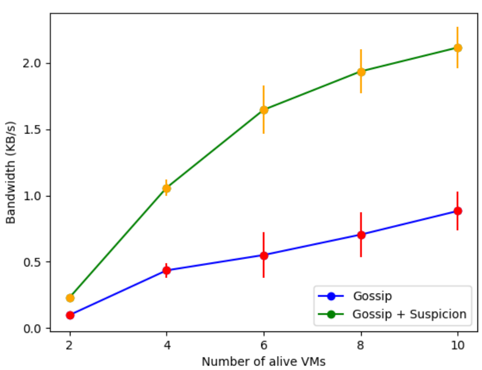
CS425 MP2 Report

Yuechuan Zhang (yz134)
Xinshuo Lei (xinshuo3)

Design

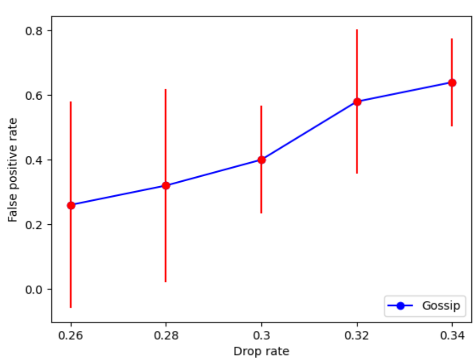
At each server, we maintain two threads, a heartbeat receiver thread, and a sender thread. For the introducer, there will be an extra thread that listens for join requests on a port different from the regular membership port. When a member first joins, it sends a join request via UDP to the introducer and retrieves an initial membership list. The introducer, upon receiving the request, adds the new joiner to its membership list. The heartbeat sender periodically wakes up and sends out local membership lists to a configurable N alive members and the receiver merges any inbound membership lists. The membership list is implemented as a linked list sorted by node ID for the convenience of performing a merge similar to merge sort. A global mutex is imposed on the local membership list for any read and modification to avoid inconsistencies. We also integrated and refactored the distributed grep service (mp1) to allow fetching log files from all remote machines for the sake of debugging. A 5-second completeness is achieved by configuring T_fail to be less than 5 s. We also set T_fail to be greater than log(N) * T_gossip to ensure the latest membership lists have the chance to propagate before entries time out. Finally, we perform a lazy check on membership entry status upon merging, serialization, and "list_mem" to avoid the overhead of excessive scans.

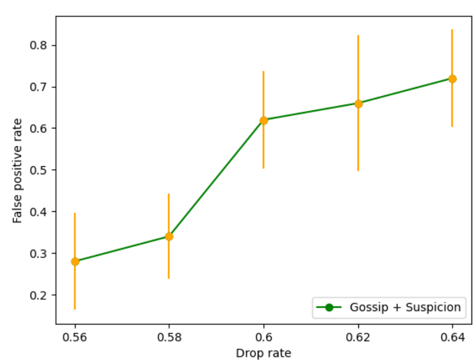
Question 1

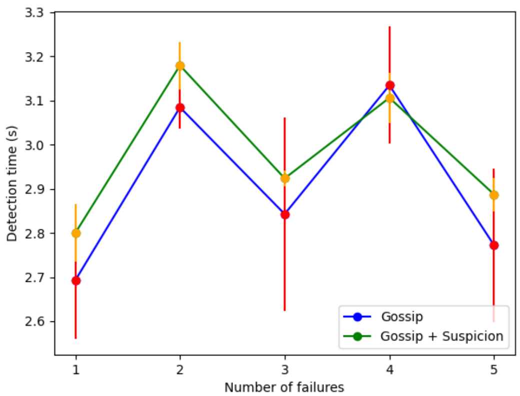


a) Bandwidth usage drops as the number of alive nodes decreases, and Gossip uses half the bandwidth compared to Gossip + suspicion.

b) Gossip + suspicion seems to be capable of tolerating a two-times larger drop rate since it is utilizing twice the bandwidth. We are unable to observe stable false positive cases until increasing Drop rate (Gossip + suspicion) to be around 2*Drop rate (Gossip) .

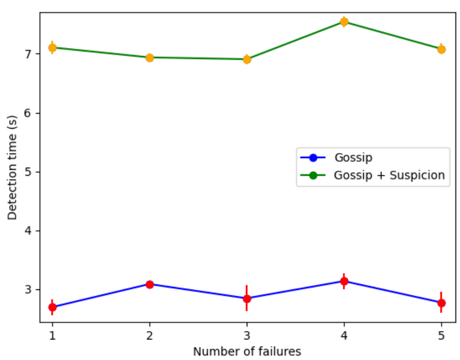




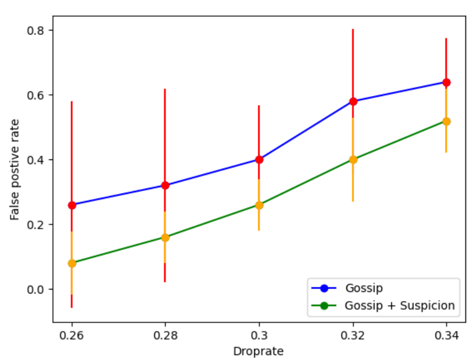


c) Detection time of Gossip and Gossip + suspicion are kept within 5 % of each other.

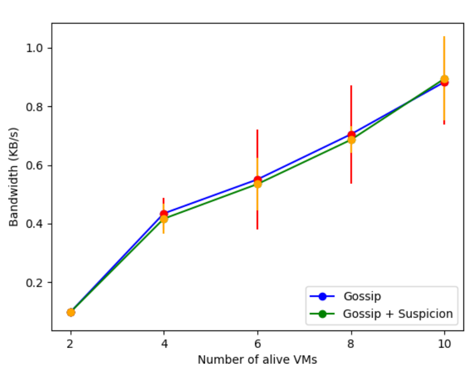
Question 2



a) Under the same bandwidth, Gossip + suspicion generally takes around twice the detection time compared with Gossip protocol.



b) Gossip + suspicion generally gives smaller false positive rates compared to gossip protocol. However this is not very stable under a 10-node cluster setting, since false positive on 1 or 2 nodes can significantly impact the rate of messages received at each node.



c) Bandwidth usages of Gossip and Gossip + suspicion are kept the same (within 5%) by using the same period and #contact configurations.