HORSE RACE GAMBLE

A project by Liel Sinn 209195155 and Roee Afriat 208499137

☒ Summary:

We will have a gambling station for horses' race.
Then, the server asks each client over TCP connection on which horse he would like to put money and the amount.
The moment the server accepted two legal (or more) clients, the race begins, and over different UDP session the server updates all clients by multicast for the current state of the race.
At each point of time, each client can choose either to double the amount of money he invested or to quit the gamble. In the latter case the server announces the quitting client that he would get only 50% refund and ask him if he is sure to do so and wait for his answer.
At the end of the race a winning / loosing message will be sent to each participant, and the prize will be split among the winners equally.

☒ Managing several clients simultaneously:

In order to serve several clients simultaneously, the server will use two sessions with each client, one over TCP and one over UDP (with select() function, the UDP will be multicast transmission), each one will be implemented over different thread.
In the thread of the TCP the client will select() between the session with the server and some input from the keyboard (for quit / double money), while the server will only select() between all clients' sessions.
In the thread of the UDP the server will manage a matrix, in which each row will represent a path of a horse and the number of columns will represent the length of the race path. In each row we will have one '*' which will represent the location of this horse on the path. We will also drill a horse which we will increment him by one step and multicast this matrix to all clients. In the client UDP thread we will wait for transmissions of the matrix and print it to screen.

☒ Communication protocol:

In each type we have the same message structure:

The message is char[256], type is 1 char, all variables are 4 chars (they will contain int).

There are 9 types of messages:

Type 0: hello message. The server sends it with the multicast address (in variable 1) and the client sends an ack back with just the message type.

Type 1: choose horse message. The server sends it with the number of horses in variable 1and the client sends back with the chosen horse in variable 1.

Type 2: choose money to gamble message. The server sends it with the type only and the clients sends back with the chosen amount of money in variable 1.

Type 3: quit game message. The client will send it with type only and the server will send back with amount of money to be returned to quitting client in variable 1.

Type 4: double money message. The client will send it with type only and the server will send back an ack with type only as well.

Type 5: data request message. The client will send it with type only and the server will send it back with variable1 – number of horses in race, variable 2 – total money in the bet and in later variables the amount of money that was bet on each horse.

Type 6: winning message. The server will send it with the winning prize in variable 1, no ack will be sent from client to server.

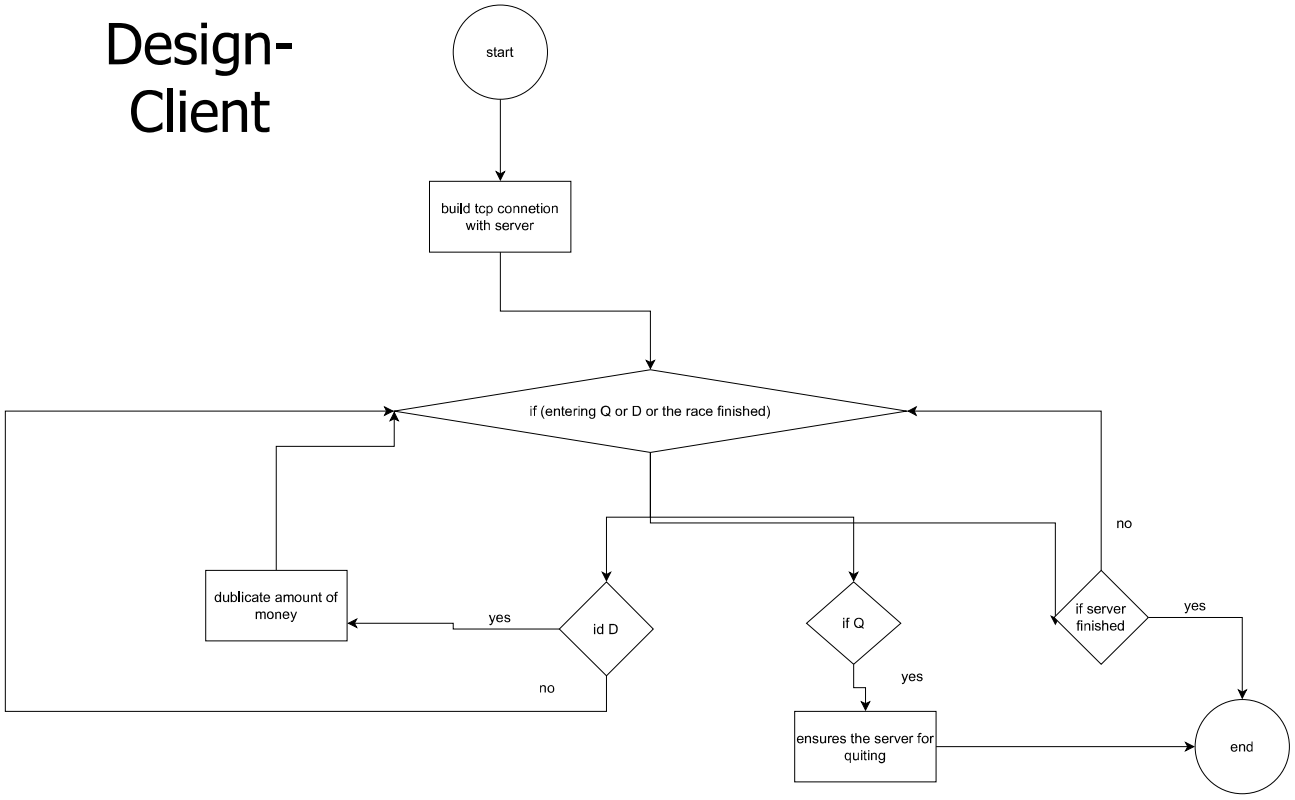
Type 7: loosing message. The server will send it with type only, no ack will be sent from the client.

