# Summary Document

**Gaurav CS5700**
**Northeastern University: The Roux Institute**

# Title: A Detailed Analysis of a Networked Tic-Tac-Toe Game Implementation

## 1. Introduction

The Tic-Tac-Toe game is a popular and simple game played by two players using a 3x3 grid. This networked version of the game, written in C, allows players to engage in the game over a network. The program uses socket communication to transmit game data between the server and the client.
This document describes a networked Tic-Tac-Toe game written in C, allowing two players to play the classic game over a network using sockets. The game can run as either a server or a client, depending on the command-line arguments provided. The server sets up a listening socket and waits for a client to connect. Once the client connects, both players take turns playing the game until there is a winner or the game ends in a draw.
The primary components of the Tic-Tac-Toe program include the main function, which determines whether the code should run as a server or client, the server and client functions that handle socket communication, the game logic, and utility functions for displaying the game board and checking the game status.

## 2. Overview of the Networked Tic-Tac-Toe Game

The game is designed as a client-server application, with the server hosting the game and the client connecting to the server to play. The application uses the Transmission Control Protocol (TCP) for reliable communication between the server and client

## 3. Game Logic

The core of the Tic-Tac-Toe game remains the same, with players taking turns to place their marks ('X' or 'O') on a 3x3 grid. The first player to get three marks in a row, column, or diagonal wins the game. If all grid cells are filled and no player has won, the game ends in a draw.

The implementation includes a function named **checkwin()**, which is responsible for evaluating the game board and determining the game status. This function is called after each move, and it returns one of the following values:

- 1: If a player has won the game
- -1: If the game is still ongoing
- 0: If the game has ended in a draw

The **board()** function is responsible for displaying the current state of the Tic-Tac-Toe game board. It prints the board in a human-readable format, showing the positions of each player's symbols ('X' or 'O') and the available moves.

## 4. Main Function

The main function is the entry point of the program. It takes command-line arguments to determine whether the program should run as a server or a client. When running as a server, it requires the **-s** argument, while running as a client requires the **-c** argument, followed by the server's port number and optionally, the server's node name (IP address or hostname). Based on the provided arguments, the main function calls either the server or client function to start the game.

## 5. Client Function

The client function is responsible for connecting to the server and playing the Tic-Tac-Toe game as a client. It starts by gathering the server's node name and port number from the command-line arguments, creating a socket, and connecting to the server. Once connected, the client function begins the game by sending the user's chosen symbol ('X' or 'O') and turn (1 or 2) to the server. The client function then enters a loop, in which it sends and receives data over the socket to make moves and update the game board.
During each iteration of the loop, the client function calls the **board()** function to display the current state of the game board, and the **checkwin()** function to determine the state of the game. If the game has ended, either in a win or a draw, the loop is exited, the connection is closed, and the program terminates.

## 6. Server Function

The server function is responsible for setting up a server and waiting for a client to connect. It initializes a socket, binds it to a specific port, and listens for incoming connections. Once a client connects, the server function plays the Tic-Tac-Toe game with the client, sending and receiving data over the socket to update the game board and make moves.

The server function also uses a loop to manage the game state, calling the **board()** function to display the current state of the game board, and the **checkwin()** function to determine the game's state. When the game ends, the loop is exited, the connection is closed, and the server function returns.

## 7. Utility Functions

The application also includes utility functions, such as the **error()** function, which handles error messages. In case of an error, this function prints the error message to the standard error output and exits the program with an error code.

## 8. Compiling and Running the Code

To compile the code, we have to execute the Makefile. The following command can be used to run the Makefile:

./make
This command generates an executable named **ttt**. To run the server, execute the following command:
./ttt -s

To run a client, execute the following command:
.
./ttt -c [server_number] [server_node]

## 9. Summary and Conclusion

The Tic-Tac-Toe game written in C is a networked application that allows two players to engage in the classic game over a network. The program uses socket communication to transmit game data between the server and the client. The main function handles command-line arguments to determine whether the program should run as a server or a client and calls the appropriate functions accordingly.
The client and server functions manage socket connections and the game loop, utilizing the **board()** function to display the game board and the **checkwin()** function to determine the game's state. Utility functions, such as the **error()** function, help with error handling and ensuring the program exits gracefully in case of an issue.
Overall, this Tic-Tac-Toe game demonstrates a simple but effective example of networked game development using the C programming language and sockets. The application showcases essential programming concepts, such as socket communication, command-line argument handling, and game logic implementation,

making it a valuable learning resource for anyone interested in network programming and game development.