MP2 Team63 Vincent Chen(vfchen2), Jen-Chieh Yang (jy75)

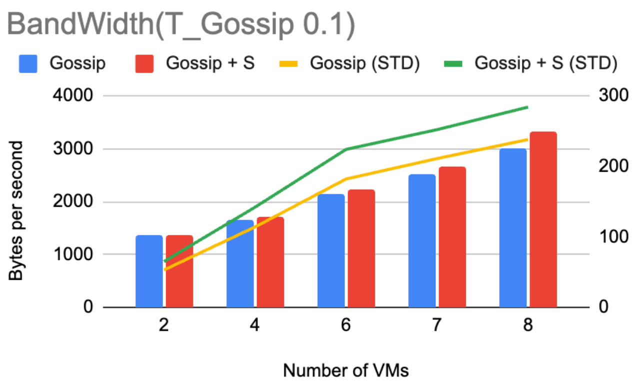
Design:

In MP2, our design comprises two components: the introducer and the server. The introducer is responsible for constantly listening and updating itself based on information received from servers. It then transmits this information to newly joined servers. Our server operates as a multi-threaded program, with one thread dedicated to actively receiving and updating based on received membership data from other servers. Another thread is focused on sending information and updating the membership list at regular intervals. We've implemented a suspicious mode using a global variable, enabling different handling based on the state within the membership list. If a user initiates a leave action, we terminate all threads related to gossiping and retain only the main thread, which remains active, waiting for a trigger to initiate another join (rejoin) process. In which, we use MP1 to grep the logs on each server to check specific states on each command. It is useful to see all logs on each server more conveniently.

1. 5 Second Time Bound

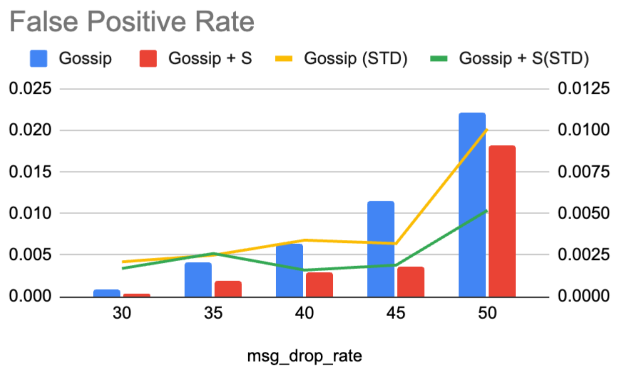
a. Bandwidth

We measure the bandwidth by adding the total send bytes and total receive bytes by each machine. As more machines are added, the membership list will be larger. Hence, It is reasonable that for more machines, the bandwidth grows. Since our design is quite robust and gossiping frequently, the Gossip + S variant doesn't differ much in bandwidth.



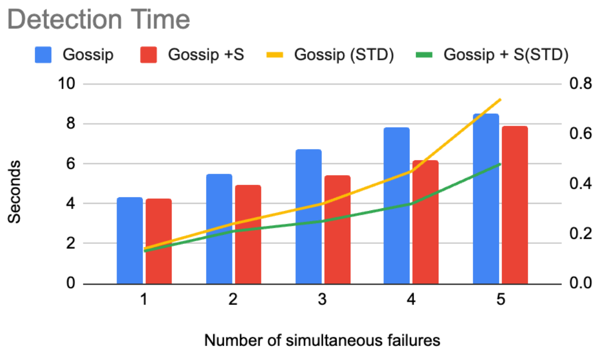
b. False Positive Rate

We measure the false positive rate by the number of times failed divided by the number of times receiving membership lists from other machines. Our starting message drop rate is 30% since it will take more than 20 minutes for one false positive to appear. The false positive rate grows as the message drop rate increases as expected. It is also proven that our suspicion mechanism works as it can maintain lower false positives than the original version.



