CS425 MP2

Design

We using the basically same strategy as SWIM paper did, but with small modification for PINGREQ target chosen. In SWIM, the PINGREQ will be sent to k random server in the system, but in our design, it will be sent to the monitored peer for current node. Basically, the topology structure for our system is a ring, each node are responsible for keep track of 4 successive node in the ring structure. When a ping failed, it will send other three nodes a PINGREQ request to make sure whether the node really failed. If yes, it will add the SUSPECT information in local piggyback buffer. For dissemination part, every PING and PINGREQ will have a field for piggyback buffer, so that the change in node status can be spread over the ring backbone. After the Piggyback massage is disemminate for a pre-defined times, such infomation will be erased from the local buffer, at the moment, if there is not ALIVE massage for the SUSPECT node, it will be CONFIRM as failed. For message marshaled part, we use the Marshal and Unmarshal function provided by json library in Go. Our system is scalable because we simply maintained a ring structure, so adding a new node just make the ring bigger, the basic logic is the same that a node monitor the successive 4 nodes, the watching list will be changed when a nodes join.

If a node failed, even if three nodes failed simultaneously, there is always a monitor node watching for it since each node have 4 monitor nodes in our ring structure. So there must be at least one node notify the failure and reflected in its local membership list as SUSPECT. The dissemination period over ring in our 10nodes system is 3 PING cycle (each node will send information to the later 4 nodes), in our system the PING frequency is 0.2s, so assuming small network latencies, our system can easily achieved the 6 seconds boud for single node reflect and 20 seconds bound for system reflect.

Test

For test section, we develop a network tracker which keep track of the total Bytes sent and total Bytes received from our network layer. And to measure the below metrics, we simply use the difference in total Bytes sent/received measured by our tracker to divide the time difference and calculate the average value from records of different machines.

Background bandwidth usage for 6 machines

	Sent(Bps)	**Receive(Bps)**
Average	1888	1888

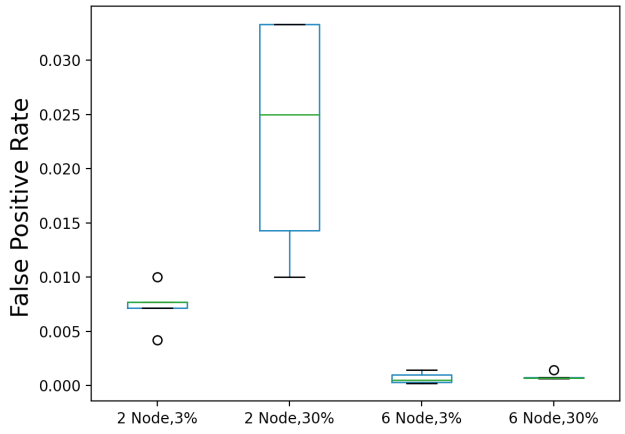
Since we do not ping everyone else, and also carry the updating piggyback massage in our PING massage, our average background bandwidth usage for Sent and Receive is 1888 Bps in a 6 machines system, which is reasonable and relatively low.

Average bandwidth usage whenever a node joins, leaves or fails for 6 machines

Action	Avg Sent(Bps)	Avg Receive(Bps)
Join	1770	3645
Leave	3630	2712.5
Failed	3630	2714

For **Join** action, the average sent Bps for machines besides introducer should be roughly the same as in original system, and from our record, it's reasonable. But for average received Bps, it should be significantly higher since every time a new node join the system, the introducer have to update its configuration to all the working node in the system. The larger average receive Bps number here verify our understanding about the working mechanism. For **Leave** and **Failed** action, we treat them as the same in our system, which is, when user is leaving voluntarily, the system just forced the process to exit, so that not other machine can estabilish communication with it, with is exactly like a Failed machine. So, the statistic for this two action is basically the same, that is larger than the stable system, since the failed means extra PINGREQ between nodes and the bigger piggyback massage as well.

False positive rate of the membership system



The false positive rate for 2 node system is significantly larger than those of 6 node system, since not other in the system can offer help using PINGREQ. Also, the group of 3% drop rate for massage have lower false positive rate then 30% group, which is reasonable as well. So the trend is, the more agent the system have, the more stable (lower false positive rate) it is, and the better internet condition is, the lower false positive rate will be.