



Green University of Bangladesh

*Department of Computer Science and Engineering (CSE)
Semester: (Fall, Year: 2022), B.Sc. in CSE (Day)*

Tic-Tac-Toe Using TCP/IP Socket Programming in Java

*Course Title: Computer Networking Lab
Course Code: CSE 312
Section: 211-D3*

Students Details

Name	ID
Md.Anas Khan	201902037
Nura Anha Tamanna	202902002

*Submission Date: 30.12.23
Course Teacher's Name: Md. Rafiqul Islam*

[For teachers use only: **Don't write anything inside this box**]

<u>Lab Project Status</u>	
Marks:	Signature:
Comments:	Date:

Contents

1	Introduction	2
1.1	Overview	2
1.2	Motivation	2
1.3	Problem Definition	3
1.3.1	Problem Statement	3
1.3.2	Complex Engineering Problem	3
1.4	Design Goals/Objectives	4
1.5	Application	5
2	Design/Development/Implementation of the Project	6
2.1	Introduction	6
2.2	Flow Chart	6
2.3	Implementation	7
3	Performance Evaluation	9
3.1	Simulation Environment/ Simulation Procedure	9
3.2	Results Analysis/Testing	9
3.3	Results Overall Discussion	13
4	Conclusion	14
4.1	Discussion	14
4.2	Scope of Future Work	14

Chapter 1

Introduction

1.1 Overview

In the early 1970s, online multiplayer games started emerging. Initially, it was LAN-connected games. The moves made by the player are sent to a local server and are written back to the other player. The inter-system communication is done by “Socket programming”. Online multiplayer games have always been good entertainment and an interesting mode of communication. It connected various people across the world. This project focuses on the implementation of a Tic-Tac-Toe Game using Transmission Control Protocol (TCP/IP) Socket Programming in Java. Tic-Tac-Toe is a turn-based game, that combines the classic game of Tic-Tac-Toe with the power of network communication using TCP/IP sockets. In this project, we’ve created a multiplayer version of Tic-Tac-Toe where players can compete against each other over a network.

1.2 Motivation

- This project presents a valuable opportunity for educational exploration. By combining the principles of Tic-Tac-Toe gameplay with TCP/IP socket programming in Java, we aim to enhance our understanding of these fundamental concepts.
- The turn-based nature of Tic-Tac-Toe aligns well with the fundamentals of multiplayer game design. Each move can be transmitted between the server and the client, and the game state can be easily synchronized.
- Tic-Tac-Toe is universally known and accessible. Choosing a game that most people are familiar with ensures that users can easily grasp the rules and gameplay, making it inclusive and enjoyable for a broad audience.
- The primary goal of the project is to explore and implement TCP/IP socket programming for multiplayer interactions. By selecting Tic-Tac-Toe, the focus can remain on mastering the networking concepts, including data transmission, synchronization, and error handling, without getting entangled in complex game mechanics.

1.3 Problem Definition

1.3.1 Problem Statement

1. Designing a robust communication mechanism to facilitate multiplayer interactions between two or more players in real-time. This involves the transmission of player moves, synchronization of game states, and handling each game room's concurrent actions.
2. In TCP/IP socket programming, establish and manage reliable connections between the game server and multiple clients. This includes handling socket creation, and data transmission issues.
3. Need to ensure synchronization of the turn-based gameplay between players. Each move made by a player needs to be accurately communicated to the server and then relayed to the opposing players to maintain a consistent game state.
4. Implementing effective error-handling mechanisms to address scenarios such as network interruptions, disconnections, or unexpected behaviors.
5. Developing a scalable server-client architecture to accommodate multiple game rooms concurrently. The server should efficiently manage incoming connections, and allocate players to appropriate game rooms.
6. Implementing a fair and synchronized mechanism for determining the starting player through a toss.
7. Defining the conditions for winning the game or declaring a draw. This involves implementing the logic to check for winning combinations or a filled game board, signaling the end of a match, and prompting appropriate responses.

1.3.2 Complex Engineering Problem

The following Table 1.1 is completed according to our above discussion.

Table 1.1: Summary of the attributes touched by the mentioned projects

Name of the P Attributes	Explain how to address
P1: Depth of knowledge required	Moderate to High. Proficiency in Java, understanding of TCP/IP socket programming, and basic game development concepts.
P2: Range of conflicting requirements	Limited. The requirements are focused on networking and gameplay, with minimal conflicting elements.
P3: Depth of analysis required	Moderate. In-depth analysis is required for designing the communication protocol and ensuring synchronization.
P4: Familiarity of issues	Moderate. Familiarity with game development and basic networking concepts is beneficial.
P5: Extent of applicable codes	Moderate to High. Implementation of socket-related code, game logic, and GUI components is substantial.
P6: Extent of stakeholder involvement and conflicting requirements	Low to Moderate. Stakeholders may include developers, and players. Limited conflicting requirements.
P7: Interdependence	High. Tight interdependence between networking components, game logic, and the graphical user interface (GUI).

1.4 Design Goals/Objectives

The objectives of this project are as follows:

- The primary goal is to facilitate seamless communication between players, ensuring that moves are transmitted accurately, and the game state is synchronized for all participants in each game-rooms.
- Implement effective error-handling mechanisms for network interruptions and to ensure a stable gaming environment.
- Create a responsive and visually representing graphical user interface (GUI).
- The design goal is to develop a scalable server-client architecture to accommodate concurrent game rooms.
- We also added intelligence to our project by checking for a win or a draw after each move. This means players experience the excitement of victory or the challenge of a draw after every strategic move.
- Implement a synchronized method for determining the starting player moves through a toss on each match.

1.5 Application

Students and developers can use this application to understand the implementation of multiplayer games, socket communication, and the integration of game logic with network functionality. Users can engage in friendly matches regardless of physical distance, fostering social interaction through a familiar and enjoyable game.

Chapter 2

Design/Development/Implementation of the Project

2.1 Introduction

Before delving into coding, careful consideration was given to the architectural design. The system was conceptualized as a server-client model, where the server manages game-rooms. where each game-rooms maintains the number of pairs of players in each game-room and after connecting each client to the server the game-room also checks if any game-room fills up or not. Each game-room needs 2 players to be ready to start the match. The server also manages the facilitates communication between players. This phase involved defining data structures, interactions, and the overall flow of the application.

2.2 Flow Chart

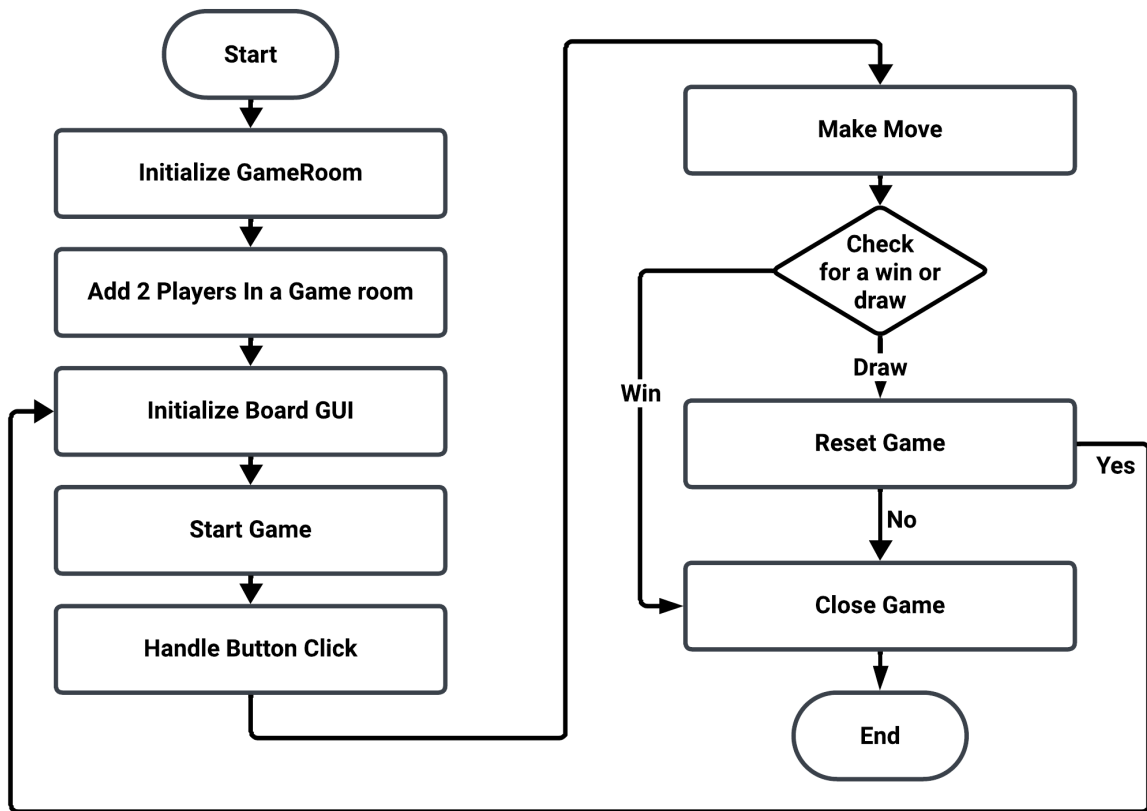


Figure 2.1: GameRoom Flow chart

2.3 Implementation

Here's an overview of our project:

1. **Server-Client Architecture:** The project adopts a server-client architecture where a central server manages the game room and facilitates communication between players. The server is responsible for listening to incoming client connections, maintaining the game state, and coordinating the flow of messages between players.
2. **Socket Programming Basics:** In Java, the `Socket` class is used for creating sockets, and the `ServerSocket` class is used by the server to listen for incoming connections. The `Socket` class facilitates communication by providing input and output streams for data transmission between the server and clients.
3. **Server Setup:** The server creates a `ServerSocket` and listens for incoming connections on a specified port such as port 8888. When a client wants to join a game room, it establishes a connection to the server using a `Socket`.
4. **Client Connection:** Clients initiate a connection to the server by creating a `Socket` object with the server's IP address and port number. The server accepts incoming client connections using the `accept()` method of the `ServerSocket`.

5. **Data Transmission:** Once a connection is established, communication between the server and clients involves sending and receiving data. Input and output streams `BufferedReader` and `PrintWriter` are used to read and write data over the sockets.
6. **Turn-Based Interaction:** The game follows a turn-based interaction model where players take turns and make moves. We've added a touch of synchronization to the game by implementing a toss for the starting player. Whether 'X' or 'O,' the synchronized toss sets the stage for each match.
7. **Error Handling:** Robust error handling is implemented to address potential issues such as disconnections, network interruptions, or invalid data. Exception handling in Java is utilized to manage errors and prevent application crashes.
8. **Threaded Model:** Multithreading is often employed to handle multiple client connections concurrently. Each client interaction is managed in a separate thread.
9. **Closing Connections:** Proper closure of sockets is essential to release resources and ensure the graceful termination of connections. The server and clients close their sockets when the game concludes or when a player decides to exit.

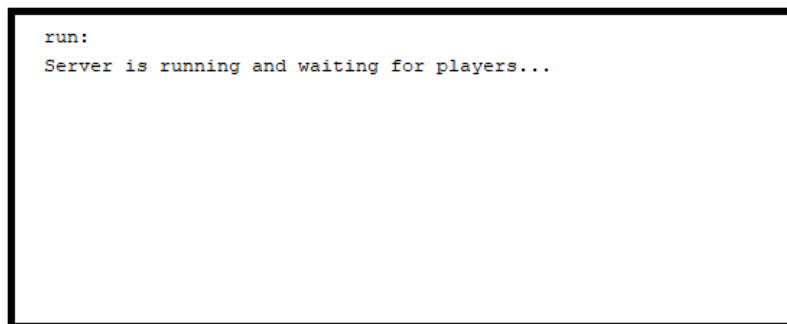
Chapter 3

Performance Evaluation

3.1 Simulation Environment/ Simulation Procedure

1. **Java Development Kit (JDK):** Need Java Development Kit (JDK) installed on a machine. Anyone can download the latest JDK from the official Oracle website.
2. **Integrated Development Environment (IDE):** Choose an IDE for Java development such as IntelliJ IDEA, Eclipse, or Visual Studio Code.
3. **Build the Project:** Use the build tool in the project such as Maven, Gradle to build the project.

