Distributed Online Battleships Game

Holappa, Joona
jholappa19@student.oulu.fi

Hokkinen, Ville
vhokkine19@student.oulu.fi

Suutari, Lauri
lsuutari19@student.oulu.fi

February 12, 2023

Abstract

This is the project proposal for Distributed Systems (2023) course held at University of Oulu.

1 Introduction to the Challenge

In today's evolving digital age online services are very important part of our society. These services range from entertainment to business services and they all require reliable performance and scalability to satisfy the end-users needs. These requirements are solved by developing the services based on a distributed systems methodology.

In game development distributed systems can be used to host games on different servers and to make sure that if one server fails the other servers can make sure that the game stays available. Another option is to deliver content with priority to make the user experience feel more responsive by reducing loading times by only loading the required content. Nowadays distributed systems are most famously used to solve the scalability issues when talking about online video games, one good example is the recent Apex Legends video game that managed to launch the video game and keep the experience smooth even with a 10 million player base in the first 72 hours.

This projects goal will be to create a simple online battleships game with a chat system with distributed systems methodology. The game and chat systems will be hosted on different servers to make sure that the project aligns with the distributed systems methodology.

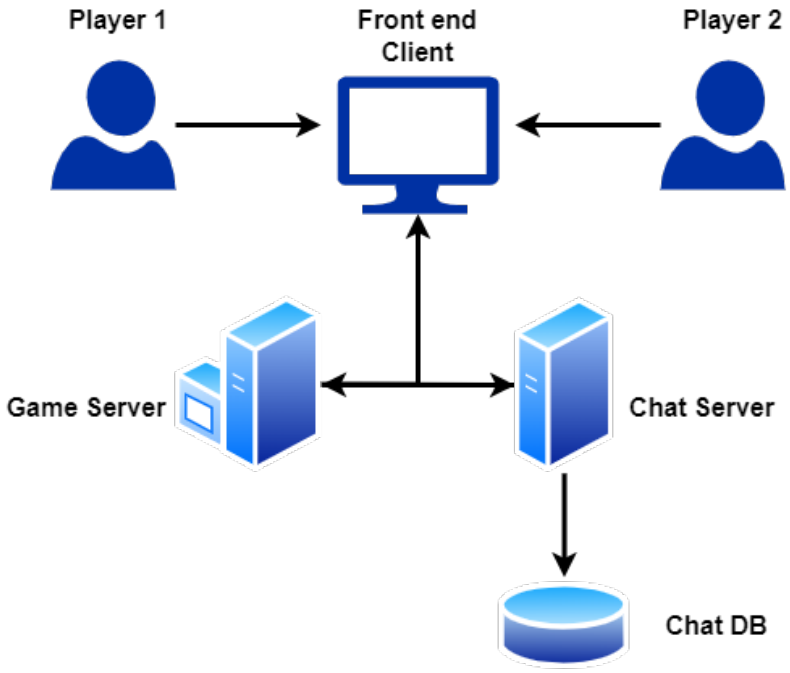

Figure 1: Initial project architecture

2 Our Solution

In this section we define the different parts of our solution to the aforementioned problem. The proposed name of this project is Pattleshibs. The end goal is to have two different servers, one for the video game and another for the chat system. This is to ensure that both servers handle different loads. The servers will process requests made by users, which can be either game related or chat related.

2.1 Battleships Video Game

The video game will be hosted on the game server which handles all the game logic and the gameplay elements of the service. It's main responsibility is to process player moves and determine if the win condition is fulfilled. It will also communicate with the chat service to make sure that both of the players can chat with each other in real time. The system will run on a client-server architecture, where the game is hosted on the server and the user requests the game state and posts moves.

2.2 Chat System

The chat system will be hosted on the chat server and it sends player's messages to the game server to show them in real time. For example, WebSocket API could be used to connect the two servers. The initial project architecture can be seen on figure 1 . The chat logs could be saved to a database such as MySQL so that they could be inspected in case of any toxicity from the players. The system will run on a client-server architecture, where the user will subscribe to the server for the chat of the current game and publish messages of their own.

3 Detected problems

3.1 Latency

Latency is an great issue in many distributed systems but our game is a turn-based game so if the latency is a couple thousand milliseconds, it is still fine. In the game and chat services the latency still acts as a problem to take into account, even though its effect might be minimal. To solve it we

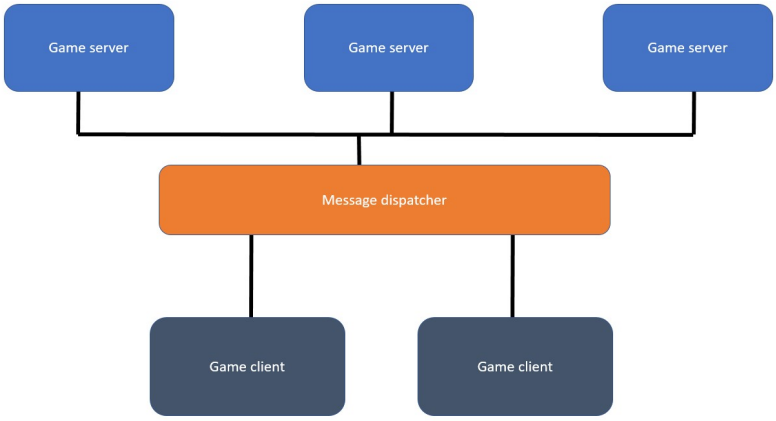

Figure 2: Game service architecture

are going to user the method of server-side-rewind, which will allow us to basically ignore round trip delay.

3.2 Scalability

With a growing number of users, serving the game and chat services can be an issue. This is inherent to online games as the number of user can reach millions. To solve this we are going to implement a system with which to do load balancing so that we can allocate the needed resources to the services flexibly, always utilizing only the required amount of resources.

3.3 Maintenance

To be able to maintain such a system we will implement a logging system for all actions and an admin user, which will be able to review these logs and make the necessary changes on the machines,

3.4 Accessibility

As a video game, the services need to be accessible with minimum downtime, so that the quality of the product stays high for the user. To achieve this, the distributed nature of the game and chat services will compensate for this and allow and hide failures and updates of servers or services from the end user by using another available server to process the request.

References