Programming Assignment #1

CSC 688 Go Language and Programming
Department of Computer Engineering
Santa Clara University

Dr. Ming-Hwa Wang — Summer Quarter 2017
Phone: (408) 805-4175 — Email address: mwang2@engr.scu.edu
Course website: http://www.cse.scu.edu/~mwang2/language/
Office Hours: Friday 9:30-10:00pm

Due date: midnight June 4, 2017

Centralized P2P File Sharing, Part I (200 points)

Please implement a centralized P2P file sharing system (e.g., Napster) using Go client-server programming and Go concurrent programming with shared variables. In a centralized P2P file sharing system, a server needs to support huge number of clients for uploading music file locations and get music file locations concurrently. A client can share its music by publishing its IP address and port number on the centralized server. Any client can get the shared music by retrieving the owner IP address and port number from the centralized server and then download the music from one of the owner directly using its IP address and port number. Part I of this programming assignment is to do the core engine of the server program (the parallel hashing), and Part II is to do the server and client programming to put/get music file. To simplify your work, both server and client run on the Linux machines.

To implement parallel hashing or thread-safe hashing, you need to use md5sum (e.g., "md5sum - <<< "<string>", or echo "<string>" | md5sum) and open hashing with linear chains. The hash table size is 2^{2k+1}-1, where *k* is a positive integer starting from 1, i.e., the initial table size is 7. To get O(1) performance the table size should be at least the number of music files. When number of music greater than the table size, you should do a rehashing by increasing *k* by one. To simplify your work, you don't need to implement the reducing table size. The format for inserting a music to the hash table is put("<music name>", <ip>, <port>) which returns a Boolean indicating success or failed. The format for retrieving a music from the hash table is get("<music name>") and which returns a Boolean indicating success or failed and a pair of <ip> and <port> if success. The format for delete a music is delete("<music name>", <ip>, <port>). You should allow multiple owners share the same music (with different <ip> and <port> pairs, but an owner only has one music entry in the hash table. For each put/get/delete, you should have a separate thread handles it to make it parallel hashing. Since this is concurrent programming, any thread needs to display the whole output atomically. Note that you need to pass the Autotest before submitting your p1.

Student Name:

ID:

Score:

Correctness and boundary condition (60%):

Error Handling (5%):

Display output atomically (5%):

Modular design, file/directory organizing, showing input, documentation, coding standards, sympathy/typing point (25%):

Automation (5%):

Subtotal:

Late penalty (20% per day):

Special service penalty (5%):

Total score: