Industry track

Student List:

Group Member	Student ID	Email	Chosen destiny
Hung Trinh	2307229	Hung.TRINH@student.oulu.fi	Corporate
Mazen Hassaan	2307227	Mazen.Hassaan@student.oulu.fi	Corporate
Umer Yaseen	2307645	Umer.Yaseen@student.oulu.fi	Corporate
Ida Haataja	Y69019019	ida.haataja@student.oulu.fi	Corporate

Project Title

Multiplayer Rock, Paper, Scissors Game

About the project

Course Project Overview

The project aims to design and implement a distributed system used for a multiplayer Rock, Paper, Scissors game. The software and system architecture aim to address key distributed system functionalities, including synchronization algorithms, resource naming and sharing mechanisms, and secure communication. The application involves smart IoT devices acting as game nodes, a central server managing game state and communication, and features request queuing and prioritization based on game conditions.

Project Objectives

- Design and implement a distributed system for a multiplayer Rock, Paper, Scissors game.
- Implement synchronization algorithms to ensure real-time gameplay between players.
- Develop resource naming and sharing mechanisms to manage game state and communication.
- Implement secure communication protocols to protect sensitive player data.
- Utilize smart IoT devices as game nodes to enhance gameplay experience and interactivity.

Expected Outcomes

- A functional multiplayer Rock, Paper, Scissors game accessible to players from different locations.
- Real-time gameplay experience with immediate feedback on match results.
- Efficient resource utilization and scalability through load balancing mechanisms.

- Secure communication channels to protect player data and ensure privacy.
- Demonstrated application of distributed system principles in building responsive and scalable gaming applications.

Implemented components:

Detailed description of the system architecture (Application-specific system components):

- System must have at least three nodes (e.g, containers)
- Each node must have a role: client, server, peer, broker, etc.

Participating nodes must:

- Exchange information (messages): RPC, client-server, publish/subscribe, broadcast, streaming, etc.
- Log their behavior understandably: messages, events, actions, etc.

Nodes (or their roles) do not have to be identical For example, one acts as server, broker, monitor / admin, etc. Each node must be an independent entity and (partially) autonomous

Detailed descriptions of relevant principles covered in the course (architecture, processes, communication, naming, synchronization, consistency and replication, fault tolerance); irrelevant principles can be left out.

Built with:

Detailed description of the system functionality and how to run the implementation

- If you are familiar with a particular container technology, feel free to use it (Docker is not mandatory)
- Any programming language can be used, such as: Python, Java, JavaScript, ..
- Any communication protocol / Internet protocol suite can be used: HTTP(S), MQTT, AMQP, CoAP, ..

Getting Started:

Instructions on setting up your project locally: **How to Run the Server**

Open `settings.py` and change the IP to the network IP of the device running the server and change the PORT number to any open port as desired such as `ip = <Your Local Network IP address>` and `port = <CHOSEN PORT NUMBER>`.

```
$ git clone https://github.com/manhhungking/Rock-paper-scissor.git
$ cd Rock-paper-scissor
$ python server.py
```

How to Run the Client

Run as many instances for the clients as you want using the terminals. At least 2 clients have to be opened and connected to start a game

```
$ python client.py
```

Demo

One client connected and ready to play, One client open

Instruction on gameplay after clicking on "Ready to play" screen

2 client screens after successfully connecting

1 client has made a move, the other doesn't

Both players have made a move and score is updated

When one player reach 3 wins first

Game is refresh

Results of the tests:

System Evaluation:

The implementation of the multiplayer Rock-Paper-Scissors game was evaluated based on several criteria to assess its performance and reliability.

Criteria:

1. **Number of Messages and Latencies:**
    - The system was tested under varying loads to measure the number of messages exchanged between client-server interactions and the latency experienced by players during gameplay.
2. **Request Processing Efficiency:**
    - Request processing times were recorded for different payloads to evaluate the efficiency of the server in handling incoming game moves from players.
3. **System Throughput:**
    - The overall throughput of the system, measured in terms of the number of games processed per unit time, was calculated to assess its scalability and performance under concurrent player interactions.

Evaluation Scenarios:

Two evaluation scenarios were designed to compare the performance of the system under different conditions:

1. **Scenario 1: Small vs. Large Number of Messages:**
    - This scenario involved simulating gameplay sessions with a small number of players (2 players) compared to sessions with a large number of players (6 players). The goal was to observe how the system scales with increasing player counts and its impact on message handling and latencies.
2. **Scenario 2: Small vs. Large Payload:**
    - In this scenario, requests with small payloads (e.g., simple move commands) were compared to requests with large payloads (e.g., complex game state updates). The aim was to analyze the server's efficiency in processing requests of varying sizes and complexities.

Test Case Data Collection:

For each evaluation scenario, the following data were collected:

- **Number of Messages Exchanged:** The total number of messages exchanged over the course of the session was printed out to terminal for analysis.
- **Latency Measurements:** Calculated by timestamping client-server interactions and computing the time difference between request submission and response receipt. The latency is around 0s to 0.003s
- **Request Processing Times:** Logged by instrumenting the server to capture the time taken to process incoming requests.
- **System Throughput:** Determined by conducting stress tests with increasing player loads and measuring the rate of game completions per unit time.

Analysis:

Upon collecting the test case data, we take a look at message volumes, latencies, request processing times, and system throughput under different test conditions. Then we can tell that the results of the evaluation are pretty good, and it provide insights into the system's scalability, reliability, and responsiveness, aiding in further optimizations and enhancements for future iterations.

We can't plot a graph because all of the data was printed out in terminal for looking purpose but not saved to an excel file.

Acknowledgments:

We would like to express our gratitude to the following individuals and organizations for their contributions to this project:

- **David Barclay**: For providing the initial implementation and inspiration for the multiplayer Rock-Paper-Scissors game. GitHub
- **Stack Overflow**: For providing a wealth of knowledge and resources that helped address technical challenges encountered during the development process.
- **Teacher and T.A Tri**: For their guidance, support throughout the duration of the project.