I recently ran into a nice framework to implement network services with remote-procedure calls. Prof. Mitra and I thought it would be a good idea to use it for the MP2, so I am making changes to introduce it to the MP.

----
1) ETA for MP2 is Wednesday, but in the mean time, we want to let you know about the topic:

You will implement a system based on Chord.

Therefore, you should have a clear understanding the Chord algorithms, including those in figure 4 and figure 7 in the Chord paper (http://courses.engr.illinois.edu/ece428/spring2012/papers/chord.pdf). There are subtleties/corner cases in these algorithms that are not shown in those figures, so make sure you trace their steps (for example, using the 3- and then 4-node examples with m=3 shown in the paper), work them out, and catch these subtleties.

You do not need to understand the algorithms in figure 6 because they are inferior to those in figure 7.

You might recall that there was a slight discrepancy between the lecture slides and the Chord paper (">" vs ">="). For the MP, you should follow the comparison used in the paper (">=").

----
2) The aforementioned framework is Thrift http://thrift.apache.org/, which significantly abstracts away the network communication and RPC plumbing, the tedious and error-prone parts, so you can focus on your Chord protocol.

Even though you have to learn to use Thrift, the learning curve is not steep, and in my opinion it will be worth the time. We will recommend using Thrift, but if you are really comfortable with socket programming, you are free to skip Thrift. However, you will be required to use C/C++.

A quick tutorial on implementing a simple "ping" service is here:
http://wiki.apache.org/thrift/ThriftUsageC%2B%2B

I have installed Thrift on the EWS machines, in /class/ece428/libs/

The Thrift code generator is /class/ece428/libs/bin/thrift

When building the ping service server and client, add to the g++ command -DHAVE_NETINET_IN_H -I /class/ece428/libs/include/thrift/ -L /class/ece428/libs/lib/

```
g++ -Wall -DHAVE_NETINET_IN_H  -I /class/ece428/libs/include/thrift/ -L /class/ece428/libs/lib/ -c Something.cpp -o something.o
g++ -Wall -DHAVE_NETINET_IN_H  -I /class/ece428/libs/include/thrift/ -L /class/ece428/libs/lib/ -c Something_server.cpp -o server.o
g++ -Wall -DHAVE_NETINET_IN_H  -I /class/ece428/libs/include/thrift/ -L /class/ece428/libs/lib/ -c your_thrift_file_types.cpp -o types.o
g++ -Wall -DHAVE_NETINET_IN_H  -I /class/ece428/libs/include/thrift/ -L /class/ece428/libs/lib/ -c your_thrift_file_constants.cpp -o constants.o

g++ -Wall -DHAVE_NETINET_IN_H  -I /class/ece428/libs/include/thrift/ -L /class/ece428/libs/lib/ *.o -o Something_server -lthrift
```

Then you can run the server:

./Something_server

Similarly, you can build the client and run it, which will cause the server to execute the ping() function and "printf("ping\n");"

More advanced types supported by Thrift are http://thrift.apache.org/docs/types/ and http://svn.apache.org/repos/asf/thrift/trunk/tutorial/tutorial.thrift

I will have extra office hours on Friday if you need help with using Thrift.

> Then you can run the server:
>
> ./Something_server

You need to do:

LD_LIBRARY_PATH=/class/ece428/libs/lib ./Something_server

Or you can edit your shell's init scripts (e.g., .bash_profile/.bashrc) to add /class/ece428/libs/lib to LD_LIBRARY_PATH (e.g., for bash, "export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/class/ece428/libs/lib") and you can the programs normally.