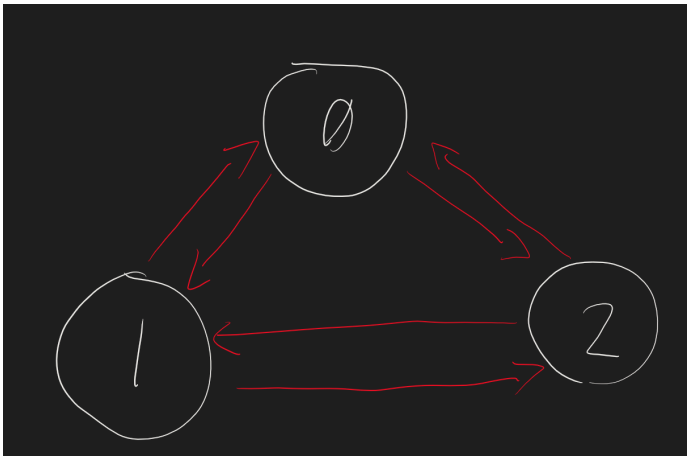
Group:
Lorenzo Pedroza
Kevin Woodman

CPE/CSC 569
Dr. Maria Pantoja
January 25, 2024

Lab 2 Assignment | Epidemic Algorithm In GoLang

@

A. Distribution of Work
    a. Lorenzo Pedroza primarily worked on the Go Routine only implementation and led the general program design such that each node runs in a concurrent go routine (run_node()) and generates periodic heartbeats, and is able to send and receiving membership tables, and is able to update the status of members in its membership tables in a responsive matter using a select statement.
    b. Kevin Woodman worked on the RPC implementation, primarily. His program utilizes 9 different processes running simultaneously and communicating by using the RPC library
B. Go Routine Only Implementation Example
    a. The following demonstrates a small-scale example scenario:
        i.
```
const (
    MAX_NODES     = 3 //8
    X_TIME        = 1
    Y_TIME        = 2
    DEAD_TIME     = 4
    CLEAN_UP_TIME = DEAD_TIME
```
        ii. Note the above times are in seconds
        iii.
```
wg.Add(1)
go run_node(&wg, memb_chans, 12, shared.Y_TIME, shared.X_TIME, 0)
wg.Add(1)
go run_node(&wg, memb_chans, 16, shared.Y_TIME, shared.X_TIME, 1)
wg.Add(1)
go run_node(&wg, memb_chans, 24, shared.Y_TIME, shared.X_TIME, 2)
wg.Wait()
```
        iv. In this scenario we will spawn 3 nodes with IDs 0, 1, and 2, and death times at 12, 16, and 24 seconds, respectively. The X_TIME is the heartbeat interval time, Y_TIME is the time interval to send the membership table to two other neighbors (which will be the prior and next ID neighbor in a circular fashion for this implementation), that is the nodes are connected as follows in this example

v. 

b. On initialization, we see all nodes and tables are initialized and tables with just themselves

c.
```
lap@LAP-SLS2:~/cpe569/CSC596_Lab2$ go run gossip_no_rpc.go
map[0:0xc00007c180 1:0xc00007c1e0 2:0xc00007c240]
Node 2 will fail after 24 seconds

Node 1 will fail after 16 seconds

Node 1 Neighbors: [2 0]

Node 1 has hb 0, time 0.0 and is Alive

Node 0 will fail after 12 seconds

Node 0 Neighbors: [1 2]

Node 0 has hb 0, time 0.0 and is Alive

Node 2 Neighbors: [0 1]

Node 2 has hb 0, time 0.0 and is Alive
```

d. Then the sharing of tables starts along with heartbeats

e.
```
Node: 1  💗 local time:= 1.000695834

Node: 1  Membership Table
Node 1 has hb 1, time 1.0 and is Alive


Node: 2  💗 local time:= 1.000990608

Node: 2  Membership Table
Node 2 has hb 1, time 1.0 and is Alive


Node: 0  💗 local time:= 1.000527853

Node: 0  Membership Table
Node 0 has hb 1, time 1.0 and is Alive

DEBUG: Node 2 : sending tables to  [0 1]
```

