CoDrawboard - User Manual

Directory Structure

- `/backend/` : The application backend, acting as both webserver to provide RESTful API to frontend and PAXOS participant.
- `/frontend/` : Stand-by frontend that can be run independent of backend server.

How to deploy?

Frontend

To run it locally, just open `/frontend/index.html` using a modern browser (like Google Chrome, Safari, Firefox or Microsoft Edge) and you are good to go.

To publish it to the world, you can choose any method available for publishing a static website, like publishing it via gitpages, saving it to public AWS S3, or exporting the directory using nginx or Apache Tomcat.

Or, you can access the published version of frontend via http://levy.at/~draw.

Backend

Before you start, compile the backend and make sure its path exists in `PATH`

```
go install github.com/cmu440-F16/paxosapp/server
export PATH=$PATH:$GOPATH/bin
```

There is a controller script at `/backend/controller.py` that you can used to start, kill, and restart back-end servers. For example, you can use `python controller.py -s 3` to start a cluster of 3 nodes. Note down the seed server address so you can use it later in the front-end. Here are some other useful commands.

```
python controller.py -s 3       # Start 3 backend servers listening
on different ports
python controller.py -k -1      # Kill all servers
python controller.py -k 0       # Kill first server
python controller.py -r 0       # Restart the first server
python controller.py -r 0,1,2   # Restart server 0, 1, 2
```

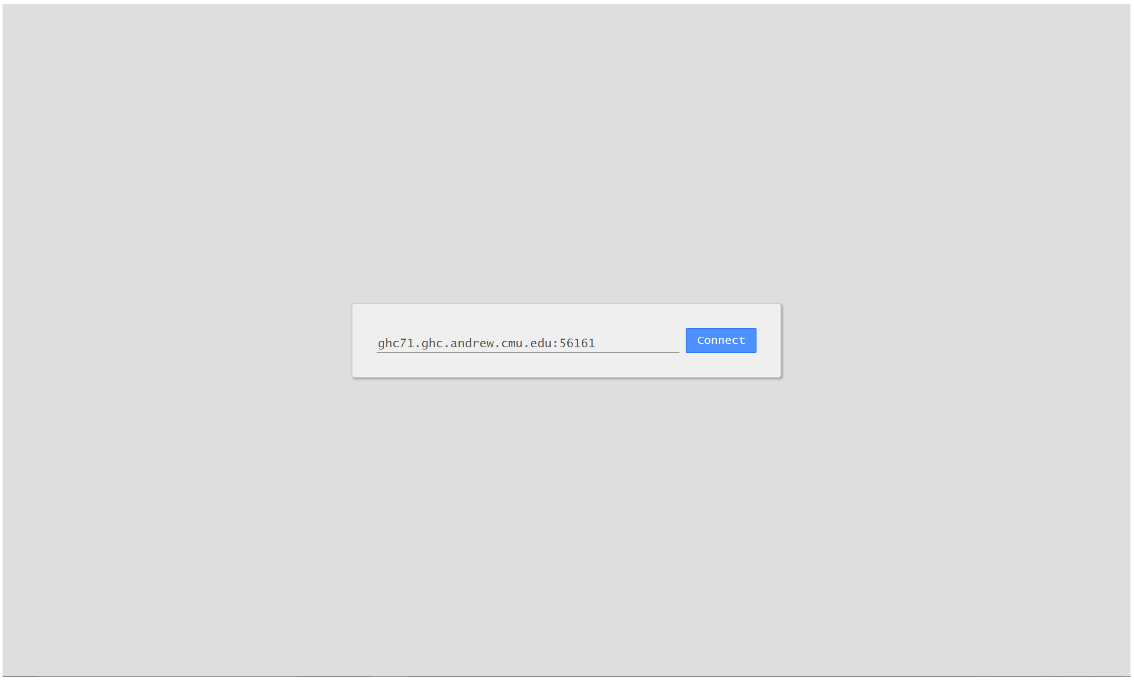
How does it works?

