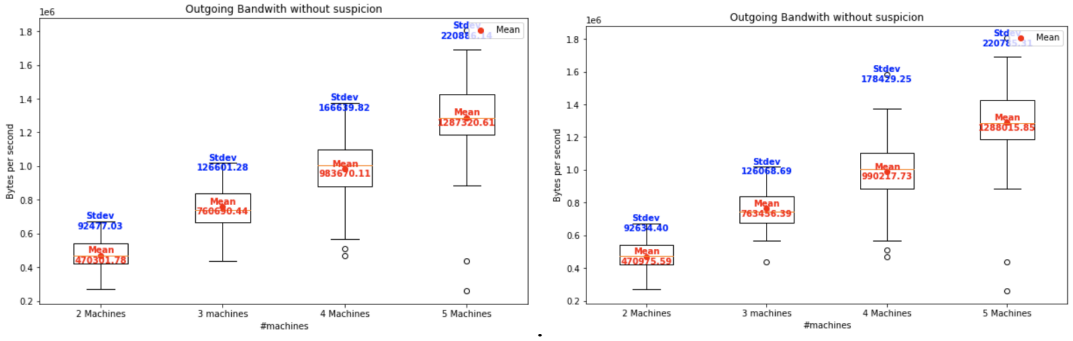
Georges Durand and Akshay Ghosh CS 425 MP 2 Report

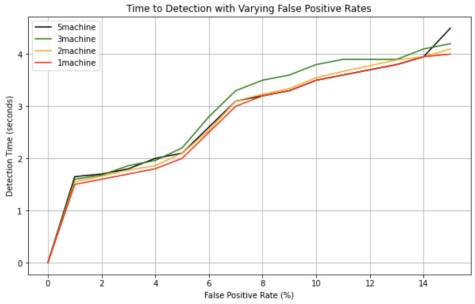
Design:

We mostly followed what the lecture discussed. To keep it brief, for Gossip we are gossipping the membership list and updating it using heartbeats (we use $T_{cleanup}$ and T_{fail} to delete stale entries). For suspicion, we also gossip the suspicion data, and once a node is suspected to have failed it will attempt to send a reincarnation message and join the cluster with a new ID.

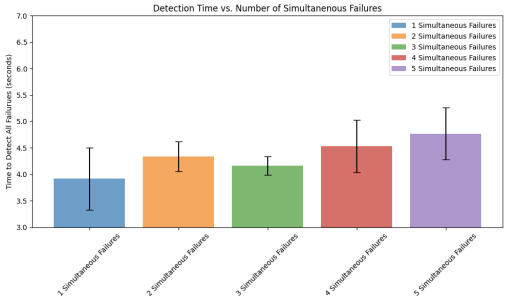
1.

a. 

The above graphs measure the outgoing bandwidth of gossip rounds with and without suspicion. We had 4 trials, each representing a cluster of 2, 3, 4, and 5 machines, where one of them is the introducer. Of these trials, we collected 20 gossip periods spanning 40 seconds (2 seconds per round). What is easily noticeable is that bandwidth is only slightly higher in gossip with suspicion. That is because our algorithm sends suspicion along with the gossip, only adding a small number of bytes to the sent json file.

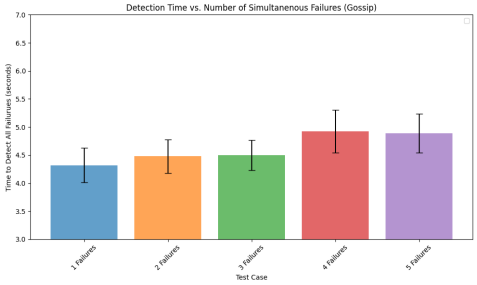


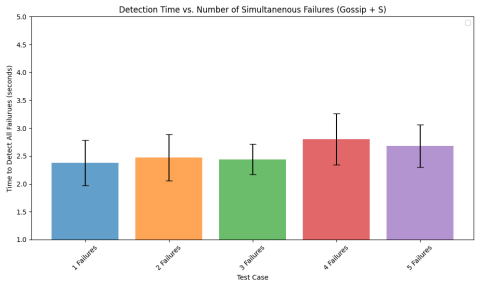
b. Looking at the chart, we can see that as we introduce more message drops, the detection time increases.



c. On average, the detection time remains about the same as failures with Gossip + S. In Gossip, failures aren't detected so to speak, they are simply immediately removed from the system. This happens at about the same time for all nodes (in our testing about 4 seconds).

