Assignment 3 (Simple Chat Client-Server)

Due: January 27, 2023, 2 pm

A load balancer is a device which allocates incoming client requests to a set of servers so as to balance the load between them. In our example, the load balancer monitors the load of the servers periodically and when a client request comes, gives it to the server with the least load. In this assignment, you will write a very simplified version of a load balancer. Since we do not have real loads, we will use a dummy load for now.

In our system, there are (i) 2 no. computation servers named S1 and S2, (i) one load balancer named L, and (ii) a set of clients that connect for service. S1 and S2 will communicate only with L (and not with any client), and the clients will communicate only with L (and not with either S1 or S2). The functionalities of the different entities above are as follows:

1. S1 and S2: They wait to receive a request from L. If it receives a request to send its current load, it generates a random integer between 1 and 100 (a dummy load) and sends it to L; it also prints a message saying "Load sent: <x>" where <x> is the load value sent. If it receives a request for a client service (see description of L and clients below), it fulfils the service and sends the result back to L.
2. L: It periodically (every 5 seconds) asks both S1 and S2 for their load by sending a message "Send Load". It stores the received load as the last known load of the servers in a table, and closes the connection (so each such load request happens with a fresh connection, 5 seconds is a long time). When it receives a load value, it also prints "Load received from <server IP> <x>", where <server IP> is the IP address of S1 or S2 whose load is being asked, and <x> is the load value received. It also waits for receiving client requests for service. The only service supported now is request for date and time. If any client requests for service, L first finds which server among S1 and S2 has reported lower load last time it probed, and sends the string "Send Time" to that server (opening a new connection to the server). It also prints a message saying "Sending client request to <server IP>" where <server IP> is the IP of the server the request is sent to. The server, on receiving this string, sends the date and time back to L. L, when it receives the date and time, closes the connection to the server, and then sends the date and time back to the client.
3. The clients just connect to L. A connection is assumed to be request for a service and it waits to receive the date/time from L, after which it closes the connection. So this client is going to be the same client you wrote for Problem 1 in Assignment 1.

Note that L acts as server to the clients and as client to S1 and S2.

Design the connections you will need to create and tear down. You can assume that not more than 2 clients will connect to L at the same time.

S1 an S2 should be TCP iterative servers (so they are minor modifcations of the time server you wrote in Assignment 1). L should be a TCP concurrent server. L should use only 1 socket to wait for all client requests.

Submit 3 files: server.c (for S1 and S2), lb.c (for L), and client.c (for clients). Each of server.c and lb.c must take the port no. they will run on as a command line parameter (if you hardcode it, two copies of server cannot be run).