Team05 - Yuankai (yuankai4) and Boyou (boyouh2)

MP2 Report: Distributed Group Membership Team05

Design: We use VM01 as the introducer and let all the joining nodes ask VM01 for an IP to join. Since we are using UDP, we have a client that is responsible for sending messages to the destination IP address and a server that keeps running to receive any incoming messages. Since we want to share the information between the client and the server, we use a single "Servent" class to include both parts. We schedule executors to ping regularly and monitor if it will timeout.

Considering that we are only required to handle 3 simultaneous failures, we decided to ping two successors and one predecessor under the ring topology. We set the monitoring thread to ping every 0.5 seconds and the timeout to be 3 seconds. Therefore, our system scales to large N, because each node will ping 3 other nodes for any group size.

We send all the messages as a "String" since it is required for Java UDP communication. We use special characters to separate the content. Therefore, the receiver will be able to decode the message with the knowledge of its structure. For all messages that send among the nodes, it has a command type part and a command content part. The type part is one of the types we listed and the content is the additional information we expect from that command. They are split by the character ";". For the node id, we combine the IP address and the join timestamp. The timestamp will be an empty string if the node is not in the group. It looks like "172.22.158.15@1664065315437". We use "," to separate the nodes if we are sending a list.

Our system meets the 5-second completeness because in the worst case that a failure happens immediately after a ping. The next ping will detect it. So the worst detection time will be 0.5+3=3.5 seconds. It meets completeness for up to 3 failures because in the worst case that adjacent nodes n_i, n_{i+1}, n_{i+2} on the ring fail at the same time, the failure of the middle node will still be detected by n_{i-1}. It would violate completeness with 4 failures when four consecutive nodes on the ring fail at the same time. The failure of the third node won't be detected because all the nodes that ping it to monitor its health are also failed. We will need three pings to send the information to the node that is the farthest from the current node. To let any change reflect in 6 seconds at all membership lists, the total time needed for the worst case (node failed) will be 3+0.5*3=4.5 seconds.

Useful MP1: We integrate MP1 with some small modifications to support the UDP communications. It helps us to only print important messages in the console and we can debug by searching and looking at the recorded log files.

Measurements:

(i) The background bandwidth usage. We used iftop to get the average bandwidth by checking the total bandwidth in a minute and calculating the per-second value.

	Test results (bps)	Average (bps)	Std
Bandwidth (bps)	6062, 6950, 5840, 7310, 7680	6768.4	709.68

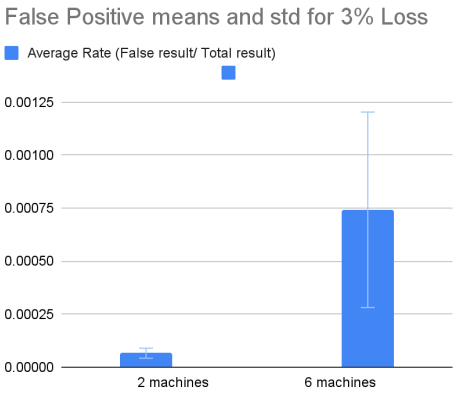
(ii) The average bandwidth usage whenever a node joins, leaves or fails. We recorded this value by the following method: 1) wait for the system to settle; 2) input

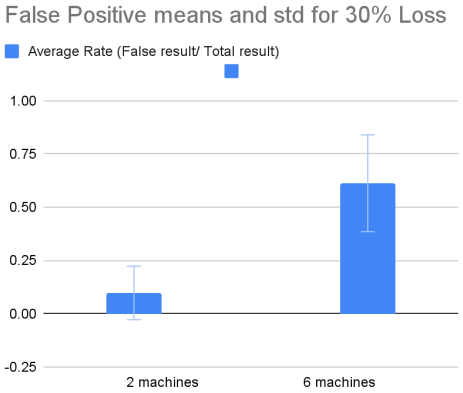
the command and wait for the system to be stable again and collect the average bandwidth usage from iftop. It is expected that we are sending more data than the stable case because we are sending other additional messages. We send the most number of messages when join, the second most when leave, and the least when fail.

	Test results (bps)	Average (bps)	Std
Bandwidth for join (bps)	46770, 45500, 43660, 46230, 38720	44176	2923.52
Bandwidth for leave (bps)	30950, 44120, 39980, 42280, 43020	40070	4757.55
Bandwidth for fail (bps)	37220, 36980, 37260, 38420, 37670	37510	506.2

(iii) Loss rate

	3% loss with 2 machines	3% loss with 6 machines	30% loss with 2 machines	30% loss with 6 machines
False Positive Rate	0.00005285, 0.00004179, 0.0001096, 0.00005610, 0.00007027	0.0005173, 0.000761, 0.0003815, 0.0004248, 0.001626	0.033333, 0.063482, 0.008734, 0.036585, 0.346154	0.283333, 0.454545, 0.859375, 0.6, 0.864865
Average	6.61E-05	0.00074212	0.0976576	0.6124236
Standard Deviation	2.36E-05	0.0004610603232	0.1254538606	0.2271969523





Since the error rate for 3% loss is too small, we draw it on two graphs. In general, the drop of ping messages has a smaller effect on the false positive because we will have multiple tries before we consider the node as timeout. It is expected that for cases with 3% loss, we have a low rate of false positives, and for 30% loss, the rate is higher. Also it is expected that it is more likely to get errors when we have more machines because the number of communication is higher. It is expected that the standard deviation is larger on 6 machines because running on more machines brings more uncertainty.