CS F303 : COMPUTER NETWORKS

MINI PROJECT EC-2

on

"Simulation of multilevel card game (Hearts) over TCP network"

PROGRESS STATUS DOCUMENT

By

Group 6

Pranjal Gupta	2013B4A7470P
Rishabh Jain	2013B3A7473P
Neel Kasat	2013B3A7670P
Akshit Bhatia	2013B3A7722P

April 7, 2017

1. Problem Statement

The software will be the Conceptualization, Specification, Design and Implementation of the famous card game called Hearts. This game can be played among a group of 4 users(clients) and will be co-ordinated by a server over a TCP network. The software also extends the basic version of the game with increasing difficulty levels which introduces subtle changes to the game rules with its progress. The game can be played by 4 players which will connect to a host server. The host server synchronizes and alternates between the user(client) activities. Each client should communicate to the server with the current state of its player. The client will prompt its user to take an action (e.g. throw a card) and will communicate the user action to the server. The server should check if the move from a user (through client) is feasible and legit, and consequently continues the same with other clients. The server should also keep scores and communicate the actions of other client to a client. The client shall have a basic user interface which will display scores and actions of other users, as required.

2. Status of progress

S.No.	DESCRIPTION	MODULE	STATUS
1.	BLUEPRINT OF ALL COMMUNICATION BETWEEN SERVER AND CLIENT	1	**Completed.**
2.	CLIENT-SERVER ARCHITECTURE : BASIC BUILD UP (WITH SERVER HANDLING MULTIPLE CLIENTS)	2	**Completed.**
3.	SPECIFIC FUNCTIONS FOR CLIENT AND SERVER FOR IMPLEMENT-ATION OF GAME LOGIC	3	**Completed.**
4.	ACTION SYNCHRONIZATION BETWEEN CLIENTS AND SERVER	4	Under process.
5.	SHARED MEMORY FOR CLIENT SERVER INTERACTION AND BASIC USER INTERFACE	5	Yet to begin.
END SEMESTER DEMO AND SUBMISSION			

3. Perceived Challenges

- **Handling Multi-client concurrent connections:** In the game, we had to connect 4 clients to one server and connect them to play one game. So, we created a child process as soon as we get a group of 4 clients connected to the server. So now, for every group of 4 clients, a new game room will get initialized.
- **Problem in consecutive send/receive between same server-client pair:** We were having data inconsistency issues on sending two consecutive messages or receiving two or more consecutive messages. So, in order to solve it, we added a sleep of 1 time unit between any two consecutive send/receive commands between a same pair of client-server.
- **Action synchronization between client and server:** To maintain the smooth flow of the action sequence to be taken between the server client pair, we were having a problem. So later, instead of sending the data, we resorted to keeping a copy of data with the client, and sending signals to the client. And, as per the signal sent, the client will use the appropriate data.
- **Information synchronization using temporary buffers:** In Level 2, we need to transfer two cards to the other player. We will store the cards in a temporary buffer at the server side. A random number, say n, between 1-4 is then generated. Now, every player transfers 2 cards to the nth player in the sequence beginning from that player.

4. Design Details:

The game will require proper synchronisation between the turns of players, handling connection drops. Highlights of the game are as follows :

a) 4 users have to play at a time on different machines.
   i) 4 players will have to connect to the server. If more than 4 players request for a connection, a new game room will get created for other players, in multiples of 4.

b) Each player is distributed cards randomly.
   i) The cards have been numbered from 1 to 52, where first set of 13 cards being of Clubs, followed by Diamond, Hearts, and Spade.
   ii) The cards are printed along with the unicode symbols of the deck.

c) Score of each user, each player move, handling game levels
   i) Score of each user is maintained on the server. The card set is also stored on the server as well as on the respective client. This has been done to ensure fair gameplay on the side of server.
   ii) As one level ends, the user gets upgraded to play the next level. At the end of the three levels, the overall winner is declared.

d) Each user has to wait for the user to complete his turn

i) As per the signalling, each player has to wait until his chance comes. He will be updated as per whose chance is going on, and what card he/she has played.

e) Levels and Rounds
   i) There are 3 levels, with increasing level of difficulty, with each level having 13 rounds. The rounds progress sequentially, with each player waiting for 3 other players to play their turn and playing its own turn in a pre-defined order. At the end of each round, the server update points scored by all the players in a particular round, also total cumulative score till that round.
   ii) Scores are maintained at server for each round and levels consequently. The scores are sent to clients at the end of each round.
   iii) The server also ensures fair play in game. The client checks if the card played by the player is a valid move based of the previous cards played by other players. This is cross checked at server. The server also notes the sequence of cards played and the collection of cards at hand for each player.

f) Error Handling
   i) The reliability of a software is purely dependant upon its ability to handle unforeseen situations efficiently.
   ii) The server responds to connection drops with clients and consequently signals other players about the drop.
   iii) Provisions are also made to cross out invalid signals while communicating, so as to ensure proper sequencing of events in the game play.

Game Room Formation:

Every client will have to connect to a particular port. The server waits for at least 4 clients to get connected. As soon as 4 clients are connected, a child process is spawned and it forms a game room. The game is played in the game room.

If more clients get connected, the parent process of the server again waits for clients in multiples of 4 and spawns child processes for that group to run the game room.

The game room further facilitates the playing of the main Hearts game. Every player will get a server generated player ID and a randomly allotted player name. Server makes use of the allotted player ID to cater to communication with all the clients

Server's sequence of actions :

The server's communication in the game room has been implemented as the sequence of signals to each of the client based on the state of play. With each signal, data is either received or sent based on the pre-defined action requirement. Each signal corresponds to a specific action in the game, which depends

upon the action of other clients in the game play. The table of signals (till the current project update), along with their corresponding action and data that needs to be sent or received, are as listed :

SIGNAL	ACTION	DATA (SENT/RCVD)
1	Connection established successfully over TCP	-
2	Game room closed. All players are allotted random names.	**SENT :** Sequence of player names in the room and id of the client.
3	Level 1 begins. Cards are distributed to all the players in the room	**SENT :** 13 cards for each player
4	Beginning a round of play	**SENT :** Round number
5	Prompts the client to play the card.	**RCVD :** The index number of the card played by a particular client
6	Wait while the other player plays his turn	**SENT :** id of the player currently playing his turn
7	Prompts that the current player has played his turn	**SENT :** Index of the card played by the other player
8	Updates score after each round	**SENT :** Scores of all the players in the game
9	Level 2 begins. Cards are distributed	**SENT :** 13 cards for each player
10	Prompts the player to select 2 cards that the client wants to swap with other player	-
11	Requests to send the selected cards	**RCVD :** Comma-separated index of 2 cards to be swapped.
12	Sends the swapped cards to the user.	**SENT :** Comma-separated index of 2 cards.

** We have yet not implemented the Level 3 of the Game. Certain other signals based on server-client synchronising will be added later*

Client Model :

The client's action sequence has been modelled in the form of Deterministic Finite Automata (DFA), wherein each state represents a specific progress in the gameplay. The state transitions are only possible

in the event of a “valid” signal from the client. Any other “invalid” signal is treated as an unwanted deviation from the normal play, and is reported as an error. A DFA diagram with state descriptions are as depicted below :

Client-side DFA diagram

(upto LEVEL 2)