Introduction

Our app allows people to have a shared video playlist at a party. The idea is that one computer is connected to a TV/Audio system and plays YouTube videos, while people at the party can add next videos to the queue using a web browser on their device. The main 2 aspects to know are the following:

1. The designated computer for playing videos (connected to TV) starts the webapp and opens http://localhost:8080/player
2. Partygoers that want to add a video to the queue go to http://localhost:8080/propose

Paxos is useful here because we need to resolve the order when multiple people propose video links to be added next on the queue. It is also useful in a case of a huge festival in the future where everyone would be proposing videos and the playlist would be so important that we would distribute it to multiple servers.

Instructions on running the app

Dependencies:

- We utilize the Websocket protocol to facilitate communication between the client and server. Since Go doesn't have this natively, we need to install a package.
- `go get gorilla/websocket` . Make sure the gorilla directory ends up in src next to github.com directory.

Launching the app:

1. In one terminal, go to `/cmu440-F16/paxosapp/app` and run `./start_paxos.sh`. This will start the 10 paxos nodes that serve as the replicated state of the video playlist.
2. Copy the string with comma-separated ports outputted by the script.
3. In another terminal, go to `/cmu440-F16/paxosapp/app` and run:
    a. `go build webapp.go`
    b. `./webapp -paxosports={`paste copied ports`} (e.g. ./webapp -paxosports=1337,6262)`
4. Open browser and go to http://localhost:8080/player. This page will display the current video playing as well as all the videos in the queue. If there are no videos playing, then there will be a placeholder until a video is "proposed."
5. In another tab, go to http://localhost:8080/propose
    a. Enter your search query into the second textbox
    b. Select one of the results. The video ID will appear in the proposal text box.
    c. Press submit, see results of your proposal.
    d. If your video wasn't found, you can paste its ID directly into the first textbox
6. Go back to player tab, see the video playing
7. Repeat step 5 and see the next video appear in the queue
8. Scroll to the end of the video, and the next one will play.

Fault tolerance:
To kill some nodes:

1. `lsof -i :1337 | grep LISTEN` -- on MacOS. Replace 1337 with a desired port. Choose a port that was output by the bash script.
2. `Kill 1337` . Again, replace 1337.

The player will play until the queue is empty, propose will cease to function when at least half the paxos nodes die.

Testing
To check our distributed storage for consistency, we used a test script that verifies that each paxos node contains the same data after playlist proposals have been made in the web app. The test replicates web browser actions by sending POST requests directly to the server. To run it:

1. Do steps 1-3 from previous instructions to launch the webapp.
2. go build apptest.go
3. ./apptest -paxosports={paste copied ports}