MP2 Report

Sagar Abhyankar (sra9) & Aditya Kulkarni (aak14)

Architecture & Design Justifications: (further info in readme on github!)

The program is written in Go. Each node maintains a full membership list which is sent by the introducer node when the node first joins the group. The membership lists are implemented as concurrency safe sync Map references, and we also use channels extensively for communication between two go processes. A node communicates only to N/2 other nodes during a direct UDP multicast dissemination to save on bandwidth. The messages used for communication are also succinct in nature. We implemented round robin style pinging, membershipLlist is randomized after each round, to ensure time bounded completeness of 10 seconds, each node pings nodes in the membership list one by one after 500ms, to cover for the worst case which takes 2N-2 rounds of pings periods.

We measure the time difference in terms of when the node was last pinged vs when the node last replied, if it is greater than 5 seconds, then in NONSUSPECT mode we simply send a CONFIRM message in the network to delete the node. However, when in the SUSPECT mode, we wait for another 10 seconds before we CONFIRM that node, simultaneously sending out the SUSPECT node message. During this time if it receives an ALIVE message with a greater incarnation Number, it will remove the node from suspect, otherwise it will multicast a CONFIRM message at the end of 10 seconds. In our design the only the owner of the suspect stores the suspected node in a separate suspect List. All the others simply update the value in the membership list as they personally do not measure the timeout. In both modes, after every addition or deletion to the membership list, the pinging is stopped and the new membership is reshuffled, and subsequently pinging resumes in a round robin fashion on it. In the suspicion mechanism, we observe that a lot of nodes keep sending alive and suspect messages around with increasing incarnation numbers, limiting the false positives to nearly none. In the case of PingAck without S, there are a lot more false positives. The modes are programmatically swapped simply by placing a global variable, which is updated using command line input. We write logs for main events such as node LEAVE CRASH or JOIN. We also used grep frequently to debug the MP2 event logs, since logging to files was more convenient than flooding the terminal, obscuring other important notifications in the terminal. We ran MP1 (5000 range) on different port ranges compared to MP2 (8000 range) to use both of them together.

Graphs:

For the graph for Q1(A), the average bandwidth usage for PingAck+S is slightly higher as expected with additional message passing done (ALIVE,CONFIRM,SUSPECT etc). These messages appear in floods, time and again, but the overall average still remains low. We get around 1 to 1.3KB/s for both variants.

For Q1(B) we induce artificial drop rates, and we see that with suspicion mechanism since the alive/dead decision relies on a single pingtimeout, with increasing message drops the number of false positives vastly increase, on the contrary we see that with suspicion mechanism, the false positive rate is highly under control since the chances of all the alive messages sent by the suspected node dropping are relatively low, and hence it joins back. We tested that at 80% message drop almost 8 nodes are false positive without Suspicion i.e. the whole net dies, but in case of sus mech, it is limited to 1-2 at max.

Q1(C) So generally the detection times (after cleanup) for both were similar, with the suspicion mechanism on the higher side, this aligns with the additional wait duration given after starting to suspect the node and before deleting it. The number of simultaneous failures does not really seem to affect detection time as long as more than half of the nodes are still alive, after which they tend to be erroneous.

For Q2, we limit the bandwidth to 1.2KB/s which is the base bandwidth of operation for the non suspicion mode. The messages were **throttled at VM level** if the bandwidth was going above the limit, that is the messages were **sent eventually** but **were delayed** rather than skipped. Delay was proportional to message size, this allowed us to control the peaks in case of suspicion mechanism (high alive and sus messages flow) and to flatten it. Since there was no time limit on detection, we do not follow the 10 seconds rule.

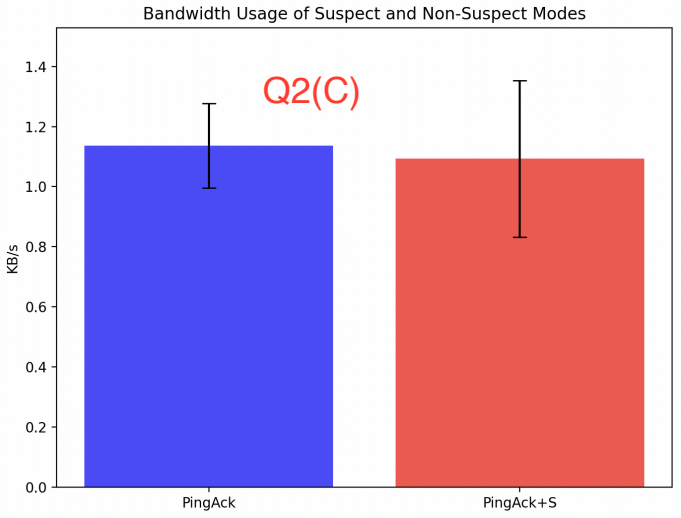
Q2(A) Limiting the bandwidth did not seem to affect the outcomes compared to Q1(C), trends remain the same for the most part.