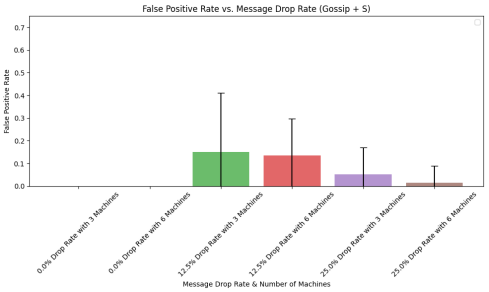
2. For each of the following data points, 5 samples were collected and their means and standard deviations were measured.

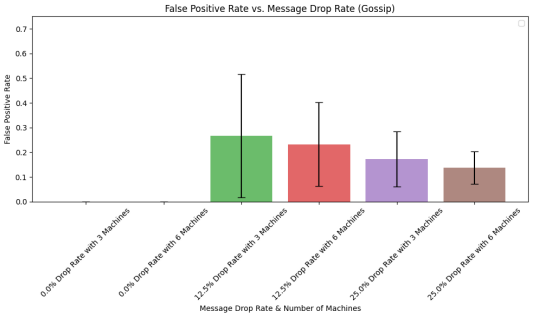




a. It is easy to see that both approaches seem to scale well with the number of simultaneous failures. Gossip + S is approximately two seconds faster than regular gossip at detecting failures, which is to be expected since failures are noticed after T_{fail} seconds (in our case 2 seconds) instead of $T_{fail} + T_{cleanup}$ seconds (4 seconds) as in regular gossip.

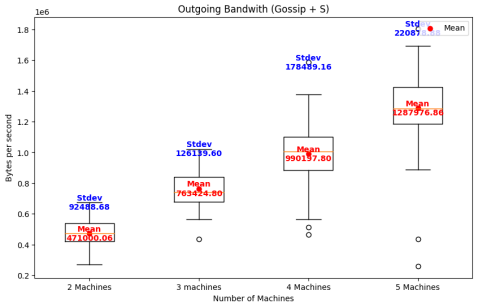
b.

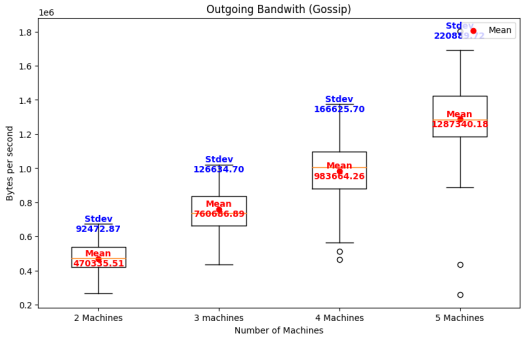




Results are as expected, we expect to see a 0% false positive rate when no messages are dropped (i.e. all failures are real). As the drop rate increases, the rate of a false positive will naturally increase. As the numbers in the cluster increase, there is a greater chance that at least one machine will receive a heartbeat from the faulty machine, which will disseminate throughout the cluster that the faulty machine is indeed alive.
We also expect Gossip + S to have a lower false positive rate since the suspected nodes have a chance to broadcast that they are indeed alive.

c.





As expected, the bandwidth between the two is almost identical (on average within the 5% bound), for our approaches, we send roughly the same amount of data, and actually send and start using the suspicion data (this isn't touched/updated when suspicion is disabled) when suspicion gets enabled by the user.