Each back-end server runs a HTTP server and a Paxos node at the same time. Internally, each stroke is stored as a key value pair. We use sequence number as the key to indicate which layer that stroke should be in the canvas and we use a set of points to describe the stroke and store that as the value. When server receives post request for a new stroke, it will propose that value using PAXOS for the new sequence number. Since we use PAXOS algorithm to achieve consensus, a back-end cluster with 2f + 1 nodes can tolerate f failures.

How to play with it?

Set up all the environment

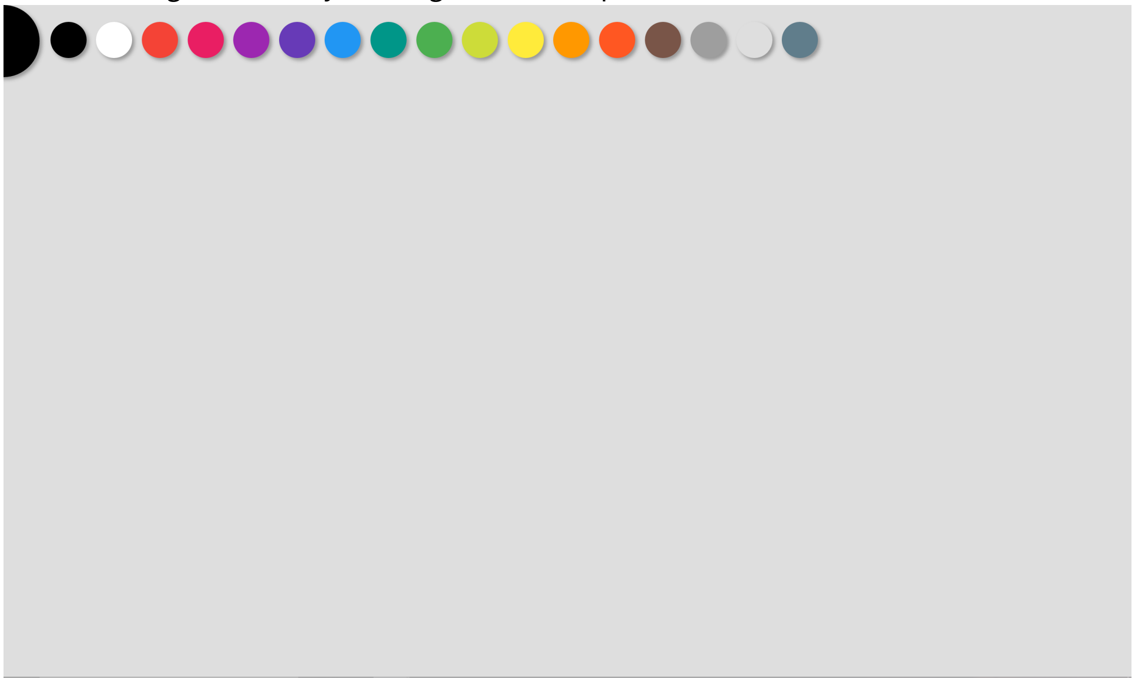
- Follow the instructions above, and setup a backend cluster. Note down the seed server address (`ip:port`), which will be used to launch the frontend.
- Access the frontend via your browser, and you'll be prompt to input a seed server.



