Dynamic Tic-Tac-Toe Game using Client-Server Architecture in C Programming

Krishna kumar k

Department of Information Technology
National Institute of Technology Karnataka,
Surathkal ,Mangalore, India 575 025,
krishnakumar.211it034@nitk.edu.in

Prasanna Kumar

Department of Information Technology,
National Institute of Technology Karnataka ,
Surathkal ,Mangalore, India 575 025,
pkrb.211it047@nitk.edu.in

Ekank Chhaparwal

Department of Information Technology,
National Institute of Technology Karnataka,
Surathkal ,Mangalore, India 575 025,
ekankmaheshwari.211it019@nitk.edu.in

Abstract— This report presents the implementation of a Tic-Tac-Toe game using C and a client-server architecture that involves two players and a server. The game allows two players to connect to a server and play against each other in a dynamic environment. The client-server model was chosen for this implementation as it allows for the game to be played over a network and enables two players to connect and play simultaneously.The game's server handles all of the game's logic and manages the state of the game. It listens for incoming connections from clients and assigns them unique IDs. The clients interact with the server by sending and receiving messages, such as moves, game status updates, and chat messages. The server updates the game state and sends it to the clients, who then update their game boards.

Keywords— Sockets, Client-server architecture, Tic-Tac-Toe, MultiThreading, Game Board. Socket API

I. Introduction

Tic-Tac-Toe, also known as Noughts and Crosses or Xs and Os, is a simple game which is popular among all ages. The game is played between 2 players , who take turns marking the spaces in a NXN grid. The player who succeeds in placing N of their marks in a horizontal, vertical, or diagonal row is the winner. The game can be played on a piece of paper or a board, but it can also be implemented on a computer.
Tic-tac-toe is a game of perfect information, meaning that both players have complete knowledge of the game state at all times and can take their turns accordingly. The game's user interface was implemented using the C programming[9] language. The interface is simple, yet elegant, and allows players to easily make moves and view the game's status. It also illustrates the effectiveness of using a client-server model in game development, enabling two players to play together[10] over a network in a two player game mode.

II. SOCKET PROGRAMMING

Socket programming is a way of connecting two nodes on a network to communicate with each other. One socket(node) listens on a particular port at an IP, while the other socket reaches out to the other to form a connection. The server forms the listener socket while the client reaches out to the server.
A socket[7] is created on each end of the communication link, one for the server and one for the client. The server creates a socket and binds it to a specific port number, and then listens for incoming connections from clients. When a client wants to connect to the server, it creates a socket and connects to the server's socket using the server's IP address[2] and the port number.
Once a connection has been established between the client and server, the two parties are able to send and receive data using the functions provided by the socket API. This data transfer[3] is done using TCP/UDP Protocol.

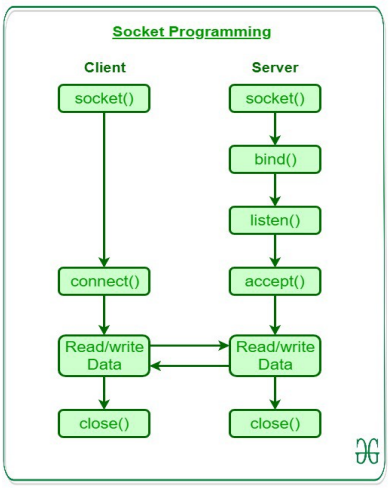

Fig 1. State diagram for server and client model of Socket

III. Multithreading in socket programming

One way to handle requests from more than one client is to make the server program multi-threaded. A multi-threaded[5] server creates a thread for each communication it accepts from a client.

A thread is a sequence of instructions that run independently of the program and of any other threads.Using threads, a multi-threaded server program can accept a connection from a client, start a thread for that communication, and continue listening for requests from other clients.

