CS425 / ECE 428: Distributed System, Fall 2022
MP 2 Distributed Group Membership

Name: Hang Yu (hangy6), Tian Luan (tian23)

Design Outline

1. **Failor-Detector:** We utilized Golang to implement the whole system. In our design, every node has an entire list of memberships, and we organize all nodes on a circle topology. Given that we won't have more than three crashes simultaneously, each node only needs to monitor the following four members (if the number of other members is bigger than four) to detect failures. This is because every node needs to be monitored by at least three nodes, or when three consecutive nodes fail, one of the failures will not be detected, which destroys completeness. In addition, we also need one other node to forward the fail message. If a member wants to leave voluntarily, it will send a leave message to the nodes it is monitoring, and these nodes will forward this message; this process is very similar to the fail message. If a member wants to join the member list, it needs to let the introducer, appointed by hard code, introduce it to all members in the current member list. Every member will have a timestamp so that the ID will be unique, ensuring that the newly added node won't be accidentally deleted. This design could be scaled to large N since every node only needs to monitor at most four node, by contrast if every node needs to link to each other or monitor each other, the network will have serious burden. We marshaled the message in Json format, every message contains source id, source ip, message type and payload, where payload contains the failure/leave node's information. Besides, we output the error log to file and use the distributed log queries in MP1 to debug errors.

2. **Introducer:** The introducer is a goroutine for a TCP server that keep listening on port 9981. The introducer can be any node in general. However, for demonstration purpose, we fix the introducer to be a goroutine for the daemon process on our VM10. Each daemon process on different VMs will also listen on their 9981 port for TCP connection. If a new node is joining, the node will call the introducer with the specific functions so the introducer will first update the membership list and the monitor list and then update the most recent membership list to the new node together with new node's monitor list. On receiving that, the new node again ask the introducer to let all the other members know its existence and then the introducer will call every other member and update the membership list and monitor list to all those members. And this is how one new node join the ring through introducer.

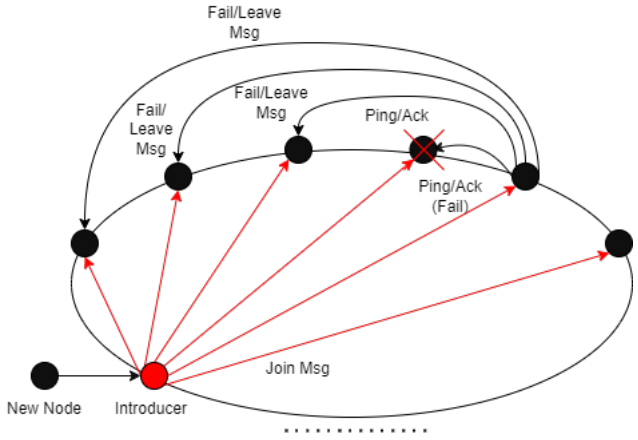

Figure 1: Design of Distributed Group Membership

Experiments

1. **False Positive Rate:** We tested when there are 30% packet loss and 3% packet loss within 2-node ring and 6-node ring. For each combination of the experiment, we ran for five times and the statistic results are shown in Tab 1. And the plot for cases when N = 2 and N = 6 are plotted in Fig 2

	R1	R2	R3	R4	R5	Mean	Std	90% Confidence Interval
30% loss, N=2	10	5	4	5	6	6.0	2.1	[3.76, 8.24]
30% loss, N=6	9	10	11	9	9	9.6	0.8	[8.75, 10.45]
3% loss, N=2	29	65	11	6	10	24.2	21.9	[0.87, 47.53]
3% loss, N=6	13	14	10	11	10	11.6	1.6	[9.87, 13.33]

Table 1: Experiment results for the false positive debut time. Metric in table (s)

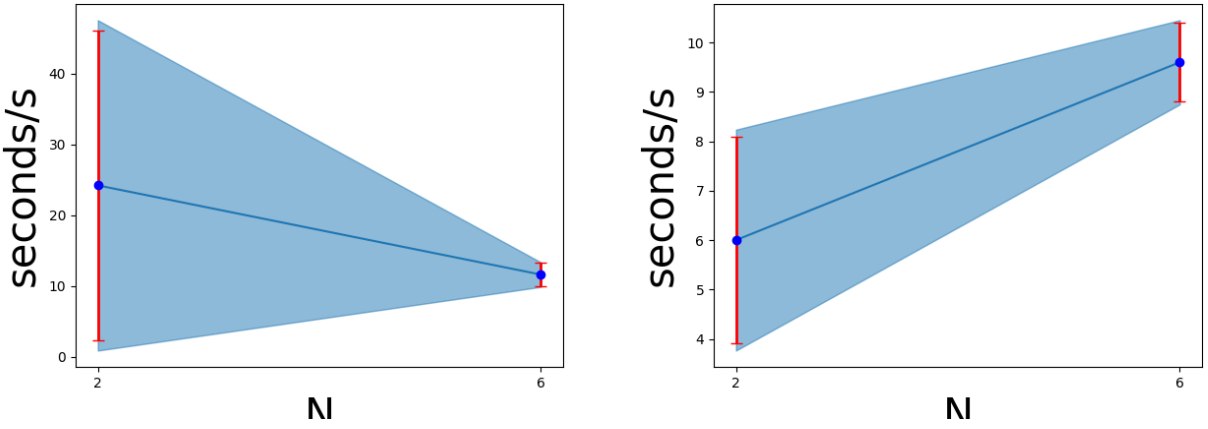

Figure 2: False positive rate experiments. Left figure shows the mean, std, and 0.9 confidence interval(red vertical line) when packet loss is 3%, and the right figure has packet loss 30%

2. **Average Bandwidth:** For the background bandwidth usage, we join the nodes to form a 6-node ring and wait 20s for it to converge and record the background bandwidth for 3 times. For join and leave, we wait the 5/6 node ring to converge and then perform join/leave and check the system average bandwidth, we repeat this procedure for 5 times and calculate the measured average bandwidth along with the standard deviation.

   - **background:** 829 ± 235 Bps
   - **join:** 656 $\pm$128 Bps
   - **leave:** 404.25 ± 78.71 Bps
   - **fail:** 369.17 $\pm$67.80 Bps

   In our experiment with the distributed group membership, our measured average bandwidth for node leave and node fail is similar, which is expected since leave partial borrows fail's code which would use fail message about the monitor list. The node join operation takes higher bandwidth compared to leave and fail since apart from the common UDP communication, this also involves that the new node asking the introducer for joining the ring system, and also the introducer introduce the new node to all other members in the ring. The background bandwidth might not be accurate since there are also other unknown communications on going in the monitor which we are unable to recognize. However, all the average bandwidth are higher than the theoretical value which should be the normal case since there might be some interfere by unrecognized network transmission.

Conclusion

In our false positive test results, combining Tab 1 and Fig 2 we can see that the larger N we have, the later that the false positive would first appear (debut). In addition, the larger N we have, the smaller variance of the time that false positive debuts would be. Also, for a fixed N, we can see that higher packet loss rate significantly shrink the false positive debut time.

In summary, the more processes a distributed system have, the smaller variance the system would have; also the less packet loss the processes have, the smaller false positive rate the distributed system would have, i.e. both increasing N and reduce packet loss improve the realiability of the distributed system.