LABs on GOSSIP & FAILURE DETECTION

Keep a consistent peers-list with gossip + Heartbeat-based Failure Detection

Lorenzo Ghiro

lorenzo.ghiro@unitn.it

Acknowledgment

- This lab is based on the work of **Kevin J. Qiu**

https://github.com/kevinjqiu/failure_detector

https://blog.idempotent.ca/2018/08/21/heartbeat-style-failure-detector-using-gossip

Outline / Goals

- Make practice with Gossiping and Failure Detection implementing a little (but complete!) distributed system
- Sketch Idea
  - Nodes can enter/leave the system and keep a "Membership List"
  - Nodes rely on gossiping to keep their local copy of the list eventually consistent
  - We kill some nodes and, after failure is detected, crashed nodes are eventually removed from the list

Component 1: MEMBERSHIP LIST

- In a distributed system subject to node churning (e.g., a P2P network), how does one node know who are the other active peers?
- Possible solution ⟹ each node keeps a list with the information needed to contact active peers (e.g., IP addresses)
- How to update the list when some nodes leave or crash?
  How to bootstrap new nodes?

Component 2: GOSSIP protocol

- **Gossip** information about active peers!
- eventually each node will have a consistent Membership List

Our Gossip Working Principle

- For each protocol period (--> every X seconds):
  - Each node chooses *M* neighbors and sends its own full list to them
  - Updates its own list according to messages received from neighbors

Good questions:

- How X and M influence the gossip convergence speed? (but also network load…)
- Given M, what proportionality between convergence speed & number of nodes N?

Component 3: Heartbeat FD

- Each node sends periodical messages to neighbors to make them aware that… "hey, we are still alive!"
- Somehow, it's like having neighbors taking the pulse of each others by checking that messages arrive regularly, this is why messages are called *heartbeats*

How to implement Heart-beating

1. Periodically, each node *P* increments its own *heartbeat-counter* and sends it to each neighbor *Q* via multicast (e.g. gossip)
2. If *Q* does not receive a heartbeat from *P* for a given amount of time (missed some beats!!!) then *Q* removes *P* from the membership list

- Network failures may occur, dropping some beats from *P* even if *P* is still alive ⟹ FD *false positives*
- To avoid them, if we miss a couple of heartbeats from *P* we first move *P* to "*suspected*" state and, if we still don't hear from *P* for a given time period, then we definitively remove *P* from the membership list

Before Starting Playing with Python

- Communication → HTTP
  - Why: because it's just an academic exercise and it's easier to rely on a convenient framework such as FLASK to quickly define an API for nodes. The focus is not Socket Programming or Remote Procedure Calls
- Message Format → JSON
  - Why: again, to keep the exercise simple and human-readable, even if a real-world Distributed systems would not use such a heavyweight format
- Network Environment → 127.0.0.1/8
  - Why: Easier than setting up a true cluster! Excellent prototyping/debugging environment! So we just need to assign local IPs and port numbers to different processes

Further Tools

- APS scheduler

  Advanced Python Scheduler (APScheduler) is a Python library that lets you schedule your Python code to be executed later, either just once or periodically.

  ```
  conda install apscheduler
  ```

- Pyinvoke

  Invoke is a Python (2.7 and 3.4+) task execution tool & library, essentially, a glorified Makefile.

  ```
  conda install invoke
  ```

What we do together

- Explore the provided code together
- Run a demo
- Then you can play with parameters and answer some interesting questions…

Possible research questions

- **[STRESS THE PROTOCOL]** Try to change parameters so that, without killing nodes, still not all nodes are able to keep an acceptably synchronized membership lists
  - Network-size
  - Protocol-period
  - Suspicion Threshold
  - Failure Threshold
- Have fun implementing a way to measure "generated throughput"
  - What if you change M, i.e., the number of neighbors to contact at each tick?
- ...

Questions?