount of tasks, the tasks repeated n amount of times in the format (task_name;start_date;end_date;), followed by the project start and end date. The return value is the OK tag followed by the original input.

- ***TAKE;USER:name;PROJECT:project_name;task_name***

  To assign a user to a task, supply the take tag, the user name, the project name, followed by the task name to assign. The IP and the port of the

client is also saved to the task. The return value is the OK tag followed by the original input. Ex:

- ***GET_PROJECTS***

  This tag returns the amount of projects saved in the database and each project name.

  Example output: ***OK;PROJECTS:n;project_name***

- ***GET_PROJECT;project_name***

  This command returns the OK tag followed by the content of the project.

  Example output: ***OK;PROJECT_DEFINITION:project_name;TASKS:n;task_name;start;end;user_name;IP;port;status;project_start;project_end***

## 2.4 Accessing the Database

When a client sends a request for information, the database object contains methods to get the information. Each request has a method that build a SQL query and returns the formatted output.

- ***GET_PROJECT =*** `getProject(String projectTitle)`
- ***GET_PROJECTS =*** `getProjects()`
- ***PROJECT_DEFINITION =*** `addProject(String rawInput)`
- ***TAKE =*** `assignUser(String user, String projectTitle, String taskTitle, String userIP, int userPort)`

# 3. User Manual

To compile the program, run the make command in the root directory. To start the server, run the command:

```
make
./run.sh &
```

The command starts the server on localhost and port 4232 and creates a SQLite database in the root directory. For advanced methods, refer to the READ.ME file.

To connect to the server, you can run one of two scripts. script1.sh starts a client with premade commands. script2.sh runs a client and waits for user input for custom queries. Both scripts support the use for a custom address and custom port. Ex:

```
./script1.sh
```

or

```
./script2.sh 4232
```

You can now send commands to the server. If a custom client is used, the server takes and sends commands followed by \n.

For UDP tests, run either commands:

```
./script1udp.sh
```

or

```
./script2udp.sh 4232
```

## 4. Conclusion

The program will run until the user stops the server. Each client can send multiple request per session and multiple clients can connect simultaneously. Support TCP and UDP clients.