Homework 7

DUE DATE: Thursday April 22nd 2010 @ 11:59 pm

The objective of this assignment is to help you get comfortable with some basic networking capabilities.

Implementing a chat server
As part of this assignment you will be writing a Chat Server. Clients will communicate with each other via the chat server. The assignment DOES NOT require you to provide support for forums or access control lists.

Chat Server

The chat server is required to listed to communications over both TCP and UDP. The TCP port over which the server will accept incoming socket connections is 4567. The port over which the chat server will accept incoming UDP datagram packets is 4567.

When a chat server receives message it will **broadcast** the message to all the clients that are registered with it. The only client that will not be part of this broadcast list is the client that originated the message in the first place. So if bob, alice and eve are connected to the chat server and bob originates a message, the message is broadcast only to alice and eve.

Client

Communications with this chat server will be using TCP or UDP depending on the option that is specified. The client has to perform the following functions

1. **Initiate** communications with the chat server based on the specified communications protocol.
   1.1. TCP: In the case of TCP you will be establishing a socket connection with the server.
   1.2. UDP: In the case of UDP, you will be sending a datagram packet to the specified host:port pair. The server is expected to respond to this connect message indicating that it did get the connect request. Otherwise, you have no way of knowing if either the host/port was incorrectly specified.

2. **Register** the client with the chat server. This would involve specifying a name during the registration process. The chat server will reject attempts to register another client with the same name. For example, if you are registering "bob" with the chat server, and there is a "bob" that is still actively registered with the chat server then the chat server should return an error message to the client.

3. **Accept** messages from the chat server: The chat client should accept and print out all messages published by the chat server.

4. **Send** messages to the chat server: Once a client has been registered, the client is ready to send messages to the chat server. The client should disallow attempts to send messages to the chat server if the client has not been previously registered with the chat server. The client has to append "username: " to any message that is being sent. For example, if the message being sent is "Hello world!" from a client that was registered as "bob", the message actually sent to the chat server is bob: Hello world!

5. **Deregister** from the chat server: This indicates that the client is no longer interested in sending/receiving messages from the chat server. Upon receipt of a deregistration request the chat server has to
   5.1. Send a message to the client indicating that the request was received and will be processed
   5.2. Garbage collect any resources (termination of socket connection) that were created to facilitate communication with the client in question.

6. **List**: When this command is sent to the chat server, the chat server returns a string containing the list of all registered users in the chat system.

Example of executing commands at your client:

This assumes that your chat-server is running on `bach.cs.colostate.edu`

connect hostname port transport-type

Example 1: **`connect bach.cs.colostate.edu 4567 udp`** :This establishes a UDP link to the server, the client should receive a message indicating that there is indeed a chat server at this host/port that is accepting datagram packets.

```
connect bach.cs.colostate.edu 4567 udp
Chat-server Response: Received connect request over Datagram Socket 4567
```

Example 2: **`connect bach.cs.colostate.edu 4567 tcp`**: This establishes a socket connection to the server. The server should send a message to the client confirming the establishment of the socket connection

```
connect bach.cs.colostate.edu 4567 tcp
Chat-server Response: Received connect request over TCP Server Socket 4567
```

register username

e.g. **`register bob`**: This register's the user bob with the chat server. If there is a user bob that is actively registered with the server, the server should respond with an error message describing the problem.

```
register bob
Chat-server Response: Register user bob with the chat server
            OR
Chat-server Response: Problem with registration request. There is another
active user bob that is connected on this server.
```

send text

e.g. send Hello world! This command sends a message to the server. The client should append "username: " to the message before it is sent out to the chat server.

Deregister username

e.g. **`deregister bob`**: This deregisters a user bob from the chat server. Upon receipt of this message the chat-server should send a response-message indicating that it is processing the deregistration request. The chat-server also garbage collects any resources set aside for the deregistered user's link.

```
deregister bob
Chat-server response: De-registeration request received. Request will be
serviced immediately.
Socket connection to bach.cs.colostate.edu:wxyz terminated
```

List

This lists all the active users on the chat-server.

```
list
The following is the list of active registered users at the chat server
Tom
Bob
Alice
```

Grading

The assignment will be worth 7.5 points towards your course grade. The assignment in itself will be graded on 15 points. The points distribution is as follows.

Support for exchange of messages between client and chat-server
3 points for TCP, 3 points for UDP {Total 6 points}
1 point for each command {Total 6 points}
3 points for handling the various error conditions

You are required to work alone in this assignment. Check the course website on late policy.

What to submit

All of your source files and a Makefile in a tarball or a zip. Follow usual naming conventions.