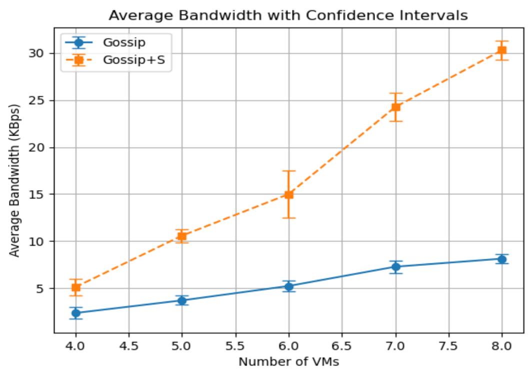
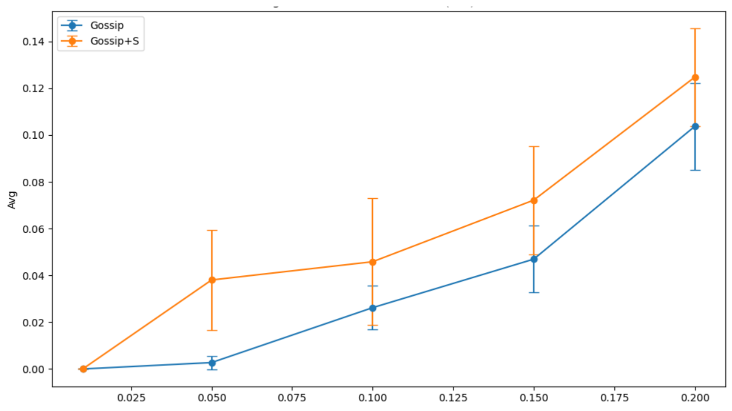
MP-2 Report by Colin Zhou (colinz2) and Keyang Xuan (keyangx3)

Design & Implementation:

In this distributed group membership service, all VMs form a ring topology, communicating with its previous 2 and next 2 machines. For the gossip and gossip + suspicious mechanism, our group decided to have one file called server.py which runs on each VM. The server file has send and receive functions to send and receive packets using the UDP protocol from other machines. We decided to use a dictionary as a membership list, with an incarnation number field for suspicion. Our group also used a time table, which keeps track of the cleanup time, and the time ranges to suspicion or failure. Based on the timetable entries, then the membership list will be modified accordingly.



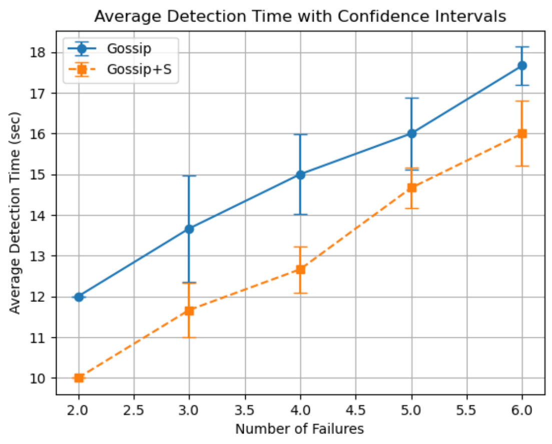


Experiments:

1 (a) . (Top left image) We set a fixed detection time bound (5 seconds) that applies to both Gossip and Gossip + S and calculate system bandwidth vs # of joined VMs. For the bandwidth, we grep the log file generated during gossiping and then use os.path.getsize() / time bound to calculate. When more VM's join, the gap between bandwidth of Gossip and the one of Gossip + S becomes bigger.

(b). (Top right image) We set various drop rates (1%, 5%, 10%, 15%, 20%) and test the false positive rate on systems with 7VMs and the figure below shows our results. Based on the trend, we notice that Gossip + S is more prone to make False Positive than Gossip.

(c). (Below image) We compare the detection time with various number of failures between both protocols. If the number of failures increases, the Gossip+S detection time is much cheaper than Gossip.