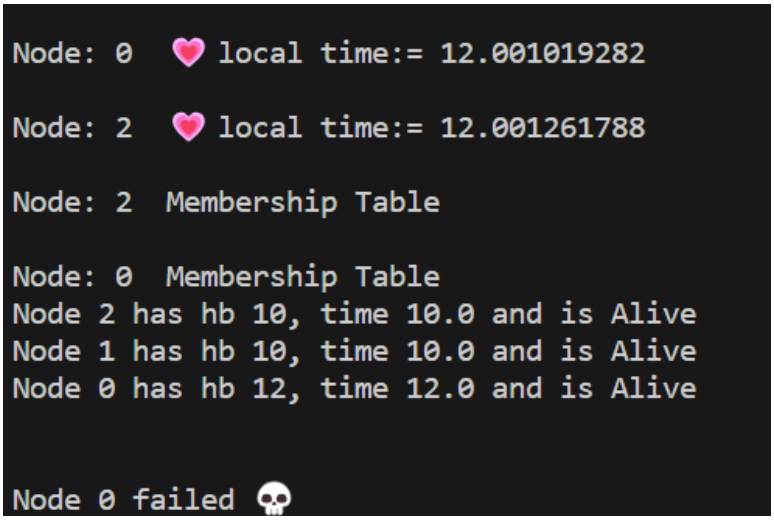
f. Eventually every node's table learns of all the other nodes after 3 seconds of the simulation.

g.
```
Node: 0  💗 local time:= 3.0008988

Node: 0  Membership Table
Node 2 has hb 2, time 2.0 and is Alive
Node 1 has hb 2, time 2.0 and is Alive
Node 0 has hb 3, time 3.0 and is Alive


Node: 1  💗 local time:= 3.001170872

Node: 1  Membership Table
Node 0 has hb 2, time 2.0 and is Alive
Node 2 has hb 2, time 2.0 and is Alive
Node 1 has hb 3, time 3.0 and is Alive


Node: 2  💗 local time:= 3.001211775

Node: 2  Membership Table
Node 1 has hb 2, time 2.0 and is Alive
Node 0 has hb 2, time 2.0 and is Alive
Node 2 has hb 3, time 3.0 and is Alive
```

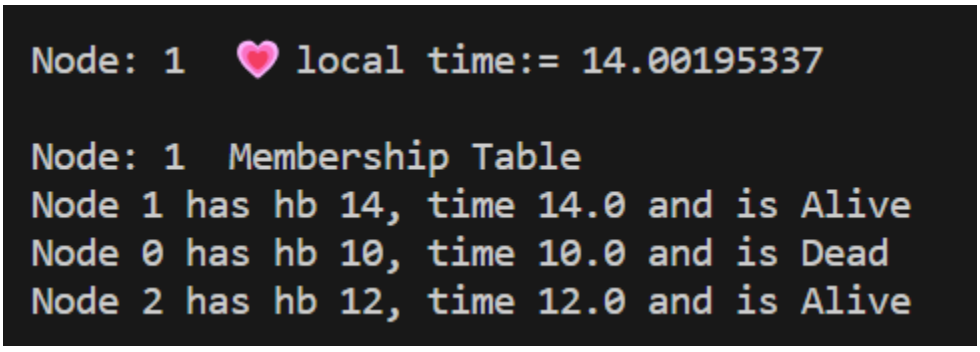
h. As the simulation progresses, the timestamps get updated and heartbeats increase

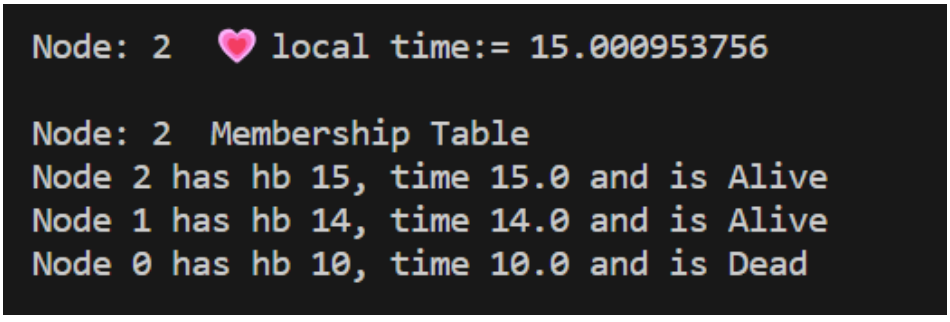
i.
```
Node: 0  💗 local time:= 9.002440935

Node: 0  Membership Table
Node 1 has hb 7, time 8.0 and is Alive
Node 0 has hb 9, time 9.0 and is Alive
Node 2 has hb 8, time 8.0 and is Alive


Node: 1  💗 local time:= 9.002687152

Node: 1  Membership Table
Node 2 has hb 8, time 8.0 and is Alive
Node 1 has hb 9, time 9.0 and is Alive
Node 0 has hb 8, time 8.0 and is Alive

Node: 2  💗 local time:= 9.002751276


Node: 2  Membership Table
Node 2 has hb 9, time 9.0 and is Alive
Node 1 has hb 7, time 8.0 and is Alive
Node 0 has hb 8, time 8.0 and is Alive
```

j. Eventually node 0 dies after 12 s

k. 

