

Project Milestone #2

Dmitry Kulakov <dkulakov2014@my.fit.edu>

Xiaohan Yang <xiaohan2014@my.fit.edu>

Nicholas Zynko <nzynko2008@my.fit.edu>

March 5, 2017

1. Introduction

This document describes the specifications for the C++ iterative server that stores a set of peers and gossip messages in a sqlite database. This server can recognize commands both over TCP and UDP on the same port. The following three types of messages are accepted by the server:

- *GOSSIP:[SHA256_HASH]:DATE:[MESSAGE]%*
- *PEER:[NAME]:PORT=[PORT_NUM]:IP=[IP]%*
- *PEERS?*

The GOSSIP command stores the message, date and hash in a GOSSIP table, with the hash acting as a primary key to prevent duplicate messages. It also broadcasts the message to all the known peers over the socket that is currently open. The PEER command stores the known peer as a primary key along with its port number and ip address. The PEERS? command lists the total number of peers, and lists the peers currently in the database.

2. Architecture

2.1. Overall

This C++ program is designed to allow the server to get the port number to listen on and file path to sqlite database. The program

The TCP thread is used to listen the incoming connections and spawns a thread for a new connection; The UDP thread is used to listen the incoming datagrams without spawning a new thread to handle them.

2.2. Variables

This project contains the following global variables (including default values):

- `#define MAX_CALLS 5`
 - Defines the maximum number of calls that the server will accept when it is listening for sockets
- `#define MAXLINE 128`
 - The maximum number of bytes in allowed in a buffer
- `PathSeparator`
 - Used to distinguish betwe
- `Inr tcpfd, udpfd, confd`
 -
- `sqlite3 *db`
 - The global reference to the sqlite database
- `string latestGossip`
 - Contains the latest gossip message to be broadcast

2.3. Functions

This project contains the following functions:

- `void signal_stop (int a)`
 - Called upon hearing the control-C or control-Z signal, used to close all available file descriptors and database connections to prevent zombie processes from lingering after the server is shut down.
- `void udp_handler(int sockfd, sockaddr* client, socklen_t client_len)`
 - Creates and handles the udp connection. Calls the reader method on information that it receives from the client, and also sends the list of peers to the client if the result returned from the reader is not null.
- `void tcp_handler (int confd)`
 - Creates and handles the udp connection. Calls the reader method on information that it receives from the client, and also sends the list of peers to the client if the result returned from the reader is not null.
- `char *reader (string fulltext)`
 - Parses the user's commands and stores them in the sqlite database. Returns a reference to the result of the select statement from the callback method, which can be used to send all the available peers.
- `void sig_child (int signo)`

- The method called when a child is forked by the main process.
- static int callback (void * result, int argc, char ** argv, char **azColName)
 - Called during the sqlite3_exec method call in order to structure the output of the select statement after a PEERS? command is received. The void * result holds the result of the select statement, and can be found in the reader method after the exec finishes.
- static int gossipCallback(void *NotUsed, int argc, char **argv, char **azColName)
 - Called during the sqlite3_exec method for broadcasting after a GOSSIP message was received.
- void sendUDP(string message, string address, int port)
 - Interface to send messages via UDP socket.
- void sendTCP(string message, string address, int port)
 - Interface to send messages via TCP socket.
- int setup_database (char *filePath)
 - Sets up sqlite database for storing gossip information and peers.
- int main (int argc, char *argv[])
 - Opens the sockets for tcp and udp connections and selects between the two as messages arrive over the two sockets. Calls udp_handler or tcp_handler based on the file descriptor which is receiving input.

3. User Manual

This project can be runned by the following commands.