Type 8: disconnect message. The server will send it with type only, no ack.

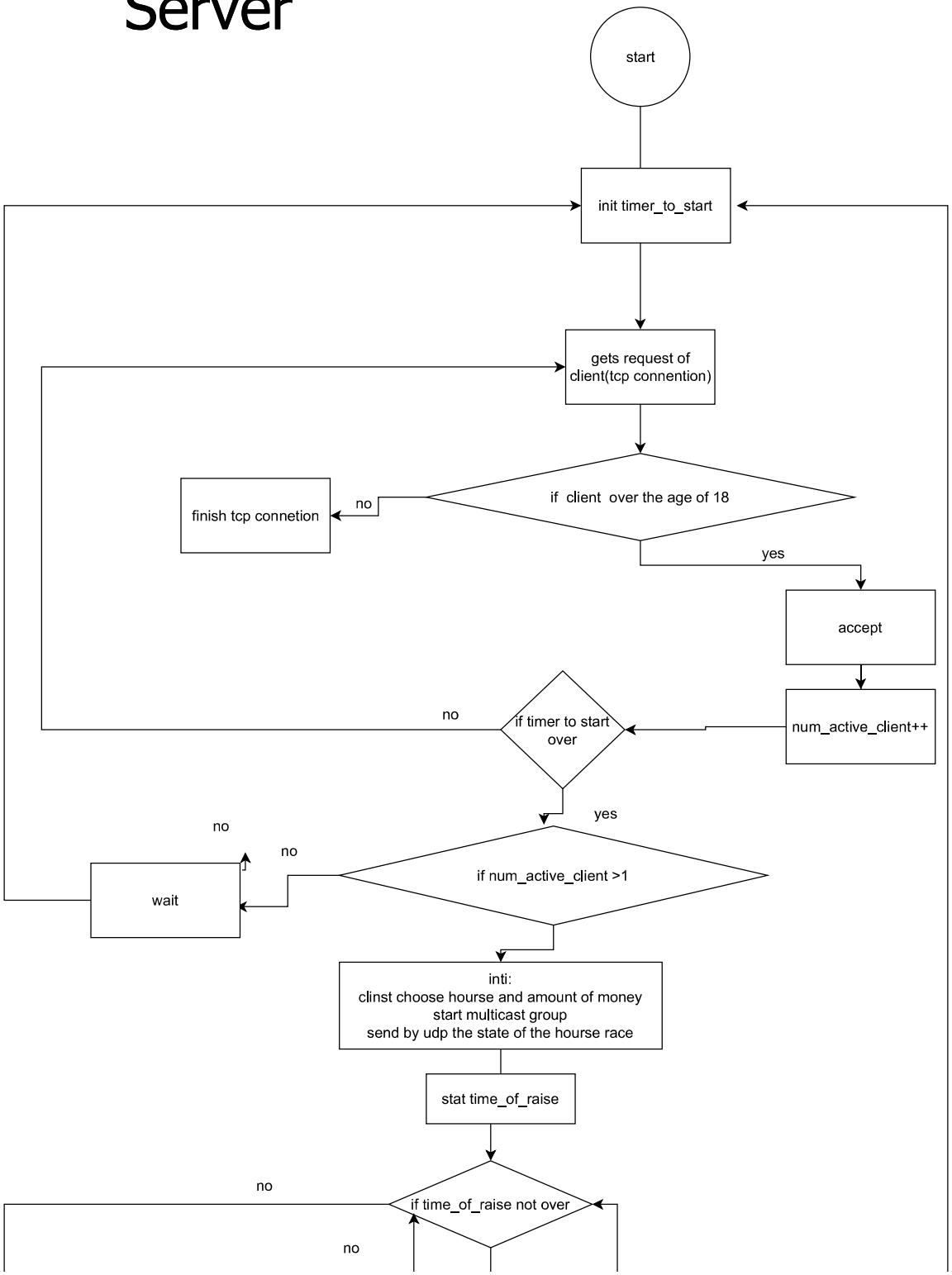
☒ Test the program:

