MP2 Report

Group 52: Shashwat Jaiswal (sj74) and Xinying Zheng (xz112)

Design: Our failure detector is implemented using Java v11. It uses gRPC v1.58.0 for remote execution of its services and Protocol Buffers (ProtoBuf) for data de/serialization. However, we limit the usage of gRPC to the services that require reliable communication like introductions and commands. We implement heartbeats using UDP but couple it with Google's Protocol Buffers for ease of management. We leverage Apache Maven as a package manager for Java for easy cross-platform deployments.

Our package hosts a service, namely HeartService which is expected to run on every node that wishes to join the system. The introducer is contacted implicitly upon instantiation of a node which then facilitates its membership into the cluster. The system starts in Gossip mode by default but can be switched to Gossip+S mode and back using the command utilities. We ensure that our service is as modular as possible for it to be extensible and reusable as it is with minimal efforts and adaptations required. Furthermore, we implement several utility commands like leave, getMembershipList, getNodeId, and switchMode for better abstraction and usability of the system's services. Lastly, the system is configurable using parameters for all practical attributes like T_CLEANUP, T_GOSSIP, T_FAIL, and T_SUSPICION.

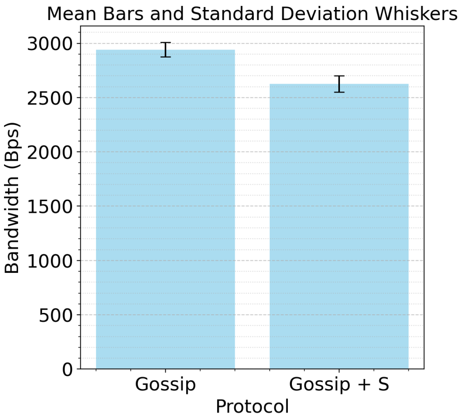
Evaluation: We construct the network across 10 virtual machines and run it in the gossip and gossip+s mode. We plot the bandwidth, false positive rate, and detection time in each mode.

For the first part, which asks us to meet the detection time bound of 5s, we set T_GOSSIP to 500 ms, and T_CLEANUP, T_FAIL, and T_SUSPICION should be log(n) * T_GOSSIP, which is set to 2000 ms. (We assume log n to be 4).

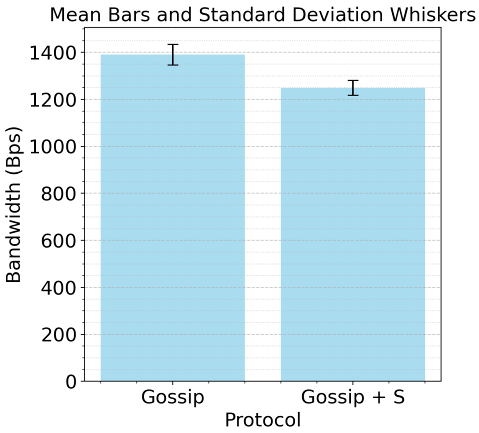
For the second part, to meet the bandwidth requirement (we assume the limit to be 2kB/s), we set T_GOSSIP to 1000ms, and T_CLEANUP, T_FAIL, and T_SUSPICION to 4000 ms. In such a situation, the frequency of gossip is decreased, and the bandwidth is expected to decrease as well.

We observe that:

- Gossip+S mode has a lower bandwidth than the basic gossip mode. A lower gossip frequency also saves a lot of the bandwidth as shown in Q2. From the graph below, we can see that the average bandwidth in part 1 is 2500~3000 Bps while in part2, we only need nearly half of the bandwidth.