3.2 Results Analysis/Testing



```
run:
Server is running and waiting for players...
```

Figure 3.1: Server Running

```
run:  
Connected to the Tic Tac Toe server!  
Wait for another player!
```

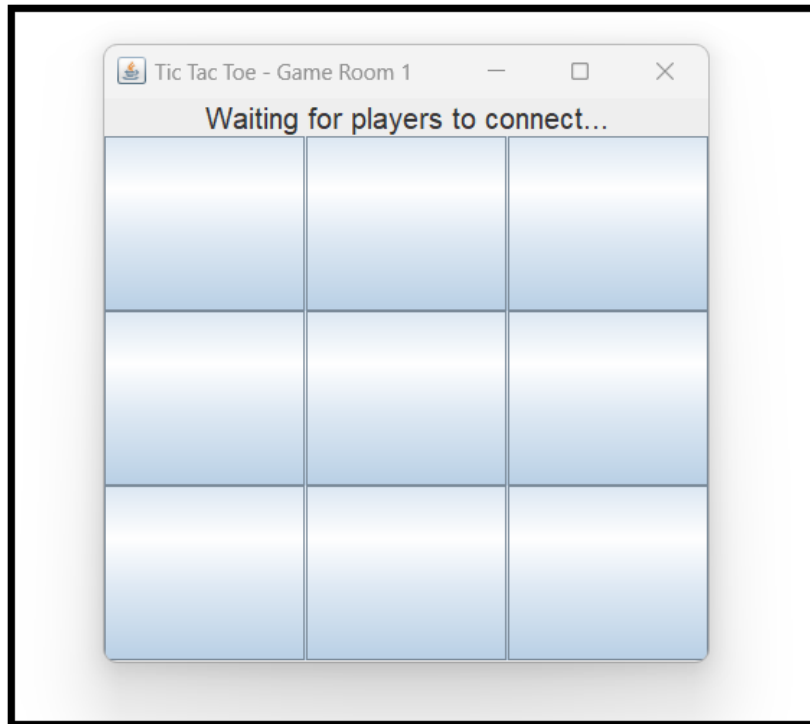


Figure 3.2: Player waiting

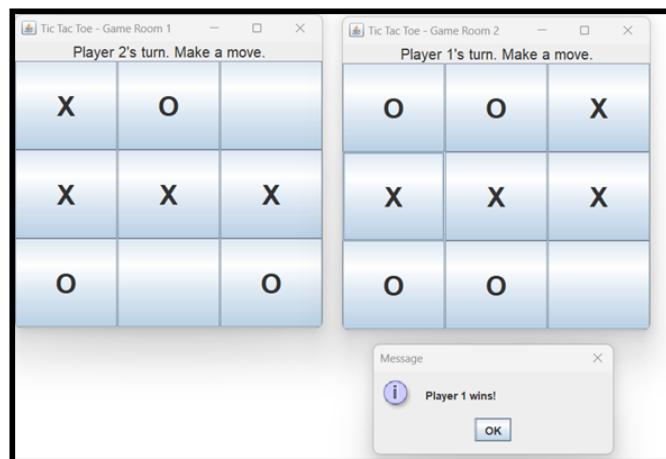
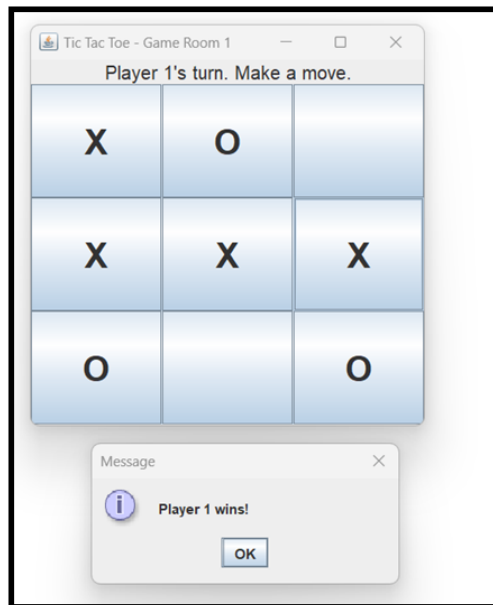


