CS 425 MP2 Report

Author: Carl Zhang(hz5), Ruian Pan(ruianp2)

Algorithm design:

Our system uses gossip protocol for dissemination, and SWIM style ping-ack with suspicion mechanism for message detection.

For dissemination, we assume the node 01 as the introducer. Every node who wants to join the membership needs to send “join” to the introducer. Introducer, in turn, adds the new node to its list and send the lists back to the newly joined node and gossip a message to up to 5 randomly selected members in the list every 5ms, where each node add the new node to its list and continues to send gossip messages until all have received it. When the node wants to leave, it sends “leav”(not a typo) to one of the members, who, then will remove it from its list and gossip a message with a similar fashion as the join procedure.

For the failure detector, every node in the group periodically send “ping” to a random machine from the membership list and wait for “pingack”. In case of no ack received, we first suspect the node by sending another ping to it. If there is still no reply, then we officially mark it as failure and remove it from the list. Every node will eventually detect the error.

Since gossip has a time complexity of log(N), it scales to larger number nicely. With a large N, we only need to change the amount of target we gossip to. Failure detector will eventually find every failure, though to improve performance we may increase the number of targets to send ping to.

Message format:

Our messages are all sent in string format and the indexes and timestamps are converted to digits locally , so there will be no multiple platform data types conflict issue.

Join: “join”
Leave: “leav”
Gossip join message: “ga” + vm_index(1 digit) + timestamp(13 digits)
Gossip leav message: “gs” + vm_index(1 digit) + timestamp(13 digits)
Introducer reply to joining node: concatenation of list of vm_index(each 1 digit) + t + concatination of list of timestamp(each 13 digits) + l + timestamp of joining node(each 13 digits)
Ping: “ping”
Ack: “pingack”

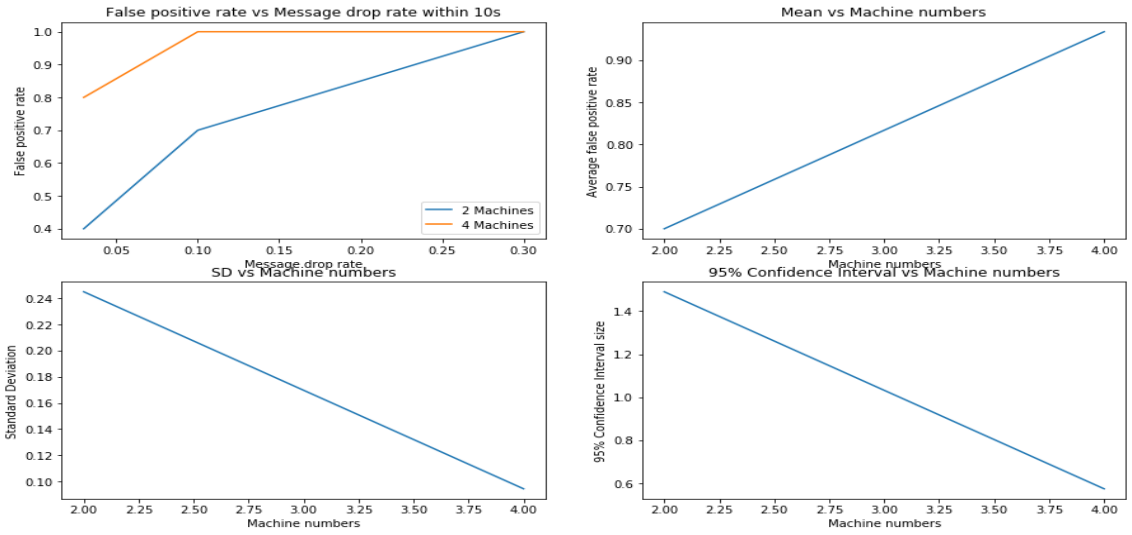
We used mp1 for debugging, by grepping the logs generated by this program. The logs would specify the changes(join, leave and fail) in the local membership list, so by comparing the logs and seeing how many grep results we have, we can easily tell whether each message is sent correctly.

Bandwidth Usage:

1. Background bandwidth usage(4 machines): 848 Bps
2. Bandwidth usage for joins(4 machines): 7074 Bps
3. Bandwidth usage for leaves(4 machines): 4794 Bps
4. Bandwidth usage for fails(4 machines): 768 Bps

Explanation: In our design, when no node joins or leaves, the only thing occupying the bandwidth is the failure detector who periodically(200ms) sends out ping message. Even in the case of failure detected, it only modifies its own list, so no difference on failure. When node joins or leaves, that's where the program begins to gossip messages to other members, and because the interval between gossip messages is 5ms, the bandwidth usage increases on the events of joins and leaves.

Plot:



Analysis:

As message drop rate increases, the false positive rate increases since more messages are dropped. And the false positive rate increases as node count increases, since for all the node, as long as one of them is considered false positive, then the entire system will off. So mean fp rate is lower for 2 machines. The standard deviation is lower for 4 machines since they tend to have high fp rates. Same reason, 95% confidence interval width is narrower for 4 machines.