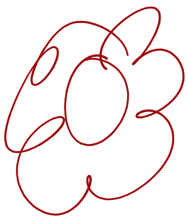

Distributed Systems – CSM13001
Course project: Design plan
Mobile curling

Sami Pentti
Benjamin Blinnikka

3 p - good idea
- missing message description, and discussion about the required features

Introduction:

The plan for the course project is to implement a modular service for a real-time mobile game, which in this case is curling. Our motivation for the project is to have a well structured real time backend service that's modules can be reused in other projects. Also we want to have a simple yet competitive and fun game for different kinds of social gatherings. Curling is a rather known and or simple consept, which makes it possible to easily get to know new people (in a pub for example) while playing the game. During the game two players throw/slide their curling disc during one's turn while the other player waits. Whichever player gets more points based on the final location of the discs wins.

Overview of the implementation:

Beside the clients the service will consist of four server nodes: a real-time server, a game state server, a lobby/matchmaking server and a authentication/authorization server. Players then communicate with one or more of these nodes to play the game.

We will use a Dart, Flutter, PostgreSQL, Docker stack and their respective libraries in the implementation of the project. Our plan is to deploy the project using Google Cloud or DigitalOcean, depending on our implementation decisions during following weeks.

Overview of the nodes:

As mentioned in the overview, the service will consist of a real-time server (RTS), a game state server and a lobby/matchmaking server, which all have their respective responsibilities.

The RTS' main responsibility will be to communicate with other nodes and deliver real-time updates to and from the clients about the state of the game. Thus the RTS will be the main node with what the client(s) communicate with. The real-time communication will most likely be implemented by using web-sockets.

The game state server keeps track of the ongoing match's relevant state information, such as scores and positions of the discs. This node also helps separate the real-time enabling logic and game logic from each other.

The lobby/matchmaking server's main responsibility will be to enable connecting clients to lobbies and game sessions accordingly. Therefore the server will create and manage game sessions and for example calculate balanced matchmaking based on players' stats from previous matches (matchmaking rating, MMR).

The authentication/authorization server's responsibility is to improve the security of the system by authenticating users and authorizing their requests.

Communication:

The overview of the service communication graph is as follows.

Real-time server:

- Communicates with the game state server.
-- Requests updates regarding to the game state (scores, calculations regarding the locations and movement of the curling discs etc.)
- Communicates with the lobby/matchmaking server.
-- To handle player matchmaking, create new game sessions and managing the game lifecycle.
- Communicates with the auth server.
-- To validate the identity of the players during a game.
- (Communicates with the client)
-- Delivers the real-time feedback of the game's state (scores, locations of the curling discs etc.)

Game state server:
- Communicates with the real-time server.
-- Receives and handles the requests sent by the real-time server. Provides the scores and calculations regarding the state of the game.
- Communicates with the lobby/matchmaking server.
-- To receive information about new players entering the game. Also communication regarding the management of the game session.

Lobby/matchmaking server:
- Communicates with the real-time server.
-- Handles requests related to the matchmaking, game session creation and management.
- Communicates with the game state server.
-- To handle information about the current game state, scores or other data.
- Communicates with the auth server.
-- To validate a player during matchmaking and lobby actions.
- (Communicates with the client)
-- To achieve creation, joining and management of new game sessions. Client sends request about joining/leaving the sessions.

Authentication/authorization server:
- Communicates with the real-time server.
-- To validate the identity of the players during a game.
- Communicates with the lobby/matchmaking server.
-- To validate player during the matchmaking process and to authorize actions related to the lobbies.
- (Communicates with the client)
-- The login process including validating the user credentials and giving an auth token for the client.

Comments:

Our goal is to implement the service as described and within the possibilities try to implement some nice but minimalistic UI for the game. Scalability could be considered within the real-time server as the number of requests may be quite heavy combined with the calculations requested and delivered.

We could improve the scalability by implementing load balancing/replication, i.e. distributing the requests among multiple instances of the server. This could also improve the fault tolerance of the service, as if one of the servers crash, another one could take over the session.

This replication process could also be considered for the game state server, as if the state server crashes, it will brick every session without a backup instance.

Simple load balance would be to make game state server to handle each game fully and then if necessary add more gaming nodes for more games.

- Message description missing the communication parties defined but the messages (meaning, content, etc.) are missing

- Also difficult to see the required features and how they are addressed in your design.

- It looks like this will be fine for the task, but partially difficult to evaluated due to missing information.

Please focus on your system's node functionalities in more details