Proposal [DRAFT]

Problem Description

Design of a synchronized, real-time chat application that can support multiple clients with on-the-fly joining/leaving capabilities. The application should be able to support a large amount of concurrent users and must be able to maintain a consistent message log in each of the respective users' chat room. Notification of any state changes (i.e. an update to the chat log, or whether a message was 'seen') should be done in real time and not require any sort of polling from the client side. These problems involve clock and data synchronization across clients.

Users can choose to either create or join an existing chat room through a dedicated server, which advertises available chat rooms and connects clients together.

Proposed Approach

We will design two CRDTs to maintain state of the chat and logging information. Further state information might be added (such as whether a message was "seen" by a user). In the context of a distributed system, we will plan to provide high reliability for these chat rooms, particularly the message logs, by replicating these logs amongst several nodes (the nodes in this case are the clients/users). Since these message logs are going to be consistently updated and read from at a very high rate, we intend to synchronize these message logs by representing these logs as a CRDT. This approach attempts to decentralize the state of the chat rooms to multiple clients while still maintaining reliability and consistency amongst each user.

Strengths

- experience integrating external libraries

Weaknesses

- limited knowledge of Go language

Opportunities

- numerous examples to study that already exist

Threats

- course load of each member