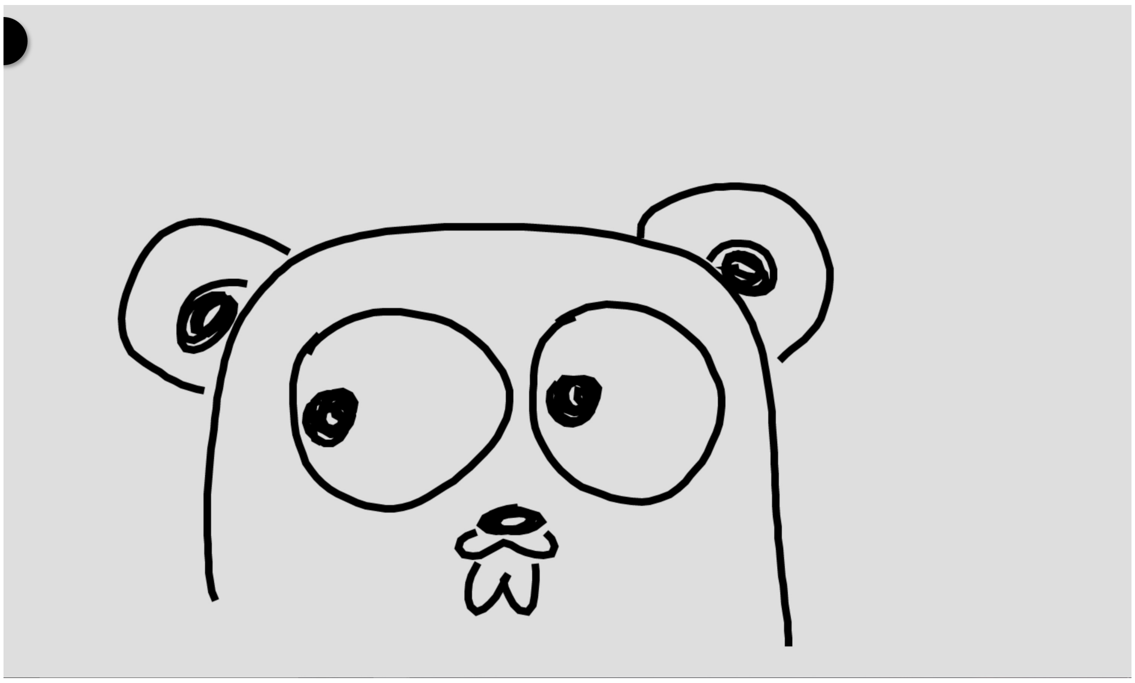
- Frontend will connect the seed server to fetch the address list of the whole cluster, and randonly pick one as connecting point for subsequent communication.

Draw on the board!

- Then you can use your mouse (or pen on touch screen) to paint on the whole canvas. You can switching the color by clicking the circle top-left.

