Project 2: WeTube

For this project, you will implement, in the Go language, a peer-to-peer system that will allow multiple users to watch YouTube videos synchronously (or close to it), using the YouTube API.

A WeTube session will start when a "director" invites WeTube peers (at known addresses). The invitations will either let the invitees (a) "viewers", who can only watch the video, (b) "editors", who can both watch and direct the video (meaning, they can pause, seek, and replay), or (c) "directors", who have all of the above privileges plus the ability to end the session. Any director can also upgrade or downgrade the status of editors and viewers (but not other directors).

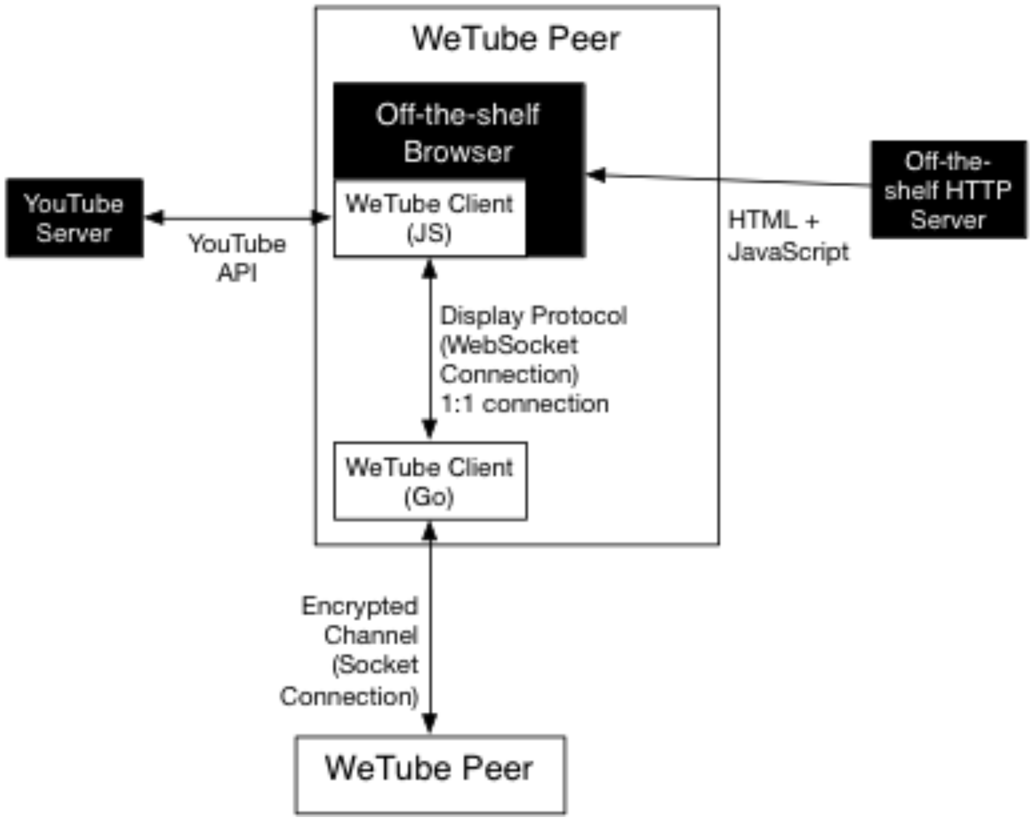
For convenience, all display and user interactions should go through a browser using the YouTube player and the YouTube API.

To prevent hijacking of control, all communication will need to be encrypted, and messages will need to be accompanied with digital signatures. (Remember that this is a *peer-to-peer system*; you can't trust all of the peers on the network!)

Your WeTube design should be fault-tolerant. If all of the directors leave the system, the remaining peers must elect a single new leader who gets to be the new director.

System Diagram

To make the above description a bit clearer, here's a simple diagram showing the various components of the system. The boxes in white are what you'll need to implement:



You will need to write the HTML and JavaScript that comprise the **WeTube Client (JS)**, whose only responsibility is to fetch and display videos, and to process user interactions (e.g. pause/play/seek/end session/send invitations). You can deliver the HTML and JavaScript to the browser using a simple HTTP server.

The **WeTube Client (JS)** will connect to a single **WeTube Client (Go)** running natively using HTML5 WebSockets. This connection does *not* need to be encrypted, as these are two trusted components running on the same machine. The **WeTube Client (Go)** only accepts a connection from a single browser client; these two components form a single **WeTube Peer**.

The **WeTube Client (Go)** will connect with other **WeTube Peers** through an encrypted socket connection, through which all of the elections / player directions / etc will occur.

Useful Links

- YouTube API
- YouTube API TypeScript Typings
- Go WebSocket API
- HTML5 WebSocket Guide
- A simple and fast HTTP server