Tatu Bogdan – Distributed Programming

1. Provide an example of an alternative communication technology that could replace one of the technologies you used in the second project. Discuss the differences and the advantages or disadvantages of the respective technologies.

In our second project we made use of REST in the making of the tables and chat databases, to POST and GET the chat messages and to POST, PUT, GET and DELETE the game tables that are currently in use. We also used WebSockets for the chat for real-time message transfer between all active users and for the actual game, so users get a low-latency experience when making moves, playing, joining or leaving.

We could've used something like Firebase's real-time database for the chat, so that we wouldn't have to use both the MongoDB database and the WebSocket API to create it. One of the big advantages would be that it could seamlessly handle such case in which one communication works but the other one fails:

- Message was sent to database, but not through socket, the message would be available to new users but not the ones that were already chatting.
- Message wasn't sent to the database, but passed through the socket, the already chatting users would continue to talk normally, but any new user would be confused as to what was discussed during that time.

We could also replace the short-polling used in getting the table database's update every second with Server-Sent Events (SSE), since the server isn't updating frequently, but the information from the database should also be sent to the clients as fast as possible, so polling in longer intervals is not an option, but there's also no need for WebSockets since the communication shouldn't be full-duplex and the user is not always sending back information to the server, only when he/she joins a game. The main advantage would be the relief of strain on the network that is caused by the frequent request, even though new data is not being sent back. Even though component updates are not triggered every request because of data caching, it is still a waste of resources to send so many requests.

2. Enumerate three platforms or frameworks for developing distributed applications that are popular today.

- *Kubernetes*
  Kubernetes is Google's open-source system for automating deployment, scaling, and management of containerized applications. It was written in Go, and was originally interfaced with the Docker runtime, but have now moved on to interfacing directly with the container through Containerd and replaced Docker with CRI-O.
- *Apache Kafka*
  Another open-source platform used for distributed applications is Apache Kafka, developed by the Apache Software Foundation and written in Scala and Java. It was made to handle real-time data feeds and provides libraries for stream processing applications.
- *Dapr (Distributed Application Runtime)*
  The last distributed application platform that I will mention is Dapr, which provides runtime with APIs that simplify microservice connectivity by removing the complex challenges that come when building distributed application, such as service discovery, message broker integration, secret management etc. . The runtime system was designed to support cloud native and serverless computing.