This allows the server to handle multiple requests at once. In a Tic-Tac-Toe game that uses socket programming, multithreading can be used to create separate threads[8] for each client connection, allowing them to play the game in parallel without interfering with each other. On the server side, each thread would handle the communication with its respective client and update the game state accordingly. Additionally, multithreading[8] can also help to prevent the server from becoming overwhelmed with requests and ensure that the game remains responsive to the clients.

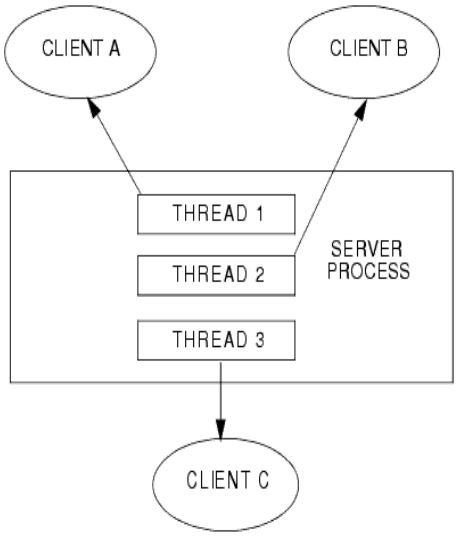

Fig. 2 Block Diagram of the Multithreading in Socket Programming

IV. server Side socket programming in tic-tac-toe game

In a Tic-Tac-Toe game using socket programming with a client-server architecture, the server handles all of the game's logic and manages the state of the game. The server needs to be able to handle multiple client connections at the same time, and one way to achieve this is by using multithreading[6].

Here are the some of the main functions that the server handles in order to handle multiple client connections and play the Tic-Tac-Toe game:-

A. Listen function: The server needs to listen for incoming connections from clients and wait for them to connect. This is typically done using the listen() function provided by the socket API.
B. Accept function: Once a client has connected, the server needs to accept the connection using the accept() function. This function will return a new socket descriptor that will be used for communication with the client.
C. Thread creation: The server creates a new thread for each client connection, so that each client can play the game simultaneously. ‘‘pthread.h’’ is the library used in C Language to handle Multithreading[5].
D. Game Board Size : Based on the Integer Value provided as an argument while running the server side C program, the game board gets initialized to [10] that size of Matrix.
E. Game management: The server needs to keep track of the state of the game and update it as necessary. It includes Initializing the game state, including the game board, Loop through the game, alternating between the two clients and allowing them to make their moves.When a particular client makes a move, the server should update the game state and send the updated game state to both the clients.
F. Communication: The server needs to be able to send and receive messages to and from the clients. This includes sending game status updates, messages, and move information to both the clients using read() and write() functions. If one of the clients wins or the game is a draw, the server should send a message to both clients indicating the outcome of the game and close the socket descriptors.

V. client Side socket programming in tic-tac-toe game

The client side socket program will be responsible for[1] rendering the game board and allowing the player to make moves alternatively. It would use the socket API to create and manage the socket, and it would use the send() and recv() functions to send and receive data with the server.

As soon as the game starts the initial table will be displayed[7] for the convenience of players and all the intermediate tables in between will also be shown on the client's side.

The following functions and features are employed here to make the player experience smoother:-

A. connect() function : It is used in socket programming to initiate a connection with a server. This function is typically used by a client to establish a connection to a server. The connect() function takes three arguments: the socket descriptor, the address of the server, and the size of the address. Once the connection is established, the client can then use the socket to send and receive data from the server. The connect function returns 0 on successful connection and -1 if the connection fails.
B. Game Board Size : Based on the Integer Value provided as an argument while running the Client side C program, the game board gets initialized to that size of Matrix.
C. Game management: The client will be able to make a move in the NXN Grid by entering a number from 0 to N*N - 1,for N*N unique positions in the grid matrix. .When a particular client makes a move, the client receives the game state and updates from the server.
D. Communication: The client is able to send and receive messages to and from the server. This includes receiving game status updates, messages, and sending the move information to the server using read() and write() functions through socket API
E. Game Status : The client who is able to succeed in placing three of their marks in a horizontal, vertical, or diagonal row will be receiving the message from the server about their winning status of the game.

