ostracon

Team name: FunkMasterPrimeDeluxe

- Daniel Kim
- Louis Rassaby
- Scott Jacobson
- Jeremy Max Goldman

Summary

We aim to create a Python library for realtime distributed voting for use in games, distributed music composition, and crowd-sourced intelligence. We were inspired by the recent viral craze, "Twitch Plays Pokemon", where millions of users controlled (and eventually won) a complex game through voting via a text-based interface.

We want to take that idea one step further by creating a Python library that can be used for distributed voting through collecting keystrokes and sending them realtime to a server that compiles them and selects the winning keystroke from any number of keystrokes received over a given cycle of time.

The most basic use of our library would be a maze game with two or more "players" attempting to reach the end of the maze before the other players. Each "player" can be a team of between 1 and n users. The advantage of having multiple players vote is that many minds looking at the same maze may be able to solve the maze faster if they work together.

Some other applications would rely on an API. For instance, games or music composition distributed via the web might use an API to contact the host server where the Python library is running.

Another use case is social psycology studies that aim to learn about teamwork behavior between people who don't know each other across the internet.

Deliverables

We aim to break the task of realtime distributed voting into 5 categories, in order of how necessary they are to the completion of the project:

1. Python distributed voting system

1. [illegible]
2. Applications using the library via the Python interface
3. A REST API that can be used to access the voting system
4. Web applications using the API
5. A Javascript library for access that API

At a minimum, we will complete items 1 and 2.

First Steps

We will first build the Python library as discussed above. Along the way, we will investigate the merits of UDP and TCP for transmission of votes. While TCP is higher reliability, UDP is lighter weight and can therefore support more clients.

After the Python library is built, we will move into creating a simple game, such as the maze discussed above, using the library.

Forseeable Problems

Timing distributed systems is an incredibly complex problem. Delay on networks could cause votes to arrive outside of their expected time block. Depending on the application, this may or may not be an important concern.

Also, we have to decide the mechanism by which an application can collect data when our program is ready to give out that data in real time.

Overall, the timing of this concurrent system will make it a complex problem.