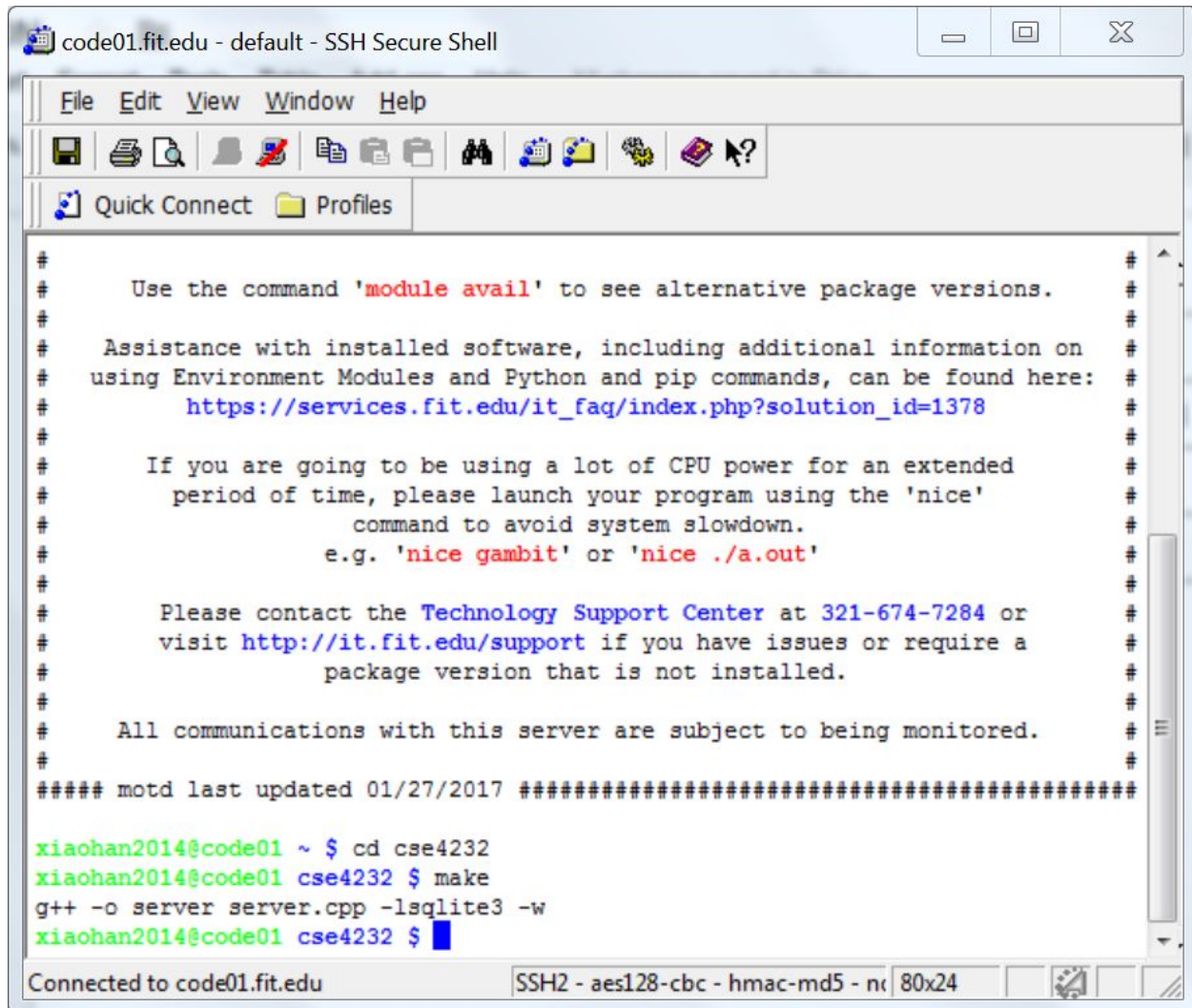
Usage: /server -d name_of_db -p portnum	
-d name_of_db	Assigns the name of the database file to be used
-p portnumber	Assigns the port to be used
-u	Switches to UDP, the default is TCP

To compile the code run, simply type **make**

To run the program, execute **./run.sh [-d <filepath>] [-p <port>]**

4. Snapshots

- Compile the project.



The screenshot shows an SSH terminal window titled "code01.fit.edu - default - SSH Secure Shell". The window has a menu bar (File, Edit, View, Window, Help) and a toolbar with various icons. Below the toolbar is a "Quick Connect" section with a "Profiles" button. The main terminal area displays a series of messages and commands. The messages include instructions on using the 'module avail' command, assistance with installed software, and a warning about CPU usage. The commands executed are 'cd cse4232', 'make', and 'g++ -o server server.cpp -lsqlite3 -w'. The terminal output shows the compilation and execution of the program.

```
# Use the command 'module avail' to see alternative package versions.
# Assistance with installed software, including additional information on
# using Environment Modules and Python and pip commands, can be found here:
# https://services.fit.edu/it_faq/index.php?solution_id=1378
# If you are going to be using a lot of CPU power for an extended
# period of time, please launch your program using the 'nice'
# command to avoid system slowdown.
# e.g. 'nice gambit' or 'nice ./a.out'
# Please contact the Technology Support Center at 321-674-7284 or
# visit http://it.fit.edu/support if you have issues or require a
# package version that is not installed.
# All communications with this server are subject to being monitored.
# ##### motd last updated 01/27/2017 #####
xiaohan2014@code01 ~ $ cd cse4232
xiaohan2014@code01 cse4232 $ make
g++ -o server server.cpp -lsqlite3 -w
xiaohan2014@code01 cse4232 $
```

Connected to code01.fit.edu SSH2 - aes128-cbc - hmac-md5 - nc 80x24

- Connect to the server

- ```
[np_final_project-master] 187 // Storage = error code.
 188 char* zErrMsg = new char();

root@kali: ~#
File Edit View Search Terminal Help
root@kali:~# nc -l 127.0.0.1 8080 -u $SOLITE_OWL
PEER: John:PORT=2356;IP=163.118.239.68%
PEER: JOHN:PORT=2356;163.118.239.68%
PEER: JOHN:PORT=2356;163.118.239.68%
PEER: JOHN:PORT=2356;163.118.239.68%
Ncat: Connection refused.
root@kali:~# nc -l 127.0.0.1 8080 -u % user commands and stores them in an sqlite3
PEER^C
root@kali:~# nc -l 127.0.0.1 8080 --server (string fulltext){
PEER: JOHN:PORT=2356;163.118.239.68%
PEER: JOHN:PORT=2356;163.118.239.68% name, port, ip;
PEER: JOHN:PORT=2356;163.118.239.68% digest, message, date;
PEER: JOHN:PORT=2356;163.118.239.68% salt;
PEER: JOHN:PORT=2356;163.118.239.68% values;
PEER: JOHN:PORT=2356;163.118.239.68% adcsat = false;
PEERS?
PEERS?
121 bool peersRequest = false; // Set to true when PEERS? app
122
123
124 istream stream (fulltext);
125 string split;
126
127 // Buffer for error handling.
128 char** zErrMsg = new char*[512];
129 for(int i = 0; i < 512; ++i)
130 {
131 zErrMsg[i] = new char[512];
132 }
133
134
135 getline (stream, split, '\n');
```