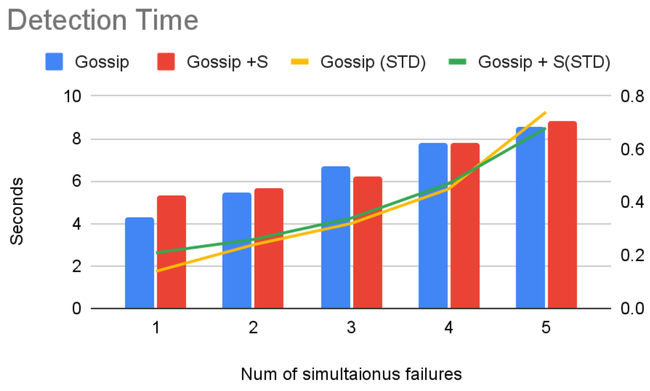
c. Detection Time

We measure different numbers of failures at the same time with respect to the seconds to detect all the failures. For normal mode, our T_fail is 4 seconds. For suspicion mode, T_sus is 2 seconds and T_fail is 2 seconds. Hence, we would need at least 4 seconds to detect failure. Our T_gossip is 0.1 while the probability of sending to 1 of the 10 machines is 0.1 in each gossip. Hence, bounded by 5 seconds. For more machines to be detected, we would need a longer time as there might be latency for printing the failure logs or the human latency on closing the VMs. But generally, the suspicion mode will not affect detection time as expected.



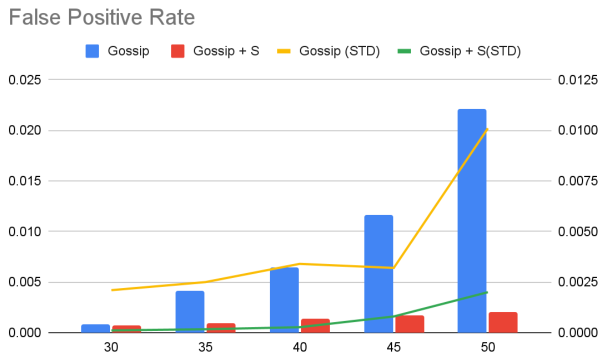
2. Same Bandwidth within 5% difference
   a. Detection Time

To allow Gossip and Gossip + S to have similar bandwidth, T_sus is modified to be longer (3 secs) in order to allow less suspicions to be arisen. With less suspicion pings, the bandwidth could decrease. However, the detection time for Gossip + S will need at least 5 seconds. The trend would be similar to the detection time in the first question.



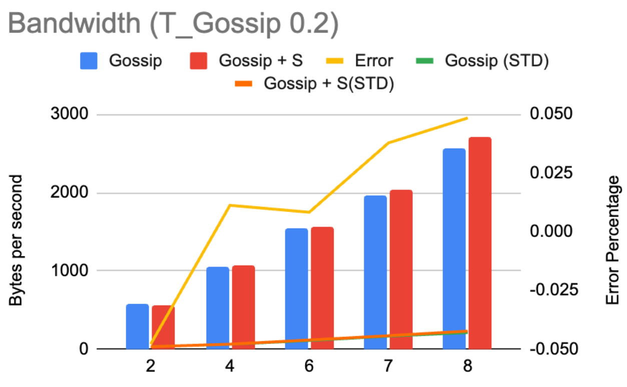
   b. False Positive Rate

Since T_sus is now longer, and allows a longer time to detect failures, it is now more robust to false positives. Hence, the false positive rate drops largely compared to the original gossip + S version. As message drop rates increase, the standard deviation will also increase as we expect.



   c. Bandwidth

For bandwidth, our T_gossip is different compared to the first plot as T_Gossip is modified to 0.2 . Hence, the times we gossip is two times more, resulting in 2 times bandwidth. And after our adjustment with T_sus, now the bandwidth between two modes is less than 5 percent as indicated.


Bandwidth (T_Gossip 0.2)
Gossip
Gossip + S
Error
Gossip (STD)
Gossip + S(STD)
Bytes per second
Error Percentage