Sean Kennedy

HW1 – Design of Network Application Updated

Example of a pick-up basketball application

The objective is to design and implement a network application that correctly simulates n games of pick-up basketball.

There will be one server that contains information about the players avaialable to be drafted and the score counter for the n games played. The server will keep track of team name, what client drafted what player (in the multiplayer version) and also their overall roster and total win count.

An issue that needs to be addressed is if the same player is attempted to be drafted twice. To ensure this does not happen, the server must synchronize data after every draft selection. When a player is drafted, they must be removed from the list of available players and not be seen as an option anymore by the next one to draft.

Another synchronization issue that could come up is that the draft does not properly rotate between selections (meaning that the picks are not going back and forth properly, but instead maybe 2 picks for one team consecutively). I plan to address this issue with checks to make sure that one team's player count does not go 2 players above the other's. This will add a small extra bit of complexity to the application, but will be worth it to avoid a major issue. I also added the constraint that a number smaller than 0 could not be selected when picking a player, as well as a number above size list.size()-1. This got rid of that second issue entirely.

Another issue to address is that the server may prompt for a team name every time a new game is started during one session. Another similar issue is that the score count will be reset to 0-0 after each game from one session. This is a potential data integrity problem. One simple solution would be to store win counts as static variables to ensure they are not changed when they should not be.

The type of messages passed by the server will mostly be prompts to the player, to which the player(s) will respond. For example, here is what a draft will look like (singleplayer version):

Server: Please select a player (Available Players: Player X, Player Y, Player Z, Player A, etc.)

Client: Player Y

Server: Selection registered; it is now my turn to select

Server: I select Player X

Server: Please select a player (Available Players: Player Z, Player A, etc.)

.....

The Client tag will be replaced by whatever team name that the player enters before the first game. These tags help differentiate who is saying what, which is useful in the draft process where there is a lot going on.

Since basketball is a two-team game, the multithreaded version will be limited to only two clients who represent one team each.

Minimalist Version:

The minimalist version of this application is a single player simluation against the server. Players will enter a team name and then draft players from the available roster, trading picks one by one with the server (who is also drafting from available players list). Once the 5 man rosters are assembeled, a game will be simulated. After displaying the result of the game and incrementing the score counter of whoever won, the server will prompt the player to play again. If they say yes, their team name and updated score count must be maintained to the next instance of the game.

client

Enter team name

loop (draft) (while teamSize < 5)

Is state DRAFTING?

Yes - Choose one of the available players for your team

Increment team size

No - Wait for server to select and return the updated list of players

Server sends updated team rosters

See result of simulated game

See updated game score

Server: "Would you like to play again?"

Yes - Start again from the draft phase

No – Exit

server

Prompt for team name

loop (draft) (while teamSize < 5)

Server: "Welcome to the draft!"

Server: "Initial roster: " + list full roster

Is state DRAFTING?

Yes - Choose one of the available players for your team

Increment team size

Change state to WAITING

No - Wait for client to select and then get the updated list of players

Once updated list recieved, chnage state to DRAFTING

Show the full team rosters

Simulate the game randomly

Use the score generated to determine winner

Display result

Update game score

Prompt to play again

Yes - Start again from draft phase

No - Return final game score and determine overall winner

Enhanced Version:

The enhanced version of the application will allow for a multiplayer game to be played between two clients. Instead of drafting against the server, players will be drafting against each other, with the server's new job being to just act as a mediator between the two players. Players must know who the other one drafted each round so that they can select from the updated list of available players. Also, player's will be able to chat when it is not their turn to draft. I hope to accomplish this using different states (DRAFTING, WAITING). These new additions can potentially lead to problems in both synchronization and data consistency if not implemented correctly.

client i

Enter team name

loop (draft) (while teamSize < 5)

Is state DRAFTING?

Yes - Choose one of the available players for your team

Increment team size

Can see messages from other client

No - Free to type messages to other client

Server sends updated team rosters

See result of simulated game

See updated game score

Server: "Would you like to play again?"

Yes - Wait for other client's response

Yes- Start again from the draft phase

No - Exit

No – Exit

server i

Prompt for team name from client

loop (draft) (while teamSize < 5)

Server: "Welcome to the draft!"

Server: "Initial roster: " + list full roster

Is state DRAFTING?

Yes - Recieve draft selection from client

Tell client that their choice was accepted if player is available

Send selection from other server

Recieve messages from client2 and display them for current client

Increment team size

Change state to WAITING

No - Wait for other client to make a selection

Allow client to send in messages

Send client's message to client2

Once other client makes selection, change state to DRAFTING

Show the full team rosters

Simulate the game randomly

Use the score generated to determine winner

Display result

Update game score

Prompt to play again

Yes - Wait for response from other client

Yes - Start again from draft phase

No - Return final game score and determine overall winner, display winner and exit

No - Return final game score and determine overall winner, display winner and exit