GoRaft

A from-scratch implementation of the Raft Consensus Algorithm

Features

- Leader Election
- Log replication
- Persistence
- Dashboard

Deps and installation

```
# Download and install GoLang version go1.15.2 from the (official page)
[https://golang.org/doc/install]
$ go version
go version go1.15.2 linux/amd64

# Logging for shiviz visualization requires GoVector
$ go get -u github.com/DistributedClocks/GoVector

# install flask and flask_cors for dashboard
$ pip install flask flask_cors
```

About files

- raft_server.go, app.go and common.go comprise the code for a raft node
- client/client.go contains the client code.
- app/* contains the flask app for the dashboard

Starting cluster

```
$ cd go_raft #make sure the working directory is at the repo folder
# Make sure these ports are free for tcp [8080 7070 7171 8181 9090] before starting
raft
$ ./start_raft.sh
```

Running dashboard

```
# cd to the flask app folder
$ cd app

# start the flask server
$ python main.py

# go to http://localhost:5000 on your favourite browser
```

Killing a specific node

```
# make sure the port number is one of the following [8080 7070 7171 8181 9090]
$ ./kill_port.sh <enter-port-number>
```

Stopping the cluster

```
$ ./stop_raft.sh
```

For Leader Election Shiviz log

```
# run the cluster with first arg as leader. This will make the cluster die as soon as
there is a leader elected. The logs will be in the logs folder.
$ ./start_raft.sh leader

# gather the logs into shiviz compatible file by runnning the following command
$ ./gather.sh leader.log

# upload this file at https://bestchai.bitbucket.io/shiviz/ to visualize the logs
```