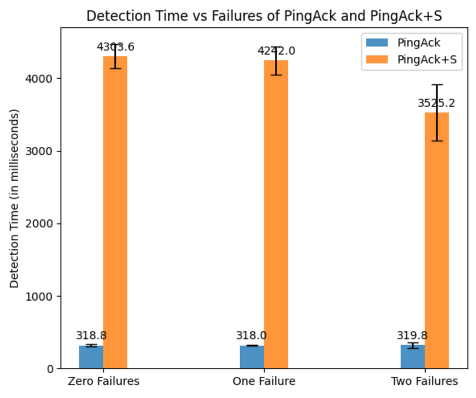
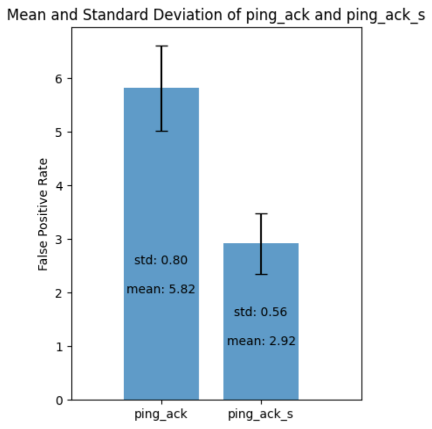
Design:
The SWIM protocol implementation involves three key components: *the introducer*, *the UDP client/server*, and *the membership list manager*.
When a node wants to join the cluster, it sends a join request (via TCP) to the introducer. The introducer responds with the current membership list and assigns a unique node ID to the new node, while also adding this node to its own membership list.
For failure detection, as per the SWIM protocol, the UDP client randomly pings one UDP server/process from its membership_list during each protocol period. Along with the ping, the client sends a list of recent changes it has made to its membership list, referred to as the change_map. The server process responds with an acknowledgment and its own change_map. Whenever a process receives a change_map (either from a ping or an acknowledgment), it invokes the membership list manager to update its membership list accordingly.
The membership list manager handles reconciling all network changes received as gossip from various change_maps. Based on these updates, it creates its own change_map, which is gossiped throughout the network after a specified timeout. To prevent network overload, old changes are eventually purged from the change_map.
The membership_list manager is responsible to increase the incarnation number when it detects that the node is put in suspicion by other nodes, it detects this from the change_map it receives from other nodes.

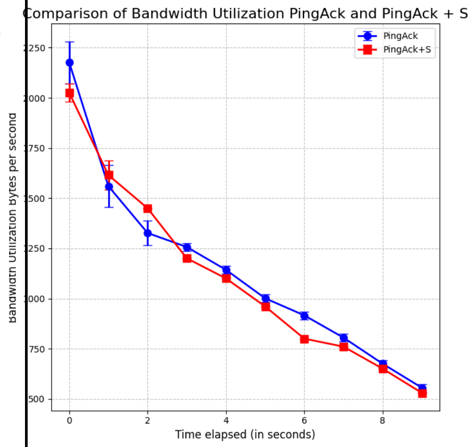
Protocol period calculation:
To ensure that a failure is detected within 5 seconds, the worst-case scenario requires a detection time of (2N - 1) protocol periods. For a 10-node cluster, this gives us a protocol period of approximately 0.26 seconds. This ensures that the first failure is detected within 5 seconds.

Once the failure is detected, the information will propagate through the cluster via gossip. With a gossip fan-out of 1, the worst-case time for the failure to spread to all nodes is T' (protocol period) multiplied by N (number of nodes). In this setup, this is 0.25 seconds * 10 = 2.5 seconds. Therefore, in the worst case, every node will have the updated membership list reflecting the failure within 7.5 seconds after the failure occurs.

False Positives Introduction and Calculation:
We introduce false positives by generating a random number between 1 and 100. If it's below the cluster's drop probability, the server drops the acknowledgement, simulating network delays. The false positive rate is calculated by dividing the number of failed machines by the total pings sent, as no processes are intentionally killed.







1b and 1c) In this graph, we see the difference between Ping-Ack and Ping-Ack with suspicion. Ping-Ack + S as we know gives every process a second chance before marking it as dead, which helps in reducing the false positive rate (for false probability set to 5%). And lead to substantial increase in the detection time. The detection time decreases as the number of nodes reduces, as the number of protocol periods needed for failure detection reduces.

1a) As the cluster is just starting, the number of nodes joining increases and the change_map becomes larger. This leads to an increased bandwidth. As the cluster converges, the size of change_map decreases to near 0, leading to reduced bandwidth.

2) By relaxing the 10-second completeness requirement, we extend both the protocol period and the ping request timeout.

2a) This extension results in a longer detection time, as the protocol period increases. However, a healthy process can still independently detect the failure of another process that fails simultaneously. Interestingly, we found that the time to first failure detection does not depend on the number of simultaneous failures.

2b) Ignoring the 10-second detection constraint allows us to increase the timeout. As the timeout increases, the false positive rate decreases because delayed packets, often caused by network congestion, are more likely to be accepted. However, the false positive rate is not entirely eliminated, as there remains a chance of packet loss. With an extended timeout, we observed a drop in the false positive rate from 6% to 4.2%, with a 5% false positive probability initially.

2c) The bandwidth usage did not change significantly, though there was a slight increase in cases involving ping-ack + S. This increase is due to more nodes flipping from suspect to alive, leading to a higher number of gossip messages circulating within the network. Bandwidth usage for ping-ack - 492 b/s and ping-ack+S - 505 b/s.