Room game

Elias Lehtinen Leevi Takala Tuula Jakobsson Peralta
Veikko Suhonen

November 2023

Contents

1 Description

Room game is an online multiplayer game that can be played in a web browser. In the game players move in rooms and interact with other players.

The web browser clients connect to the servers which form a distributed system. In the game players can move around a map. Players can see and interact with other players on the map. The map is split into rooms. Each room is handled by a different server, but the servers communicate together in such a way that players in different rooms/servers can see each other and move between rooms seamlessly. Client of player communicate mostly with the room server that player is in.

2 Implementation

Programming project contains rooms in different servers and client which shows player. Servers nodes are implemented with `Node.js` using the `Express` library

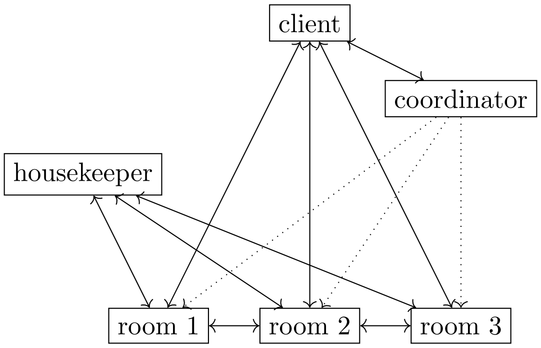

Figure 1: Nodes and their communication in the system.

and client is implemented with `React` framework. Servers communicate with each other and the clients using `socket.io`.

2.1 Nodes

The implementation has three different kinds of server nodes: housekeeper node, coordinator node and room server nodes. In addition there is a client app, which is not counted as a node. Figure 1 illustrates communications between nodes.

The coordinator node is responsible for initially authenticating clients, fetching their persistent player data and telling them which room server to connect to. After that client is connected to room where it is located.

The housekeeper node is responsible for node discovery between room servers. When a new room server is created it informs housekeeper. Housekeeper tells rooms which other rooms are their neighbours. Housekeeper creates and removes links between rooms.

The room server nodes are responsible for running the game logic loop of each room, sending and processing updates to and from clients and replicating their game state between neighbouring room servers.

2.2 Messages

Each message has a type encoded as a string and possible data as a JSON object.

2.2.1 Player join game message

Client sends message to the coordination logging in. Coordinator first creates token for player and sends it to the room server. After that coordinator returns token to the client and redirects communication to room server address.

2.2.2 Player action message

Client sends player's current button presses to the room server. Requests contain the token obtained from the coordinator for authentication.

2.2.3 Room server update message

The room servers send updates periodically. A snapshot of the local state is sent to neighbouring servers, and a combination of the local state and the snapshots of neighbours is sent to all connected clients.

2.2.4 Player change room message

Room server sends token to the room server player is moving. Room server returns the next room server address to the client.

2.2.5 Room creation

Room server informs housekeeper about itself when it starts.

2.2.6 Room information

Housekeeper broadcasts information about new room connections to room servers. The information contains addresses of room servers to let other rooms know which servers they should be connected to.

3 Prototype

The prototype will have a simple client app, a coordinator server, a housekeeper and a few (at least 3) room servers. The servers are implemented in Node.js and run in docker containers, inside separate virtual machines. Communication between servers is done using websockets. The client communicates with the coordinator with http rest and with the room servers using websockets.

The prototype does not have proper authentication, instead the clients can freely select their player identifiers.

The logs of room servers and the coordinator servers are streamed to a monitoring service such as Graylog.