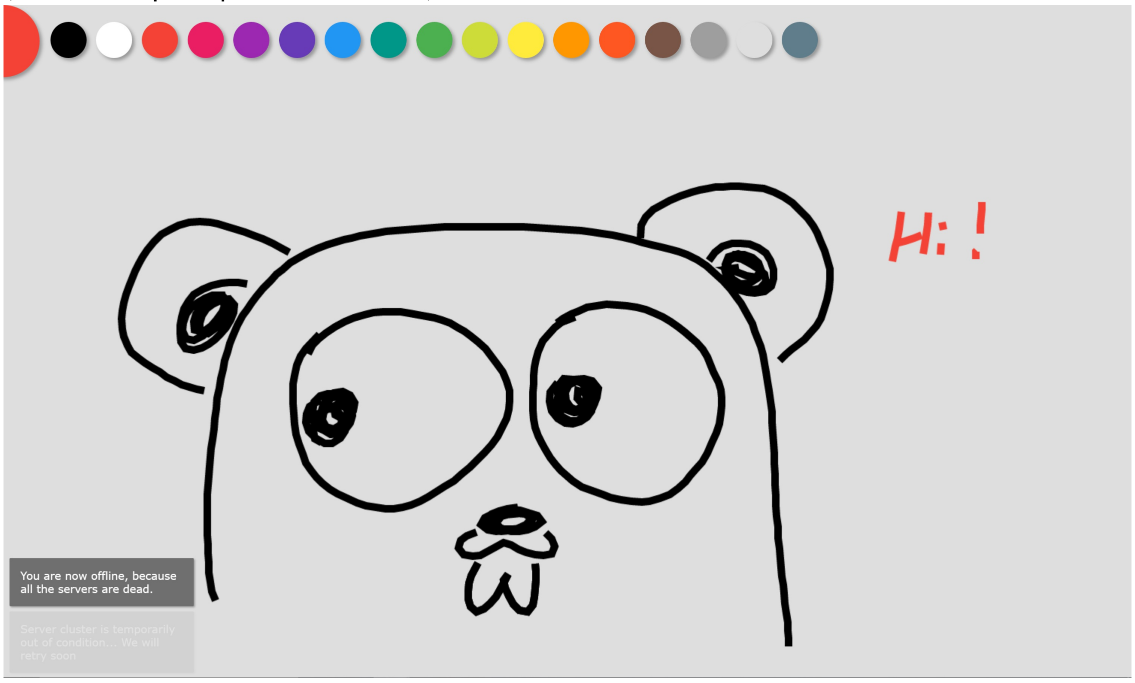


- All your strokes will be saved to server, and share with everyone connecting to the same cluster! The order of strokes are guaranteed by PAXOS.



What if server fails?

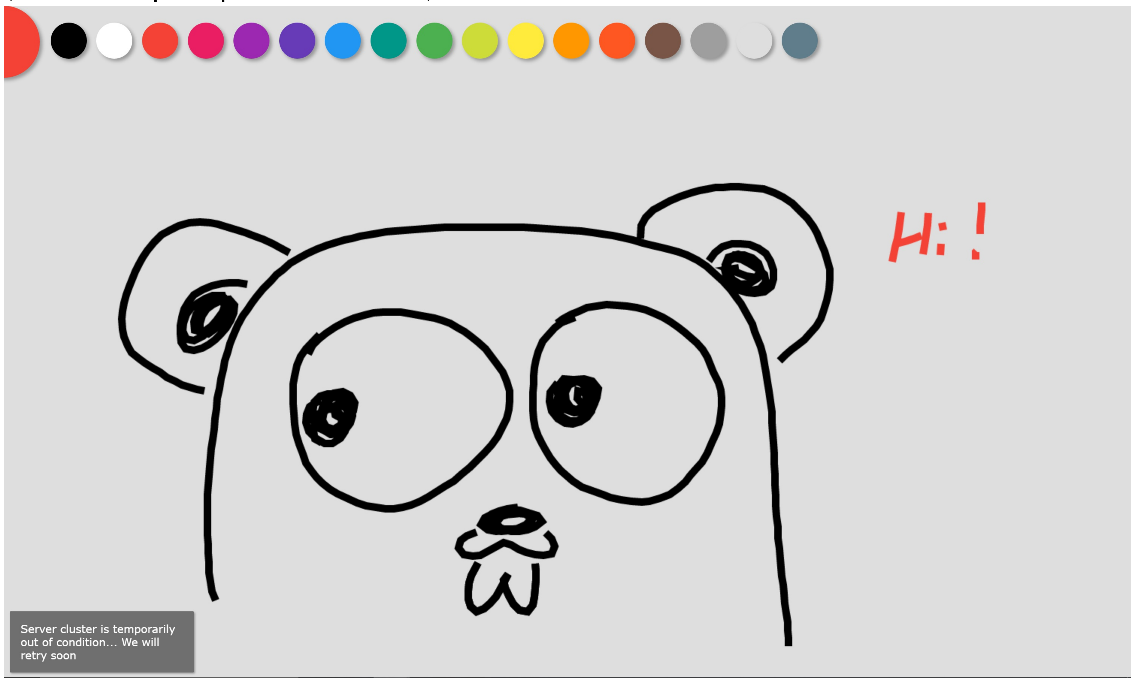
- If the connecting point fails, the frontend will mark it as dead and try to establish connection with other nodes in the address list; If success, frontend will try to update the address list in order to register the catching-up nodes. All the work, if success, will be done silently.
- If the frontend fail to connect to any server in the address list, it will go offline. From now on, any modification on the drawboard will be neither persistent nor seen by others. (Notice the prompt at bottom left)



You are now offline, because all the servers are dead.

Server cluster is temporarily out of condition... We will retry soon

- If the server is temporarily unavailable, e.g., the majority in the cluster die so that PAXOS cannot run normally, frontend will delay all the requests and retry them after 5 seconds. (Notice the prompt at bottom left)



Server cluster is temporarily out of condition... We will retry soon

Enjoy it!