CS425 MP2 Report

Fangwei Gao, Yuyao Wang
fangwei2@illinois.edu, yuyaow3@illinois.edu

Design

We use a virtual ring to locates our nodes. At every period, every node pings its three successors and monitors the acks from them.

Join and rejoin: The introducer (VM1) must join first. When other nodes request to join, they need to contact the introducer first and the introducer will update its own membership list and copy the list to the potential node then forward the "join message" to other nodes using gossip-style multicast.

Failure detection: As mentioned before, every node pings and wait for the responding acks at each period, which is about 500ms in our design. If the ack for a particular node is not received by its monitor within the first 200ms, that monitor will continue to ping that node up to 3 times with a wait interval of 100ms between each consecutive pings. If no responding acks is received for all the 4 pings, the monitor with mark that corresponding node as a failure and forward the "failure message" to other nodes using gossip-style multicast.

Voluntarily leave: The node that wants to leave will send the "leaving message" to other node using gossip-style multicast and wait for their acks. If the acks from all alive nodes in its member list are received within 500ms, it will leave. Otherwise, it will send the "leaving message" directly to the nodes that did not answer ack back and wait for their response.

Message dissemination: We choose push-style gossip multicast method discussed in lecture for the reason that we want our system to scale to large N. In all the three cases we talked before, each node will randomly pick two other nodes as its gossip targets and forward the corresponding message(leave, join or failure) to them. The node that receives the gossip message will first check whether or not it gets the same message before. If not, the node will update the member list of itself accordingly and start to gossip the same message to other nodes. If yes, it stops forwarding the gossip message to avoid an infinite loop. To make everyone "gossiped" with a large probability in the group, each node would gossip the same message 3 times according to our calculation.

In our implementation, all the messages are strings composed of 4 fields, including header, content, IP of a host, and timestamp. Different headers represent different functions (figure1). Contents vary among different types of messages. For example, the content of join and leave message are empty. But the gossip joins and leave message contains the hostname of joining/leaving node because the receiver needs to know who wants to join/leave.

Figure 1

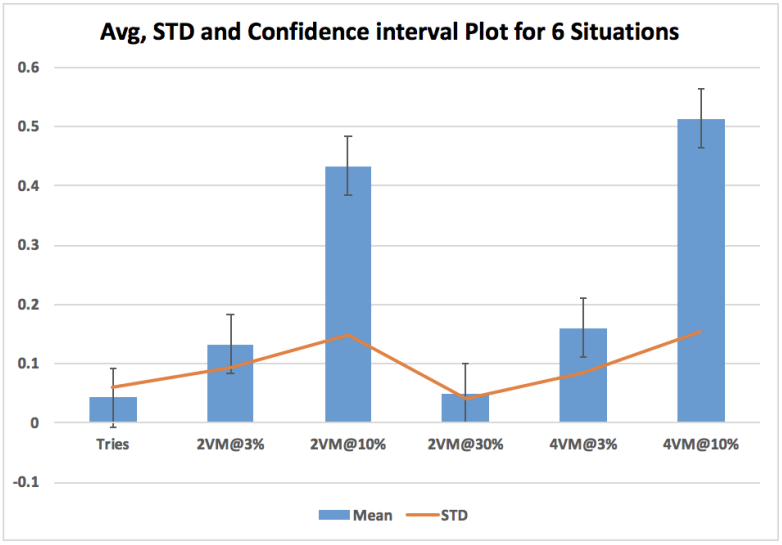
header	0	1	2	11	13	21	22
msg	join	leave	ping	gossip leave	gossip join/fail update list	ack leave	ack ping

Scalability: Our system use gossip-style message dissemination method, which is proved to need log(N) periods to guarantee most of nodes receiving the message. On the other hand, each node only pings its three successors at a constant period. Therefore, our system should have relatively good scalability for large N. Also MP1 helps us a lot in debug with logs of all vms. It grants us the ability to search for useful information among all the send/receive logs. We writes the logs into a local log file on every machine and use grep command remote to query the log information we need.

Analysis

1) ***the background bandwidth usage***
   Our background bandwidth is 2905 bit/s = 2.9kbps.
2) ***the average bandwidth usage whenever a node joins, leaves or fails (3 different numbers)***
   Node joins:4200 bit/s = 4.2kbps(three nodes existed + one join)
   Node leaves:1017 bit/s = 1kbps(four nodes existed + one leave)
   Node fails:7956 bit/s = 8kbps(four nodes + one fail)
3) ***plot the false positive rate under 6 situations***
   We take 5 readings each data point to test the to calculate the false positive rate
   (Figure 2) The vertical slim bars at the top of each data mean bar are the confidence interval for different situations.

Figure 2



From the figure above, the FP rate increases as the message loss rate increases. The VMs in the group increases, the FP rate also increases. It matches our expectation, since as the message loss rate increases, a node is more likely to be marked as failure., which leads to a higher probability of being marked as a false positive. Likewise, when the number of vms increases, the chance of being marked as false positive also increases as well.