3780: Project

Guidelines This project is to be done in teams of two. Please submit a single file. Demos are in the last week.

Objectives To build a messaging framework over the Internet.

Requirements Sockets library, please refer to the tutorial available at `http://www.cs.rpi.edu/~moorthy/Courses/os98/Pgms/socket.html` for details of the system calls. Linux machines are best suited for the job.

1 Introduction

There are two parts to this project. The first part is to implement a client server model for exchange of messages between clients, server simply acts as an intermediary. For the first part a single server may be assumed. For scalability, reliability, and other reasons we want to have multiple servers. The second part concerns the exchange and update of database of clients amongst the servers using one of the routing protocols from the textbook. It might be convenient to write `C` code. You may use `Python`, `C++`, `Ruby`, `Go` or any other programming language that you are comfortable with.

2 Client Server Model of Message Exchange

The goal is to build a messaging framework. Client server model will be used. The clients communicate with each other via the messaging server. In this simplified model; client A wanting to communicate with client B, will pass on a suitably constructed message M to the server S. The server will keep the message M, and when client B sends a GET request to the server, all the messages destined for client

B will be sent to client B as UDP packets. We will not worry about authentication. The identity of the messaging server is known to all the clients. Upon receipt of the message M, client B will sent an ACK back to client A using the server S. The communication uses unreliable datagram service. Please note that the tutorial uses reliable TCP service, and appropriate changes will have to be made to the code to provide UDP service. You may assume that each client is identified by a unique 10 digit number.

The suggested message format is as below.

Seq No	Type	Source	Destination	Payload

- The fields are as follows:

  1. Seq No, Sequence number of the message.
  2. Type, SEND for sending a message to the Destination, GET for receiving all the messages destined to the clients. ACK for sending an acknowledgement back.
  3. Source, identity of the client A
  4. Destination, identity of client B
  5. Payload, the message

- Choose appropriate sizes for the fields in the message above and create a `C` style `struct`.

- Implement the client server model using sockets.

  1. The server while in listen mode, waits for messages from clients. Depending on the message, the server will take appropriate action. For instance if the message is `send`, then the server will store the message. Upon receipt of ACK from the client, the server may delete the message. If the message is `get` then the server will look at the stored messages, and the send the messages to the client (only the ones which are for the client though). You might want to use the `fork()` function.

2. Clients have to create a socket, connect to the server, and send and receive data from the server. Clients will send `ACK` packets back upon receipt of the message.

- Test the server and the client by running the code on two different machines.

3 Multiple Servers

Suppose there are two servers (A,B) and five clients (1,2,3,4,5). Clients 1,2 are with server A, and clients 3,4,5 are with server B. If client 1 wants to send a message to client 5, then it sends a suitably constructed message to server A. The issue is that server A does not know that 5 is served by B, and the message should be routed to B. To facilitate this both A, B maintain a list of the users that they serve, and exchange their lists periodically (as users may leave or join). One can use DV routing protocols for routing updates. Once A has the user list for server B, it can infer that any message for 5 should be send to server B. Given a message if a server cannot determine the location (server) to forward the message to, then it simply hangs onto to the message. Stated other wise, upon receipt of a `SEND` message the server computes the address of the server which handles the client listed in the destination field of the message, and forwards the message to the server (again as a UDP message).

You goal is to implement a routing protocol (DV or some other) for exchange of user lists at each server. You may assume that there are five servers, and the topology of the servers is fixed. Suppose the servers are numbered 1,2,3,4,5. Server `i` exchanges routing messages with servers `i-1` and `i+1`. The IP addresses for the servers are known and static.

Test the five servers and the clients by running the code on different machines.

4 Bonus

For additional marks (out of 5), can you implement the above system on Android devices using Bluetooth sockets.