coe865-final

overview

Name : Damoon Azarpazhooh

Student Number : 500 664 523

---

in the following project we are using and implementation of djikstra's shortest path to calculate minimum cost of routing between to AS in an overlay network. the algorithm for cost calculation is relatively simple, we simply multiply cost by bandwidth and use that as vertex weight in djikstra's algorithm.

for inner connectivity and discovery , I decided to implement gossip protocol due to it's robustness and fault tolerance. In my implementation, gossip rumours get propagated as they are pushed to an agent.

the created binary expects a configuration file . since the given format was relatively not expressive enough, I created a subcommand called `parse-config` which searches a folder for all config files with extension of `.conf` and converts them into a `json` format the agent expects. you can test generation by running the following :

```
make config
```

for demo purposes , we start 4 agents and bind them to ports `8080`,`8081`,`8082` and `8083`. to rn demo quickly , you can just run the following in bash shell

```
make run
```

all outputs are redirected to log files , stored under `logs` directory. you can see the process of agent discovery gossip messages getting broadcasted in the logs.

gossip internals : overview

initially , when every gossip Shutdown daemon comes online, starts a listener and waits for incoming connection. it also takes in a list of initial bootstrap peers to connect to which it tries to establish direct connection to.

agent listener spawns a new goroutine per incoming connection. the newly created incoming connection is added to the list of agent's direct peers in case it has room. it replies back with a `hello message` in which it indicates whether the

connection was accepted or dropped based on the number of direct connections it was configured to accept. if the connection was dropped , it returns back with an `error` to Shutdown and Shutdown, by default ,drops connections if there is an error from swarm. in case the connection was accepted, agent replies back with addresses of peers it is directly connected to so that the other Shutdown can start infecting them. when Shutdown receive a `hello message` that shows the other Shutdown they are trying to connect to accepted the connection, they reply back with their own `hello message` which contains list of nodes they are directly connected to which the receiving Shutdown would try to establish direct connection to them.

gossip internals : rumour

agent, with the help of a scheduler component tries to spread a `rumor` at set time quantoms, in this case `3 seconds`. the `rumor` is a list of connected path from other domain controllers.

rumour's raw payload are encoded to string with base64 encoding. To create the byte slice for the said encoding, I am using JSON for the sake of simplicity. Once payload is ready , it gets signed with agent's public key and encapsulated in a message which gets encoded with Golang stdlib's `gob` which is a self identifying encdoing format. Process of decoding is the reverse of this. message signature gets validated, then `gob` decodes the message and then raw bytes of message gets decoded back from base64 and then we use json to unmarshall the message struct.

once a rumour is successfully decoded, the agent tries to add the node to it's internal graph representation of the overlay network and then it runs shortest path algorithm.