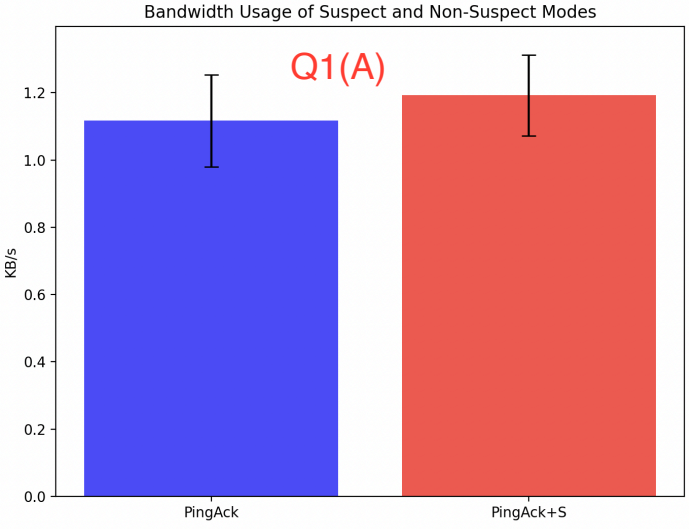
Q2(B) Again, similar results for both the parties, i.e. not much of an effect on false positives trends.

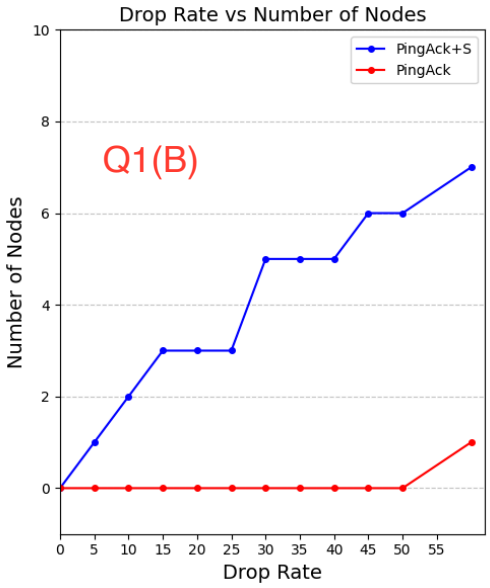
Text

MP2 Report
Sagar Abhyankar (sra9) & Aditya Kulkarni (aak14)

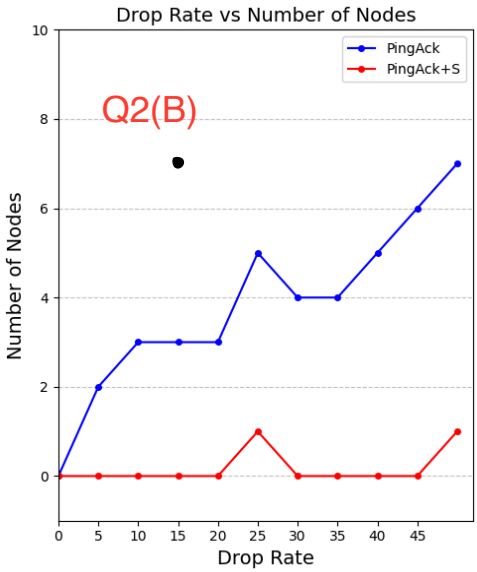
Q2(C) This shows the limited bandwidth below 1.5KB/s for both, PingAck+S the standard deviation shows that it peaks above PingAck when the alive/suspect messages flood.
Discussing on the impact of bandwidth on the operation of both modes, it shouldn't have much impact on the PingAck mode anyway, since consider its normal use as the baseline, but for PingAck+S, the alive and suspect messages are further delayed in some cases, which means it needs more time for detection, which we had increased already, leading to similar results as when bandwidth is unlimited.







Q(2)



Q1(A)