VI. Running the dynamic tic tac toe game

Requirements : The user can play the tic-tac-toe game on LINUX/UNIX [4] based operating system. The client.c and the sever.c file needs to be saved under the same folder.Open three different terminals side by side for running the server,client1,client2 of the socket program.

A. Running the Server Side Program : On the terminal type the following command to run the gcc file, "gcc -pthread server.c -o server" , and "./server 5552 N" to execute the server.c file. While executing the[7] file, the portNo and the Integer key need to be given as an argument to run the server side program. The integer key decides the board game size. Whenever a player is connected to the server using socket API. An acknowledgement message is shown on the terminal about the number of players being connected to the server. When 2 Players are connected to the server an empty game board is displayed of size NXN stating that the game has been started as shown in Fig.3

B. Running the client side program : On the terminal type the following command to run the gcc file, "gcc client.c -o client" , and "./client "Systems IP address" 5552 N" to execute the client.c file. While executing the file, the IP address, port-No. and The Integer key needs to be given as an argument to run the client side program. The integer key decides the board game size, portNo is used to connect the server which is using the same port-No.
After Successful Connections an Acknowledgement is received and the basic rules of entering the spaces in the Grid is shown to the user as shown in Fig.4

The client side program needs to be run in two different terminals since the Tic-Tac-Toe is a game of Two Players [9].

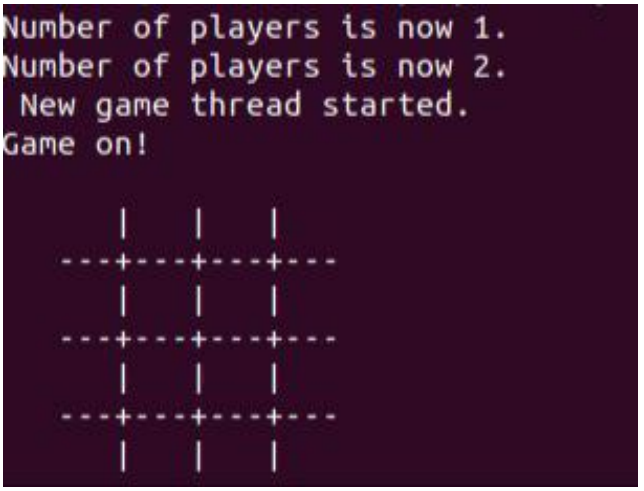

Fig 3.Successful connection -accept of both the Client with the Server.

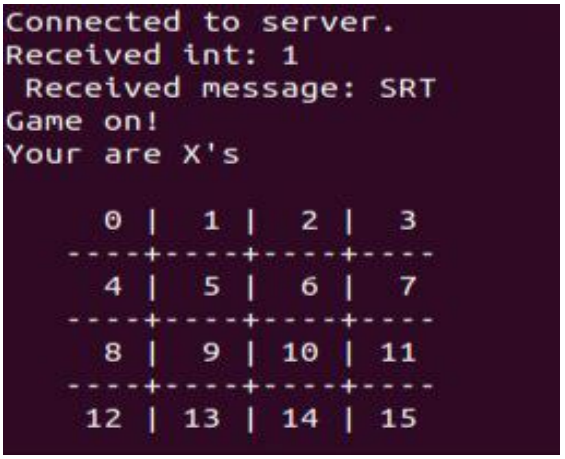

Fig 4. Successful connection of the Client- with server

The game runs under a while loop until the winner is declared or one of the clients was disconnected due to some technical glitch. Each Player is given a chance alternatively to fill the Game Board by entering the Index of that Space Matrix.

The server handles the Invalid Moves made by the clients and sends an error message to that client.

