MP2 Report

by: Nicolas Yepes (nyepes2), Ivo Kusijanovic (ilk2)

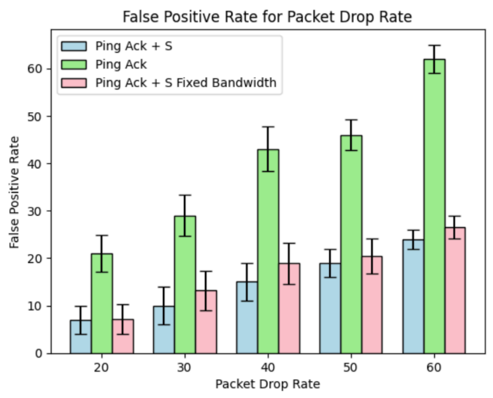
Design: There are 3 parts to our MP solution: the introducer, pinging data, and receiver. Each of this is started on a separate thread since it should not take that much resources to achieve any one of these tasks and require some communication between them. All communication is encoded through json because of its ease of use and clarity. The introducer is a TCP server that will listen for any new members and pass on any information it has about the member to the joining node. Then there is a UDP server listening for ACKs or pings from other nodes, in our system they are treated the same. The pinger will periodically ping a random target from its member list.

Furthermore, we have enabled the ability to toggle between two failure protocols PingAck and PingAck + S. The later marks the node as suspicion when an ack isn’t received, whilst the previous marks the node as failed immediately. For PingAck + S we use incarnation numbers so that we avoid marking suspicious nodes as failed when a new ack is received either directly or disseminated by other nodes.

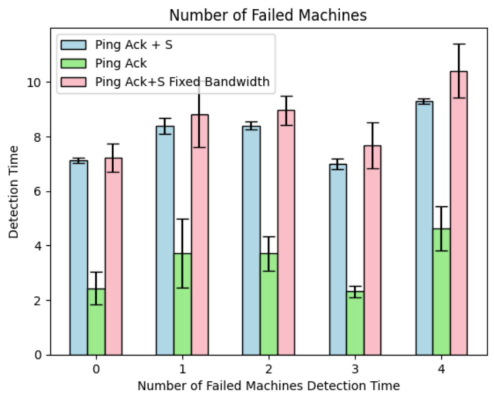
To keep track of events, we created a data structure that works as a dictionary but has a thread that will clean up and remove items after a certain amount of time. We can just send these events to the processes to keep track of any new event and reconcile in the receiver. We did this in order to separate the concern of adding and keeping track of events from sending and receiving data.

To handle changing values and reading values dynamically we created a file that had configuration about the system, before each ping this data is read and if needed distributed through as an event. We suppose this configuration will change frequently so we check frequently for any updates to this file. We could potentially check less to save some clock cycles. Similarly the members currently joined are stored in a file so that other processes can use this information. We assume that this data changes periodically and when it does we want all processes to quickly see the data, so as soon as the members list changes we update the file.

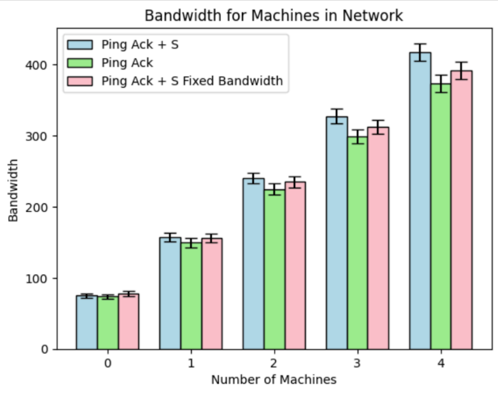
Ping Ack vs Ping Ack +S Analysis: From the bar graph we can see that the false positive rate is much higher for the PingAck than the PingAck + S protocol. This is expected as when we see an acknowledgement isn't sent to PingAck the machine which started the Ping immediately sets the



receiving machine as failed. Furthermore, notice that the false positive rate is basically equal to the packet drop rate, which makes sense as a false positive is detected only when the packets are dropped over the network. Finally, something interesting is that PingAck + S Fixed Bandwidth tends to have a higher rate than its counterpart with unbounded bandwidth. Looking at the false positive rate equation this is perhaps due to less packets being sent, which decreases true negatives and thus increases the false positive rate.



The data makes sense as it takes longer for the PingAck + S protocols to recognize that the machine has failed. This is unlike the PingAck protocol that marks immediately if a node has failed, even if it hasn't. Notice that the time difference is roughly the TTL we add for suspicion, which is the expected. Still, the time is also larger as there is more load on the router due to sending larger messages across the network (read below for why this is the case).



First we can see that the fixed bandwidth is within 5% of both other unbounded bandwidth protocols. Notice that both PingAck + S protocols always have greater bandwidth than the PingAck protocol; this is expected as there are times that this protocol, by our construction, sends a bigger packet disseminating to other nodes in the network that some node is suspicious. Notice that we designed the packet names (see constants.py) so that the status name for a node is readable (e.g. "suspicious" for a suspicious event). We can decrease the bandwidth, by using 1 character encodings. This is a tradeoff between readability and usability.