ICS 167

Project 2: Multi (2) - player Pong

Project Requirements:
For this project, each team will be required to develop a 2-player client-server pong game. You may use any game engine you wish to develop the pong clients, and you may wish to use any programming language to develop the server for the game. We recommend that you develop the clients in Unity, as it is a good beginning game engine platform and the server in either C++ or C#. Unity has a physics engine that can be used to capture the movement and bouncing (collision with the walls and the paddles) of the ball.

Clients:
The Pong clients have to communicate with the server their status, i.e., both clients must update the position of their paddle to the server. Ball movement is calculated locally on the clients (client prediction). In order for the game to be playable in the presence of delays, all parties need to be synchronized (use of global time from NTP daemon, ntpd). More on this topic will be discussed as time goes by. Knowledge of time warping (at server), dead reckoning (opponent prediction on clients) will be required and have to be coded to make the game playable.

Because the clients and the server are all co-located within the ICS domain, the actual delays will be very low. You will have to simulate a client delay of 50ms and 200ms respectively at client 1 and client 2. Delays can be simulated by holding packets in a buffer for that amount of time before releasing for transmission. The buffer needs to be implemented at both the client and the server (two way delay).

Server:
The server must synchronize the game between both clients. The motion of the ball must be programmed on the server, which maintains game state, thus, the server will know the position of the ball throughout the course of the game. The server will determine whether a player hit the ball with its paddle or whether the player missed. The server keeps track of the game score and broadcasts it to the clients. You have to determine what score is required for winning the game. The server starts the ball in motion at the center of the board and sends a packet to the clients indicating the start time of motion, the direction and the velocity of the ball. You can determine, based on who last scored, what direction the ball will take.

Logistics:
Below is an example of what, at the bare minimum, the game board and game control should look like. As the designer, you can embellish the game with different choices of ball speed, ball size, texture, paddle angle control, sound effects, colors, variable delays drawn from a distribution, etc. Extra credit will be given for these embellishments; the main grade though will be on a playable fully synchronized game in the presence of

delays. To show effect of delay and your compensation algorithms, you will have to demonstrate the game with and without latency compensation in the presence of simulated delay. Recommend that you design and develop the game first with no latency, and once the logic and communication is working, you insert the delay and incorporate the compensation techniques.