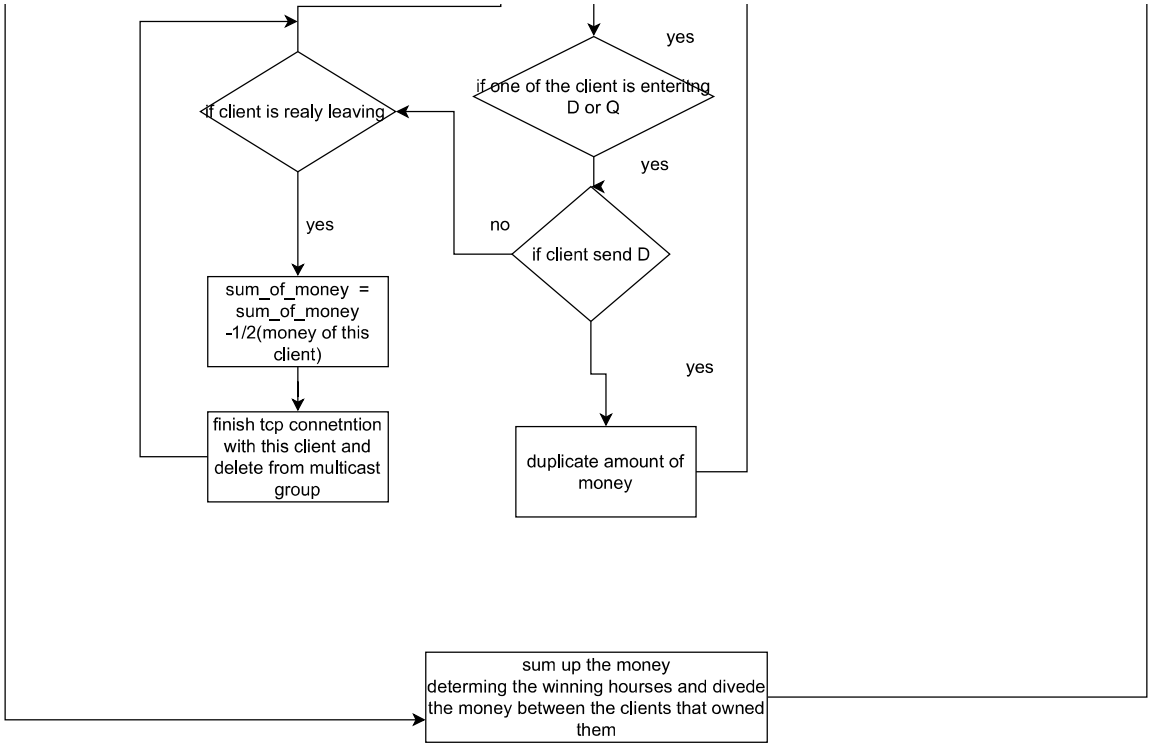
*in case the client waits to the race to begin, we will ignore its input from keyboard.
*in case the client sends the server invalid horse num the server will disconnect him.
* in case the client sends the server invalid amount of money to bet on the server will disconnect him.
*in case the server will send the client a message in type he shouldn't get right now, the client will disconnect himself from the server.
*in case the server didn't get the correlate type of message as an ack to a message he just sent, he would disconnect the client.

Design-Client



Design-Server




yes
if one of the client is entering D or Q
yes
if client is realy leaving
yes
no
if client send D
sum_of_money = sum_of_money -1/2(money of this client)
yes
finish tcp connetntion with this client and delete from multicast group
duplicate amount of money
sum up the money determing the winning hourses and divede the money between the clients that owned them