Question 1c

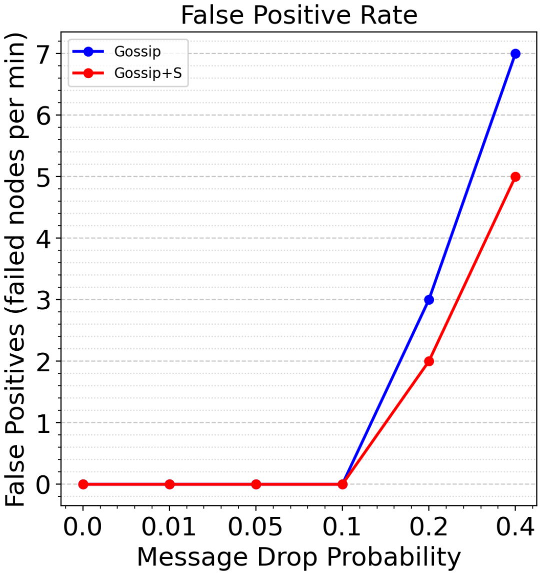


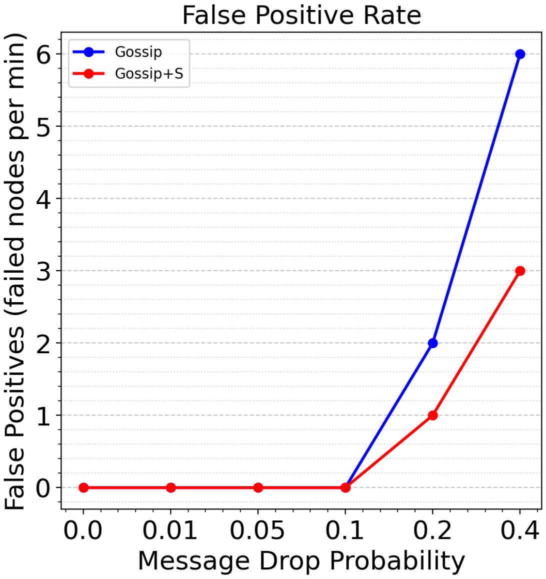
Question 2a

Question 1a

Question 2c

- We define the false positive rate as the number of false positives per minute. Generally speaking, with a higher packet drop rate, the value of false positives also increases. We don’t set the packet loss rate to be higher, since it’s rare in real-world situations. It is important to highlight that the suspicious mechanism can greatly decrease the value of false positives. Our implementation is tolerant upto 10% ıat

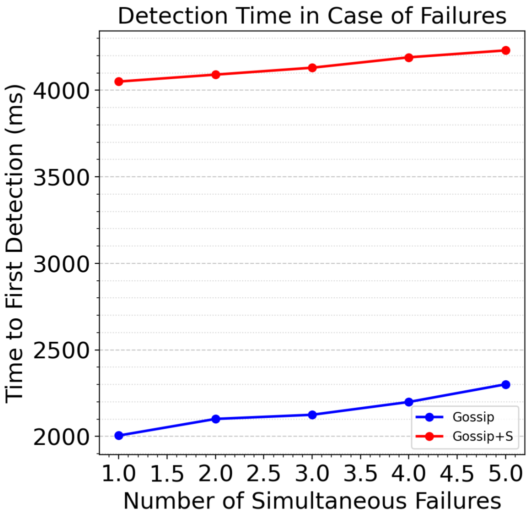




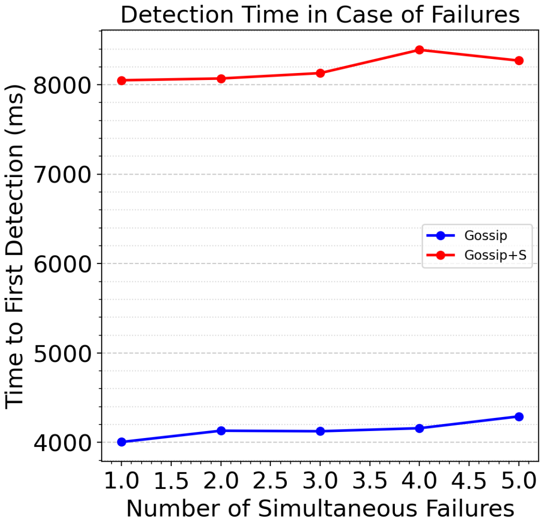
Question 1b

Question 2b

- The detection time and the number of failures show a positive correlation, with more processes crashing, the system needs more time to detect the problem. We also note that in gossip + S mode, the detection time is larger due to the added T_SUSPICION. However, the degradation is not severe.



Question 1c



Question 2a