At the end when one of the players satisfies the rules [9] of the Tic-Tac-Toe Game, The server sends an Update Message to both the clients about their winning status as shown in Fig. 5 and Fig. 6

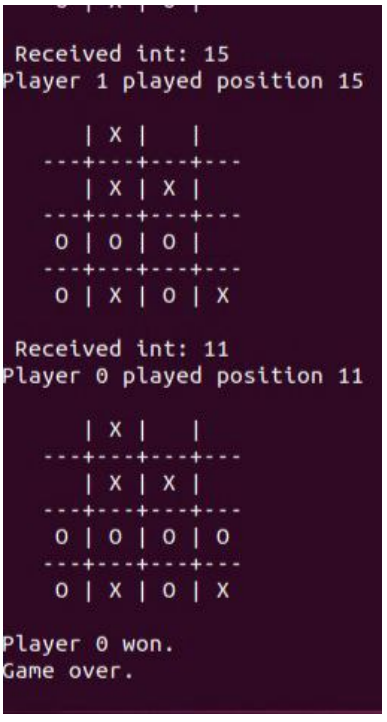

Fig 5.Displaying Player-0 Winning the Game

Fig.6 Displaying Player-1 Losing the Game

VII. Conclusions

The Tic Tac Toe game is most familiar among all the age groups.Intelligence can be a property of any purpose-driven decision maker. This basic idea has been suggested many times. Logic for playing Tic Tac Toe game using Client-Server architecture in C Programming has been presented and tested that works in an efficient way. It successfully demonstrates the use of socket programming in creating a multiplayer game. The game allows two players to play Tic Tac Toe over a network and provides real-time updates and communication between the players. The project highlights the importance of socket programming in network communication and can serve as a foundation for creating other multiplayer games or network applications.Overall the system works without any bugs.

Acknowledgment

We would like to acknowledge the contributions of Professor Geetha V in the development of this Tic Tac Toe game. We also thank Teaching-Assistants who mentored, guided and supported us throughout the project. This game utilizes socket programming to enable N*N dynamic tic tac toe functionality, and we are proud to have successfully integrated this technology into the final product.

References

[1] Olafsen, L. (1998). I/O server-client tra c generator A performance analysis.

[2] Oltean, S. E., Abrudean, M., & Dulau, M. (2006, May). Remote monitor and control application for thermal processes using TCP/IP. In 2006 IEEE International Conference on Automation, Quality and Testing, Robotics (Vol. 1, pp. 209-213). IEEE.

[3] Chawdhury, M. D. A., & Habib, A. A. (2008, December). Security enhancement of MD5 hashed passwords by using the unused bits of TCP header. In 2008 11th International Conference on Computer and Information Technology (pp. 714-717). IEEE.

[4] Wang, K. C., & Wang, K. C. (2018). TCP/IP and Network Programming. Systems Programming in Unix/Linux, 377-412.

[5] Berger, E. D., Yang, T., Liu, T., & Novark, G. (2009, October). Grace: Safe multithreaded programming for C/C++. In Proceedings of the 24th ACM SIGPLAN conference on Object oriented programming systems languages and applications (pp. 81-96).

[6] Yang, Y., Chen, X., & Gopalakrishnan, G. (2008). Inspect: A runtime model checker for multithreaded C programs. Technical Report UUCS-08-004, University of Utah.

[7] Zhang, H. (2013). Architecture of network and client-server model. arXiv preprint arXiv:1307.6665.

[8] Liu, T., Curtsinger, C., & Berger, E. D. (2011, October). Dthreads: efficient deterministic multithreading. In Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles (pp. 327-336).

[9] Karamchandani, S., Gandhi, P., Pawar, O., & Pawaskar, S. (2015, January). A simple algorithm for designing an artificial intelligence based Tic Tac Toe game. In 2015 International Conference on Pervasive Computing (ICPC) (pp. 1-4). IEEE.

[10] Ling, S. S., & Lam, H. K. (2011). Playing tic-tac-toe using a genetic neural network with double transfer functions. Journal of Intelligence Learning Systems and Application.