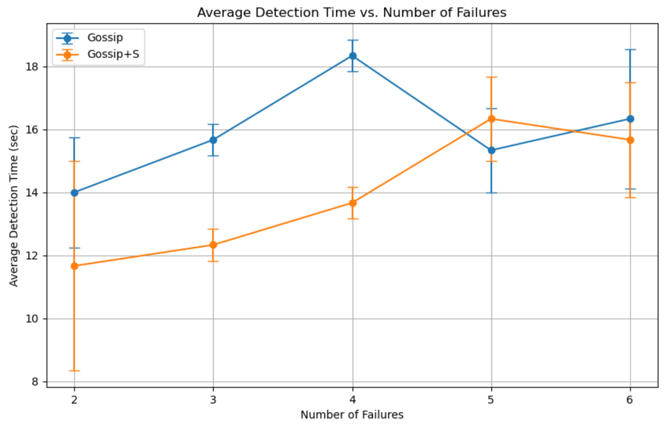


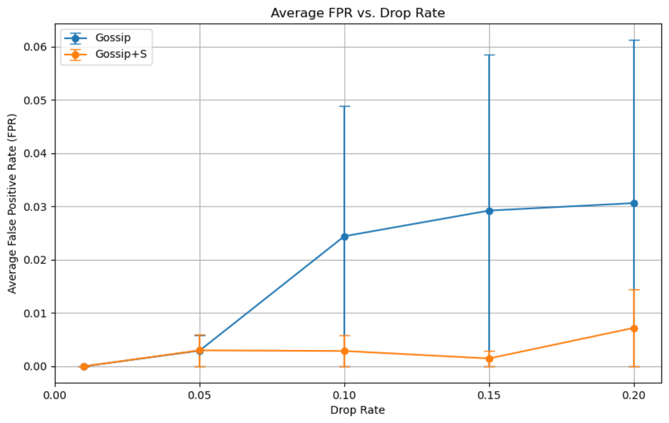
2. Based on our design, since each VM has four neighbors, it has to send pings to the other three machines and also receive pings from them. Based on that our ping format is around 400 bytes, the total bandwidth usage for a single machine is in a unit time is 8*400 + 8*400 = 6400 bytes.

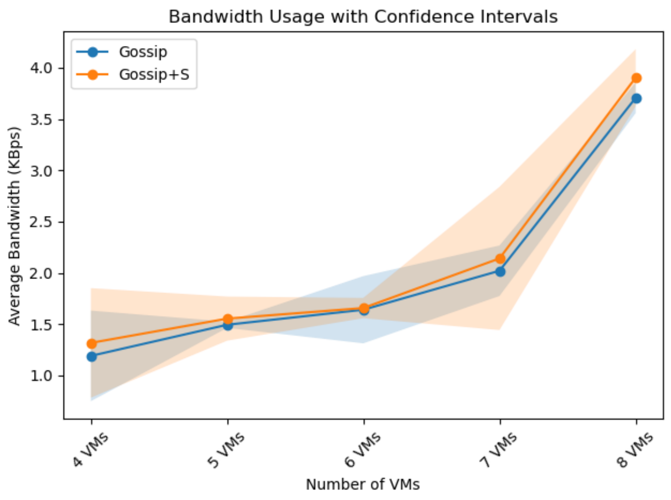
(a). Bottom left figure shows failure detection times vs simultaneous failures. No matter the protocol, the worst case is the moderate number of failures. It's possible that more failures will open up bandwidth while less number of failures is easier to be detected.

(b). The bottom right figure shows the false positive rate result under no failure case. To compare the false positives rate, we used 7VMs and calculated the PF under various drop rates. We notice that when drop rate increases, the FP rate on Gossip increases fiercely while the trend of FP rate for Gossip+S is relatively stable. It might be the case that fixed bandwidth leads to lower the level of overhead on Gossip+S, making it can better filter the false suspicions.

(c). The last image in the report below shows the bandwidth usage for both protocols. Based on the trend, we can notice that with setting the cap on average base background bandwidth usage, it controls the message size in Gossip + S, meaning that both algorithms have approximately identical average base bandwidth usage.


Average Detection Time vs. Number of Failures
Gossip
Gossip+S
Average Detection Time (sec)
Number of Failures


Average FPR vs. Drop Rate
Gossip
Gossip+S
Average False Positive Rate (FPR)
Drop Rate


Bandwidth Usage with Confidence Intervals
Gossip
Gossip+S
Average Bandwidth (KBps)
Number of VMs
4 VMs
5 VMs
6 VMs
7 VMs
8 VMs