Machine Programming 2 – Distributed Group Membership

Che-Kuang Chu(ckchu2), Jhih-Wei Lin(jhihwei2)

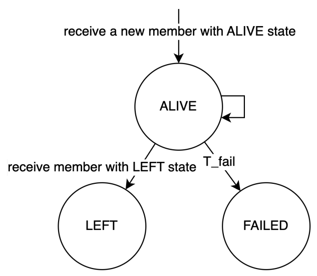
Design

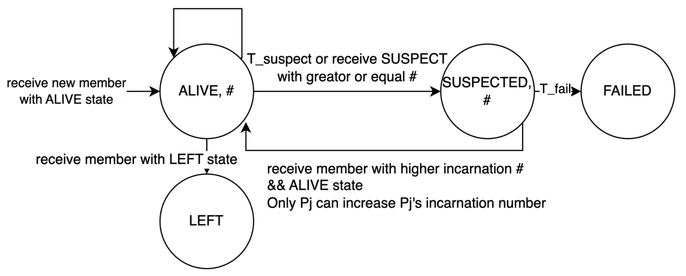
There are five main components in each program in our design.

- Heartbeat: a go-routine that sends its membership to other 4 machines every T_heartbeat with UDP.
- Receiver: a go-routine that receives heartbeat from other machines with UDP.
- Failure-Detector: a go-routine that goes through membership and detects expired machines every 1s. A machine is expired when not receiving heartbeats after T_fail.
- Cleaner: a go-routine that goes through membership and removes expired machines every 1s. A machine is removed when it's FAILED and stays in the list after T_cleanup.
- Command server: a go-routine that receives commands (join, leave, etc.) with TCP.

On receiving heartbeats, we update the members in the membership one by one according to the following state machine:

Gossip Gossip+S


receive a new member with ALIVE state
ALIVE
receive member with LEFT state
T_fail
LEFT
FAILED


receive new member with ALIVE state
ALIVE, #
T_suspect or receive SUSPECT with greator or equal #
SUSPECTED, #
T_fail
FAILED
receive member with LEFT state
LEFT
receive member with higher incarnation # && ALIVE state
Only Pj can increase Pj's incarnation number

Heartbeat target selection with Round Robbin, and Membership Permutation

We set VM1 as the introducer, and each machine will heartbeat only to the machine on joining. Then, VM1 will update its membership and mark it ALIVE. After that, it will heartbeat to machines in its membership, hence, the new machine will receive a new membership.
We applied round robin and permutation to select our heartbeat targets to make sure every machine is sent a heartbeat in 2N-1 time intervals.

Bandwidth Efficient & Message Format

To lower the bandwidth, we serialize the membership on transmitting. The serialized structure contains only essential properties and reduces the packet size to about 500B.

Why meets 5-second completeness

In our code, there is a configuration file (.msl/config.yml) to easily config T_heartbeat, T_fail, T_cleanup, and T_suspect to fit the 5-second completeness.
Also, we have a round-robbin mechanism and the messages can be sent in a fixed time-bound.
Gossip: T_heartbeat = 0.5s, T_fail = 3s, targetNum = 4, TotalNum = 10

- failure detection time: 0.5 * (10 / 4) + 3 = 4.5s <= 5s

Gossip+S: T_heartbeat = 0.5s, T_suspect = 1.5s, T_fail = 2s, targetNum = 4, TotalNum = 10

- failure detection time = 0.5 * (10 / 4) + 1.5 + 2 = 5s <= 5s

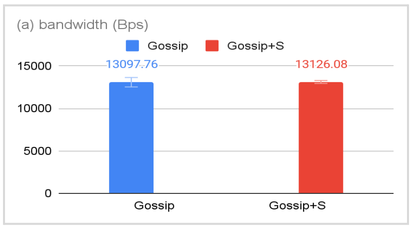
How MP1 helps MP2

In MP2, we make the logger record information both in the terminal and a log file (logs/msl.log). Hence, we can use MP1 to search through the log files and monitor the recorded state changes.

