

Chatter

Assignment Out: July, 26
Due: Aug, 2

1 Purpose

This assignment serves as your introduction to C++ programming, including the use of sockets and threads in C++, creating classes to share, use of STL, and how to work as a team in C++.

Your task is to create a peer-to-peer client that will randomly connect to other such clients and exchange messages. The assignment is designed for two people. One person should implement a set of support classes that hide the internals of threads and sockets (The pthreads and sockets APIs you have seen in class) and provide a high level interface for the remainder of the client. The second person should implement the actual client using these support classes.

Your work should make appropriate use of strings (not just char*), RAII for resource management, and the STL container classes.

2 Messages

Your program should listen on port 17329 (this might change so make it a parameter). It should then attempt to connect to other such programs on the same port on other machines. The file `/course/cs032/data/chatter/hosts.full` lists all the machines in the department. Your program should find 3-5 (another parameter) machines from this list that have active servers and connect to them.

Machines, once connected will exchange messages. Messages are simple text strings consisting of a command and a set of arguments. At a low level, a message consists of 4 bytes of length (in network byte order), followed by that number of bytes of data to be treated as ASCII characters.

The set of messages that need to be exchanged include:

WHORU This should set a IAM message to the sender.

IAM `< host > < user >` This message identifies the sender as coming from the given host and being owned by the given user.

CONN `< host > < time > < loadaverage... >` This message is a request for information about the connections known to the receiving host. The parameters identify the sending host, the current time (in seconds since the Epoch), and the load average of the host machine. (The latter can be found by reading `/proc/loadavg.`) Note that the load average might contain spaces.

Upon seeing this message, the receiving host should return to the sender one or more **LOAD** messages describing all the load average on all the machines that this node knows about.

LOAD < *host* > < *time* > < *loadaverage...* > This message provides the receiver with what information the sender has about the given host. The time argument (in seconds since the Epoch) should be used to determine if the given information is more recent than what the receiver already knows. The load average again may contain spaces.

Message processing should be done using threads so that a slow connection does not disrupt other connections.

3 Display and Interaction

Your main program should periodically print a table of all known machines and their load averages. If a machine hasn't been heard from in a while (say 15 minutes), its load average should be ignored. Your program should also periodically (say every minute) send out **CONN** messages to update its tables.

4 Handin

Your grade will be based on your design check and your handin. Your handin should be very easy to compile and run. Your code should be easy to read.

Make a **README** file which explains any quirks in your code, as well as any special instructions about running your program. You should also mention any design decisions you made which you found troubling.

```
/course/cs032/bin/cs032_handin chatter
```