Project 3 Group Project
Implementing IM Application

1. Overview:

This is a group project. You are allowed to form a group of up to 4 students. It is not recommended to be taken as an individual project.

2. Get Started:

The design of your IM system is an important aspect of this assignment. We provide a functional description of an IM system that your design should conform to. However, we will leave the precise details of the design up to you (maybe you want to make it state driven, using an Finite State Machine design, or use other techniques, maybe you want to add bells and whistles beyond what we outline all up to you). But please think about the design before diving into coding it will help. Feel free to look at some of the IM references presented at the end of this assignment, on the web, existing specs (MSNP always cite sources, no use of any existing code base of course but you can look at other open source code for sure, for inspiration, ideas, etc.). Be a detective. If you use any ideas, techniques, please cite sources, again.

When you do begin to implement your IM system we recommend you start with a basic set of features and proceed incrementally and keep it simple. Perhaps start with just two clients exchanging messages and build up from there. Test each feature as you proceed, build confidence, and keep working towards the design specification that you developed.

3. Blueprint for a Simple IM System:

Many of you are familiar with IM from a user's perspective but may not have given much consideration to its operation. The precise operation of an IM system will vary between different systems (e.g., AIM, MSN, iChat) but the essential functions remain the same. For the purposes of this assignment we require your simple IM system to incorporate the following functionality:

1) IM clients are able to join and leave the IM network. Every client has a user name and at any one particular time no two clients connected to the IM network can have the same user name. So some name checking is required in your code.

2) When IM clients join the network other connected clients are made aware of this client join within the network. Similarly, the reverse should be true; that is, when clients leave the network other connected clients should be informed. So you need notification and state management of a consistent view.

3) Connected IM clients are able to send messages to any other connected IM client. Messages are addressed to particular IM clients using their user name.

Given that constructing an elaborate IM system would be a quarter-long project we have limited the scope of this assignment in comparison to a full blown IM system such as supported by MSNP. It is perfectly acceptable to adopt the following simplifications:

1) A command line user interface is acceptable.

2) You are not required to maintain buddy lists specific to each IM client. It is fine to assume everyone in the IM network is part of a single universal buddy list. For instance all IM clients should be notified when any IM client joins or leaves the network. Furthermore, all IM clients should be able to exchange messages with any other IM client.

3) Your design only needs to support a single IM server. Although this is not a scalable approach it simplifies the task of building your IM system. You may assume all clients are aware of this single well-known IM server. So your server will sit on a known host and have a known port. Clients simply connect using sockets. By only having a single server you do not need to be concerned with maintaining consistent state between multiple servers, so essential in a real system.

However, feel free to provide any additional functionality if you wish, it will be rewarded as extra-credit. But completing the base line system with no extras would be outstanding. See how things go.

The objective of this assignment is for you to design and implement an IM system that conforms to the description outlined above. It is a fairly high level description, so the details are up to you. Supporting these features will require you to define an application level protocol that controls how participants in the system communicate. We suggest you adopt the client/server approach when designing your IM system.

The server should perform operations, such as:

(a) monitor clients that join and leave the IM network;

(b) maintain a reasonably consistent view of clients currently connected to the network and provide this state information to all the currently connected clients; and

(c) facilitate the exchange of messages between clients.

The clients should perform operations, such as:

(a) display to “the user” the currently connected IM clients; and

(b) allow “the user” to send and receive messages to and from any of these clients, respectively.

4. IM Design Spec:

Here is what we would like to see other than running code and a working simple IM system. We would like you to provide as part of the submission a short specification. We would expect a spec of 3 or 4 pages max that describes your protocol.

The spec should include:

1) A high level description of the types of participants within your system (i.e., IM clients, the IM server, and perhaps others). Describe what are the functions that each of these participants perform and how these combine to produce the required system behavior.

2) Define the protocols that occur between these participants. You should provide illustrative examples of the protocols in operation (such as seen in the textbook and class). You may find an FSM diagram useful in representing the state and operation of your system.

3) Provide a comment on reasons behind any important design decisions so we can best understand your thinking. You could even add a short section on ideas you jettisoned because of x, y, z reasons. What brought you to your final design? Could be you ran out of time and hacked it! Well, that would not be a great design methodology but industry sometimes operates like that when developing prototype ideas to test the waters against crushing deadlines.

During the design process you will determine that your server needs to be "multi-threaded" so that it can concurrently service more than one IM client at a time multithreading might be a new concept to you; it allows a software process to deal with concurrent events and operations. (We believe Project #2 has provided a great opportunity for pthreads practices.) Your IM system requires threads to concurrently access state that is shared between all thread instances. To ensure the correct operation of your server you will be required to perform some form of synchronization between these threads when they access shared data they need mutual exclusion. The pthread library provides support for threading and the necessary synchronization primitives. So this assignment brings in a lot of concepts that are important, one being shared state and the issue of concurrent access. You will come across these ideas in the software industry as designers and programmers.

5. Submisson (per team):

- Design Spec (**hard copy only**): Due Monday, 3/4, 3:40 PM
- Program: Due Tuesday, 3/12, 23:59 PM
- Pack your files into a tar package, named project3.tar, which includes Makefile, all source files, README file.
- Run the submission command in cs1.seattleu.edu
  /home/fac/zhuy/CPSC4560/submit p3 project3.tar

6. Grading:

- Failures in compilation and/or execution result in a score of zero point for software.

- Grading will be based on software, design spec, and demo (20 points).
- Demo will be based on the program you have submitted.
- Extra bonus of 5 points will be rewarded only if
    - Your additional functionality works properly and makes sense from design perspectives
    - Your required basic functionality works great

7. References:

1. Instant Messenger Systems
2. **http://libyahoo2.sourceforge.net/ymsg-9.txt (**Yahoo Instant Messenger Protocol **)**
3. **http://www.hypothetic.org/docs/msn/ietf_draft.txt (**MSNP (Early Internet Draft) **)**
4. **https://computing.llnl.gov/tutorials/pthreads/ (**POSIX Threads Programming **)**