CS 171 : Introduction to Distributed Systems
Programming Assignment 1
Due: Monday April 16, 2018

Online ticket booking services for entertainments such as movies, plays, concerts, etc., have increased more than ever. In this project, you are to design a ticket vending site for entertainments. To simplify the model, the site is required to support only two kinds of entertainment: movies and plays. In other words, users can request either to buy a specific number of movie tickets **or** play tickets. Every user goes to a *kiosk* to buy the tickets.

1 Backend servers

Your ticket vending system should consist of two servers, say S_m and S_p. Both servers start off with 50 tickets. S_m keeps track of the number of available movie tickets and S_p is in-charge of play tickets. Every request to buy movie tickets is handled by S_m and every request for plays is handled by S_p. Each request to the server specifies the type of tickets, movie or play and the specific number of tickets they want to buy. When a server receives a request for the type of tickets it maintains (movies or plays), it either has enough tickets to sell, in which case the server sends a successful reply, or there are not enough tickets to sell, in which case the server sends an unsuccessful reply. A kiosk can send a request to **either** server. If S_m gets a request for buying movie tickets, it handles it locally, but if it receives a request for play tickets, it forwards the request to S_p (and vice versa). In both cases, the server that received the request from the kiosk should be the one to reply back.

2 Application Component

Simulate kiosks (TCP clients) that take input from the user and send the request to the backend servers. Kiosks are oblivious as to which server is responsible for movies and that for plays. Each kiosk sends requests **randomly** to one of the servers. The request will be of the form

{buy: <movies, n>} or {buy: <plays, n>}

where n is the number of tickets. The kiosk then displays whether the request was successful or not.

3 User Interface

NOTE: We do not want any front end UI for this project. All the processes (servers and kiosks) will be run on the terminal and the input/output for these processes will use stdio.

1. When starting a server, it should connect to the other server. You can provide the servers IP, or other identification info that can uniquely identify each server. Or this could be done via a configuration file or other methods that are appropriate.

2. All kiosks are aware of the two server IPs. This information can be read from a configuration file or other methods that are appropriate. Each kiosk process prompts the user if they want to buy movie or play tickets and the number of tickets. The kiosk then randomly picks one of the servers and sends the request to that server. It then shows the reply from the server.

3. You should log all necessary information on the console for the sake of debugging and demonstration, e.g. Message received from kiosk XX. Message sent to kiosk YY. Messages exchanged between servers. Log when number of tickets changes. Etc,.

4. You should add some delay (e.g. 5 seconds) when sending a message. This simulates the time for message passing and makes it easier for demoing concurrent events.

5. Use message passing primitives TCP/UDP. You can decide which alternative and explore the trade offs. We will be interested in hearing your experience.

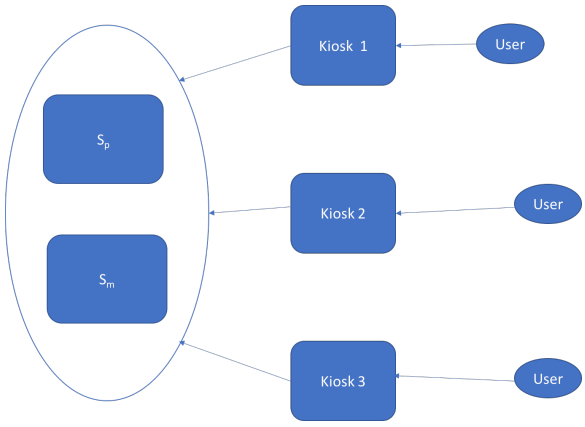

Figure 1: Example with 2 servers and 3 kiosks

4 Demo

For the demo, you should have 2 servers, one each for serving movie and play tickets. You should have 3 to 5 kiosks, prompting the user to buy tickets.

We will have a short demo for each project. For PA1, the demo will be on **Monday, 04/16** between 2-4 PM in CSIL. For this projects demo, you can deploy your code on several machines. However it is also acceptable if you just use several processes in the same machine to simulate servers and kiosks.

5 Example

Fig. 1 provides a overview of what is expected from this assignment. The number of kiosk shown is 3 whereas you can design the system for upto 5 kiosks. Some points to be noted:

- There is going to be one user per kiosk at a time.
- Kiosk can send the request to any of two servers randomly.

6 Teams

Projects should be done in team of 2. You can use Piazza to form teams.