Figure 3.3: Start Match GameRoom 1 and 2

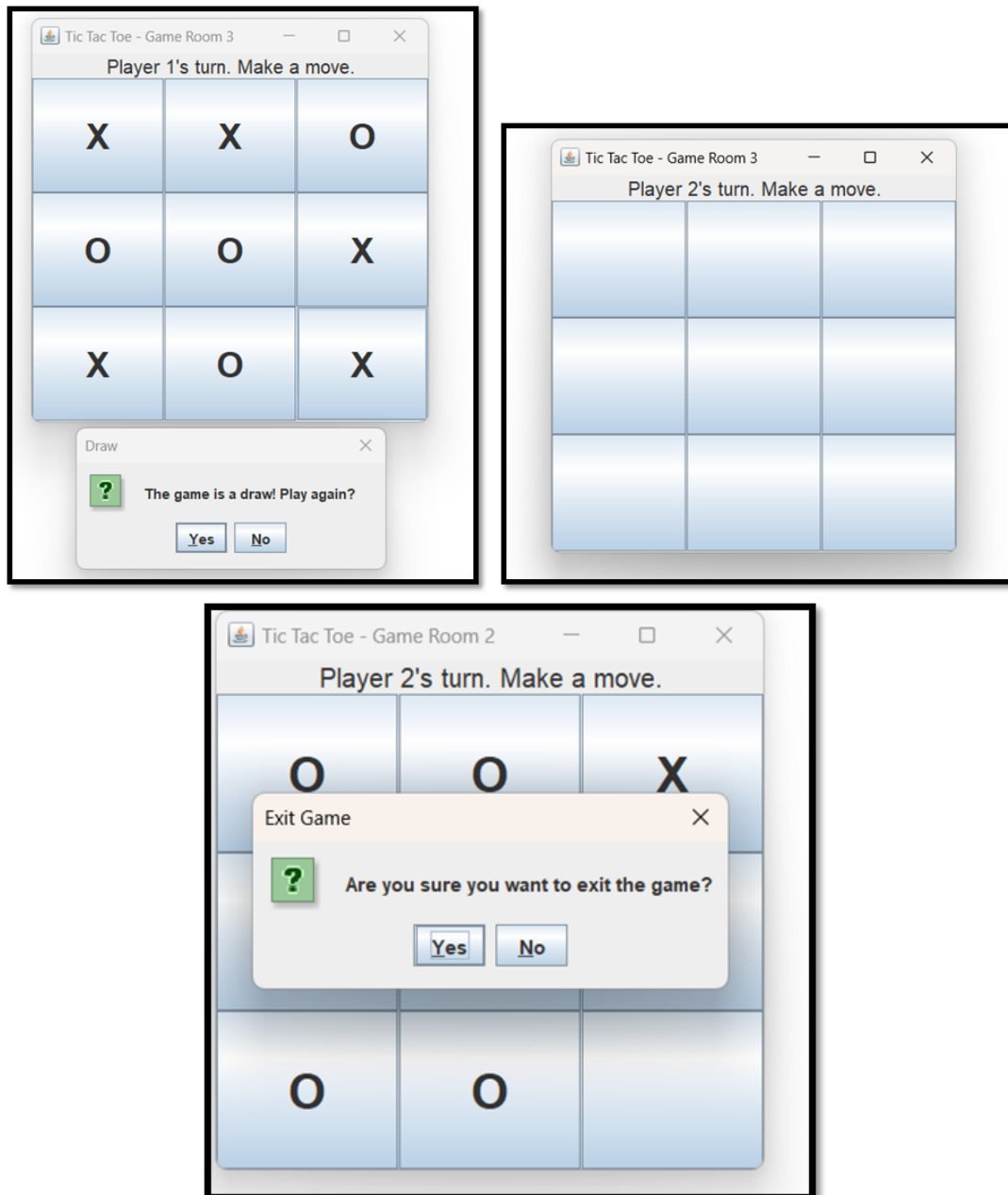


Figure 3.4: Draw Match,After clicked Yes, and Exit

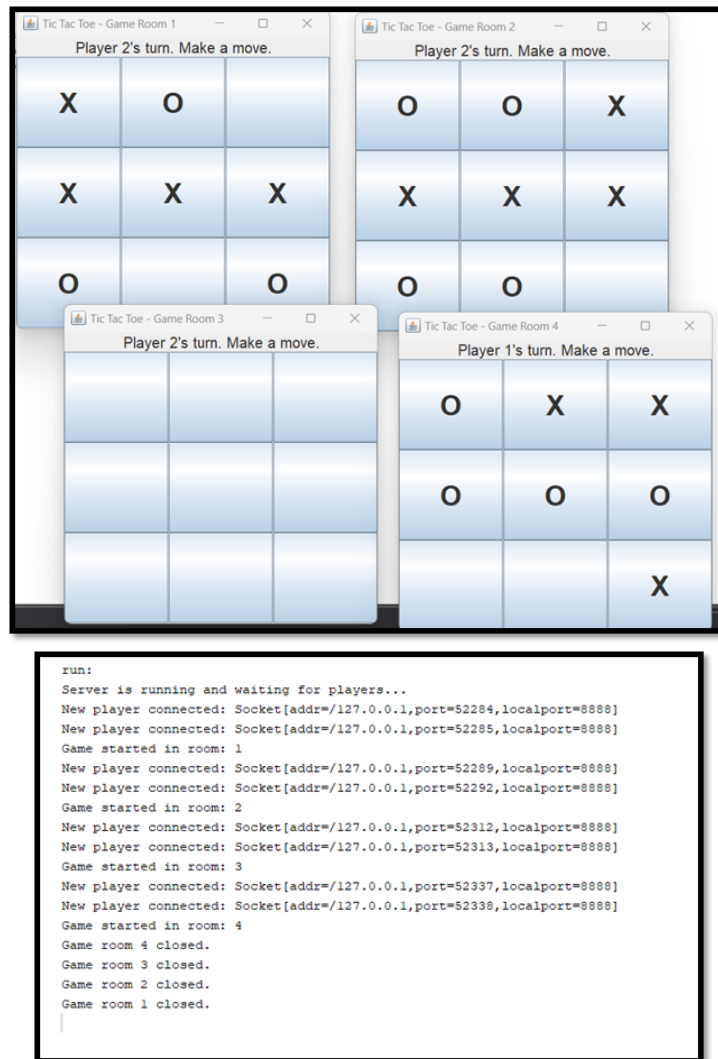


Figure 3.5: Total GameRoom, Server-Side

3.3 Results Overall Discussion

The results of the Tic-Tac-Toe project utilizing TCP/IP Socket Programming in Java reflect the successful implementation of a multiplayer game with network communication. The key aspects of the project include a functional server-client architecture, responsive GUI, and effective handling of player interactions.

Chapter 4

Conclusion

4.1 Discussion

In conclusion, the project has achieved its primary goals, demonstrating the successful implementation of Tic-Tac-Toe with TCP/IP Socket Programming in Java. The game's fundamental features, such as turn-based gameplay and player interaction, have been effectively implemented. The server-client architecture ensures seamless communication, allowing players to enjoy a classic game in a networked setting. The use of TCP/IP Socket Programming guarantees reliable and secure data exchange between the server and clients. The project's design minimizes latency issues, offering a smooth gaming experience with robust network communication. The multiplayer functionality successfully enhances the gaming experience, allowing players to connect, join game rooms, and engage in matches.

4.2 Scope of Future Work

The current Tic-Tac-Toe project lays a solid foundation for future enhancements and expansions. Several properties for future work can further enrich the gaming experience and extend the project's capabilities:

- Implementing a chat feature within game rooms would enhance social interaction between players during matches.
- Investing in improved graphics, animations, and visual effects can elevate the overall aesthetic appeal, making the gaming experience more immersive.
- Allowing players to customize the game board's theme or appearance adds a personal touch to each match.
- Organizing periodic tournaments or competitions among players enhances the competitive aspect of the game.
- Implementing a ranking system based on player performance adds a competitive element and motivates players to improve.

Bibliography

- [1] *Playing Tic-Tac-Toe with a Lightweight Robot*. Dégoulange, E., Dauchez, P.: External force control of an industrial puma 560 robot. J. Robot. Syst. 11(6), 523–540 (1994)
- [2] *Playing Tic-Tac-Toe With Robot On The Web In Real Time*. Yeung, K and Jacques, B and Du, R