Joey Stack (jkstack2)
Harsh Agarwal (harsha4)

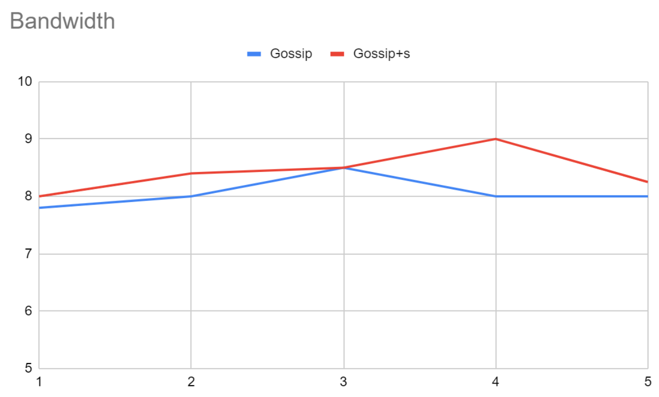
MP2 Lab Report

Introduction

In our MP2, we essentially used multicast to send heartbeat messages to all VMs that are currently running. These servers (one on each VM) then receive this heartbeat and forward the address of the socket that sent that message to the client (if running) on the corresponding VM. The client then uses this address to update the timestamps of nodes (VMS) on its membership list. If it is an address that is not in the membership list, it is added to the group of members and appended to the membership list. Therefore, nodes are able to leave the group (disconnected) and rejoin – as they are appended back into the membership list. If a node has not sent a heartbeat in a certain period of time, the node is declared dead and removed from the list (declared suspicious if running in Gossip+S) mode. In this case, every 6 seconds the timestamps are checked and the corresponding actions are performed then.

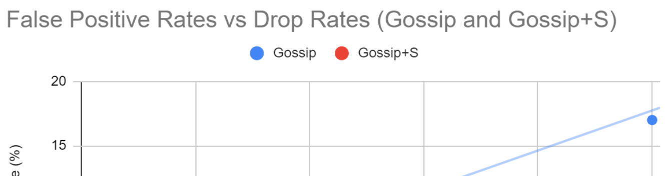
1A) Bandwidth between Gossip vs Gossip+S (no failure scenario)

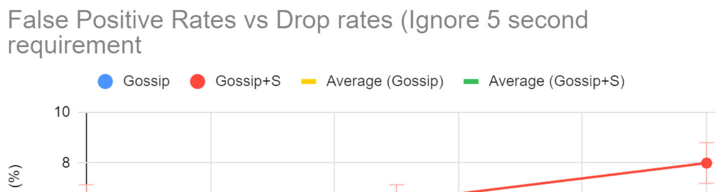
Here we have noticed that when there are no failures, the bandwidth between Gossip+S and Gossip are relatively similar, which makes sense because there were no failures (accidental or induced) in the nodes that could have caused the suspicion mechanism to be used. Therefore, the detectors are essentially the same, so the bandwidth in this scenario should be the same.



1B and 2B) False Positive Rates vs Drop Rates

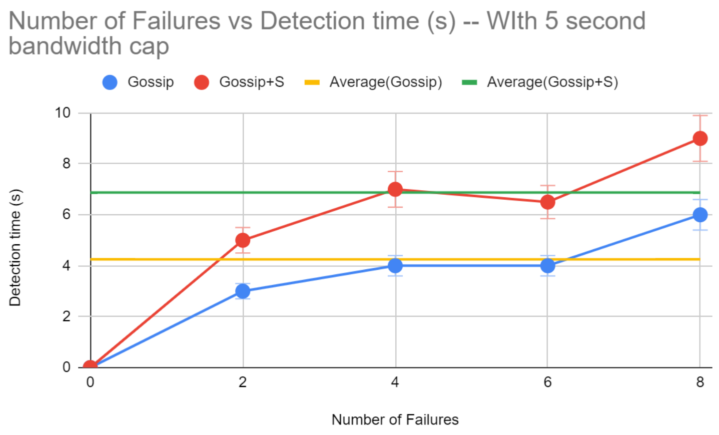
As we can see in the chart below, we observe that Gossip + S has a much lower false positive rate when the drop percentages are increasing. The suspicion mechanism allows our failure detection to be more accurate as instead of removing the node when it appears to have been dropped or killed, it simply declares it as suspicious until confirmed. This occurs in both instances – with the 5 second completeness requirement and without it.





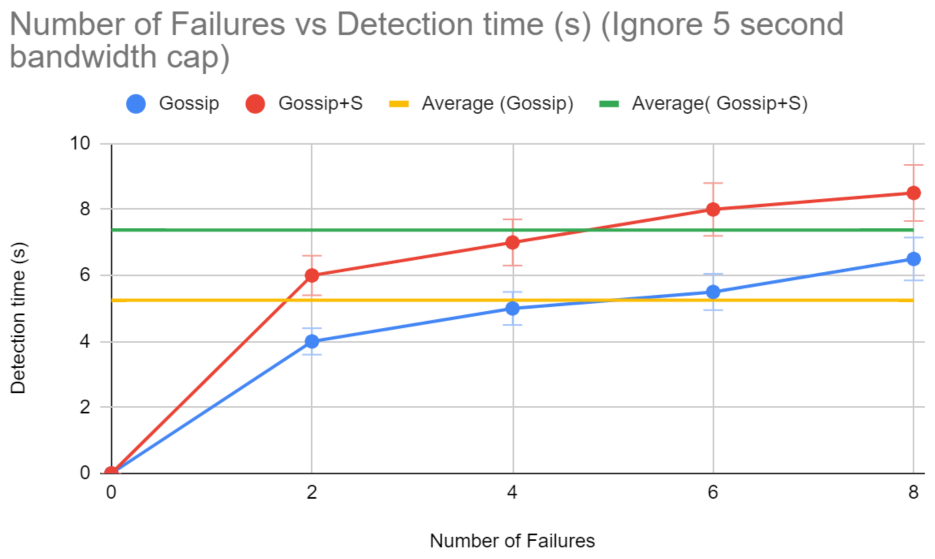
1C) Detection Time to Failure

Here we notice that the time it takes for a failure to be detected is less in Gossip compared to Gossip +S. This is due to the extra confirmation that Gossip+S has to do with the Suspicion Mechanism, while Gossip simply just marks a node as dead. Therefore, it takes much less time to detect failures in nodes.



2A) Detection time vs Failure (Ignore 5 second bandwidth requirement)

Here we notice that again, with the Gossip+S (Suspicion mechanism), the time it takes to detect a failure is greater in Gossip+S compared to that in Gossip.



2C) The bandwidth usages of Gossip and Gossip +S were indeed within 5% of each other.