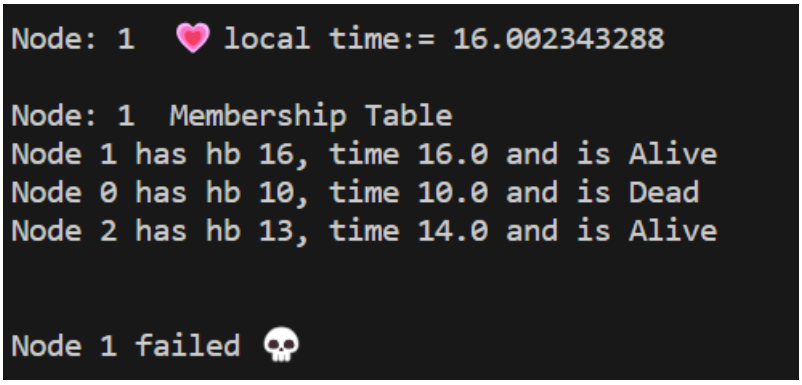
l. Thus now nodes 1 and 2 will notice node 0s death after it's heartbeat on their tables does not change for more than 4 seconds. It will mark it dead but not remvove it yet

m. 

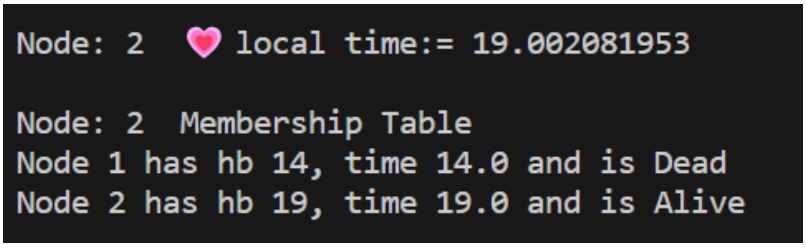
n. 

o. Node 1 fails after 16 seconds

p.

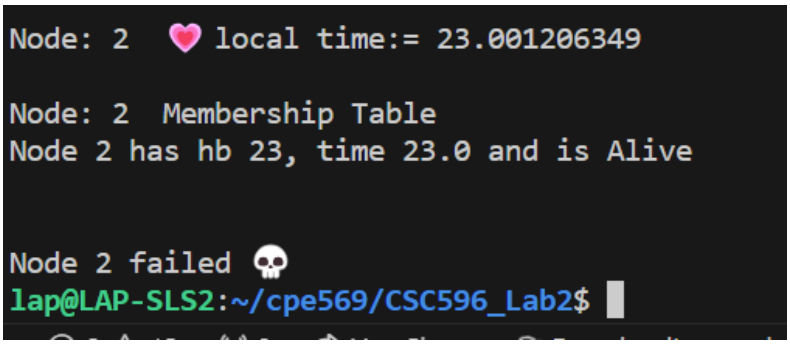


q. Node 2 finally removes Node 0 from the table another 4 seconds after determining it died. It also markes Node 1 as dead but doesn't remove it yet.



r. 23 seconds we get a heartbeat, also remove Node 1. After that, Node 2 fails, and the program finishes.

s.



C. Extra Credit: RPC Implementation

a. 8 node network example: https://drive.google.com/file/d/1ZPYF3RgbksgggAZsar-4d8ogVkanMaQe/view?usp=sharing

In this video I show an 8 node network sending heartbeat tables back and forth.

After each node is aware of all the others I kill Node 0. I then show this propagation through the system and eventually be removed from all the heartbeat tables.

After Node 0 is removed from all of the tables, I kill Node 7. This then updates to every table as well, however, killing Node 7 isolated Nodes 2, 3, and 5. This is not an error in the code but is simply a result of randomized neighbors

b. Node failing example: https://drive.google.com/file/d/1TwI5ZDXBh02v1xubBX3kxwBInkbFKeL8/view?usp=sharing

   In Example 1, the death times of the Nodes were set especially high so I could control when each node died. This example shows that Nodes will fail after a randomized period of time