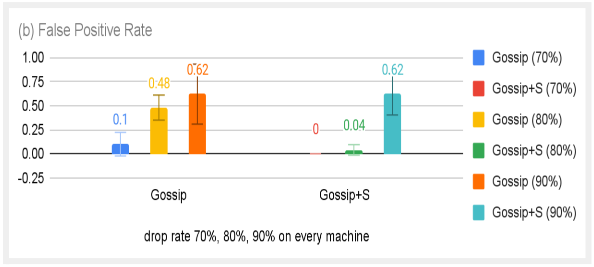
Machine Programming 2 – Distributed Group Membership

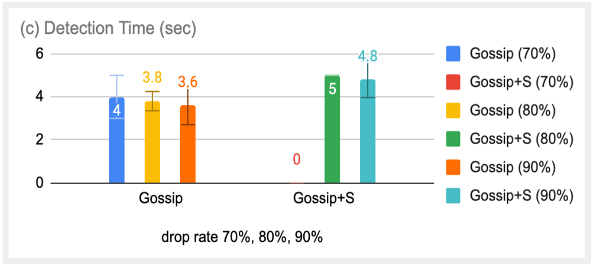
Che-Kuang Chu(ckchu2), Jhih-Wei Lin(jhihwei2)

Analysis

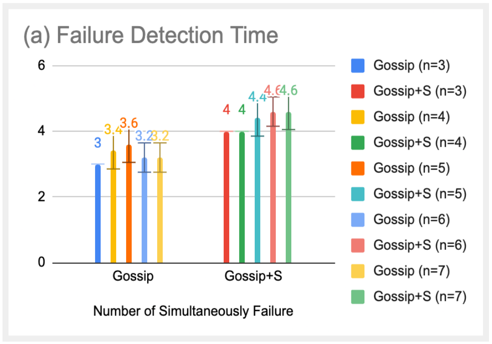


In the no-failure scenario, both Gossip and Gossip+S show similar bandwidth usage. This alignment is expected, given that no failures have been detected, and the heartbeat messages used by both Gossip and Gossip+S share a similar design in my system.

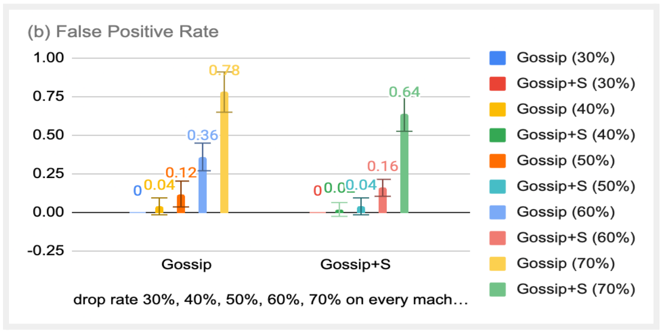


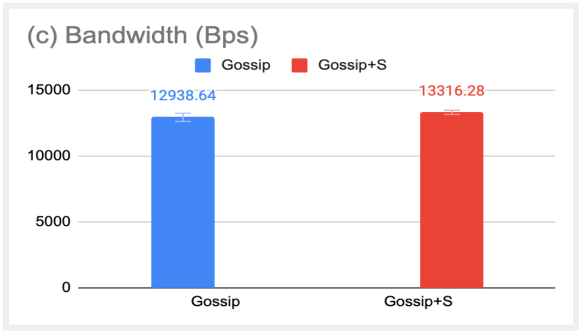


Gossip exhibits a higher False Positive Rate compared to Gossip+S. This outcome is anticipated because Gossip+S may raise suspicion about a node, which could later prove to be ALIVE, whereas Gossip promptly categorizes it as FAIL. Additionally, Gossip boasts a shorter Detection Time in contrast to Gossip+S. This observation aligns with expectations, as Gossip+S introduces an extra SUSPECT state, resulting in an extra period from SUSPECT to either ALIVE or FAIL.



In the second part, we use the 10-second time-bound. In the failure detection scenario, the bandwidth of Gossip+S and Gossip remain similar because the heartbeat rate stays the same. The number of simultaneously failed machines has little to do with the detection time, which is anticipated because all the remaining machines have the same failure timeout. The detection time of Gossip+S is higher than Gossip because of the suspicion time. The difference might arise from human error. The False Positive Rate of Gossip is higher than Gossip+S because the suspicion gives machines a chance to stay on the membership list by increasing the incarnation number. The difference is not significant if the drop rate is too large for machines to receive incarnation messages successfully.


(b) False Positive Rate
Gossip
Gossip+S
drop rate 30%, 40%, 50%, 60%, 70% on every mach…
Gossip (30%)
Gossip+S (30%)
Gossip (40%)
Gossip+S (40%)
Gossip (50%)
Gossip+S (50%)
Gossip (60%)
Gossip+S (60%)
Gossip (70%)
Gossip+S (70%)


(c) Bandwidth (Bps)
Gossip
Gossip+S
Gossip
Gossip+S