Group 70 MP2   Member: Wenjie Sun (wenjies3), Hung-Jui Chen (hungjui2)

In this MP, we implement SWIM with time-bound completeness and suspicion (optional) that uses broadcasting as a method of dissemination. We use vm01 as introducer. Messages are marshaled into JSON format, which is platform independent. Messages are transmitted using UDP only. We output our logs to stdout, which can be directed to a log file and be grep by mp_1.

To run:

python3 start_server.py –node-id 01 –is-introducer
python3 start_server.py –node-id XX

Time-bound for Pingack (+ sus):

According to the SWIM paper, we need to ensure ceil(1/(1-(1-0.1*0.7)^9)) = 3 cycles to expect at least one non faulty node can reveal the failing node. Therefore we set our cycle interval to 1s, which leaves 2 seconds for timeout and suspicion (to hit the 5 second mark).
In pingack without sus, timeout can take the entire 2 seconds. In suspicion mode, timeout and sus each take one second.
Since we are using broadcasting, the additional time needed for completeness is O(1), we are well within the 10 second completeness mark. Bandwidth is shown in later graphs.
Timeouts are done concurrently so that only the last ping cycle's timeout should be considered.

Pingack logic:

Ping messages include the suspicion status of node_j to inform j that it is being suspected. This feature is also included for pingack without suspicion. You can say that pingack is a subset of pingack + suspicion.

Without suspicion:

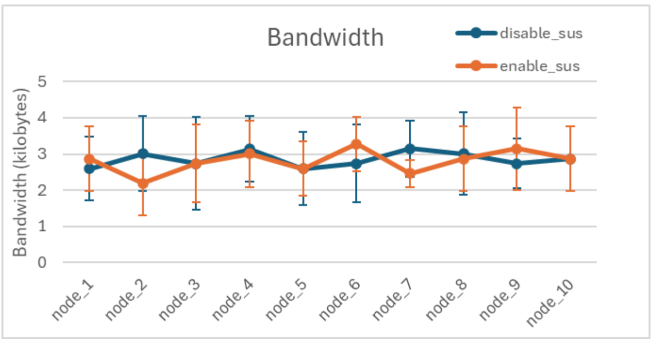
A node_i sends a ping to node_j and timeouts if ack is not received.
If timeout, node_i broadcasts the failure of node_j, and each node who receives the message will handle it accordingly.

With suspicion:

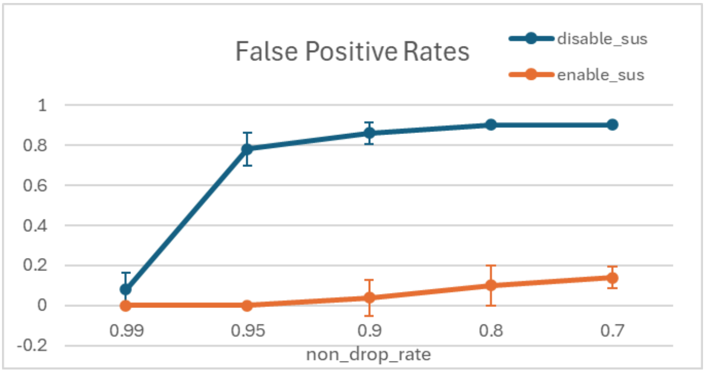
Incarnation overriding rules are the same as SWIM paper.
A node_i pings node_j. If node_i timeouts when waiting for the ack, node_j will be marked as suspicious in node_i's membership list and broadcast to the entire network. All nodes who suspect node_j will start a suspicion timeout. After timeout, if node_j is still in suspicion status, it is marked as failed.
When node_j realizes that it is being suspected, either through broadcasting or pinging, it increments the incarnation number. If it receives via broadcast, it broadcasts a message back saying it is alive, with a higher incarnation number to override the previous suspect. Otherwise, the pinging node is responsible for broadcasting the alive message.

Q1. To reach the same detection time, we drastically decrease ping timeout in suspicion mode (-1.5s) which leaves us with a total of 0.5s for ping timeout, which is a bit short. The performance of suspicion should be better under normal parameters.

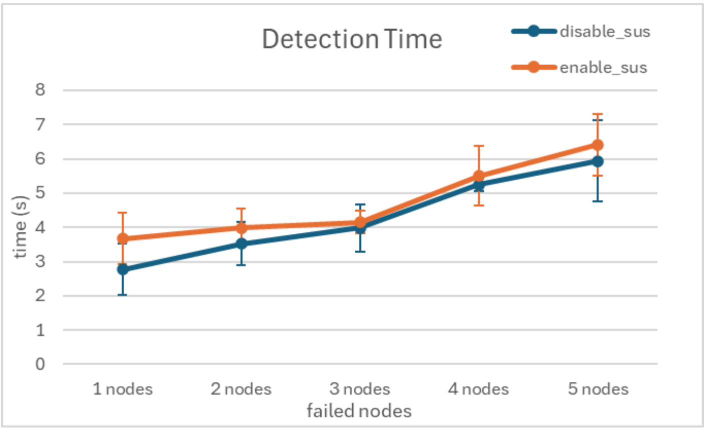
a. Each datapoint is the total bandwidth of a node, sampled every 2 seconds across a 10 second test period. Total bandwidth across all nodes are: 28.566(disable_sus), 28.02(enable_sus) kilobytes with 1.91% difference.



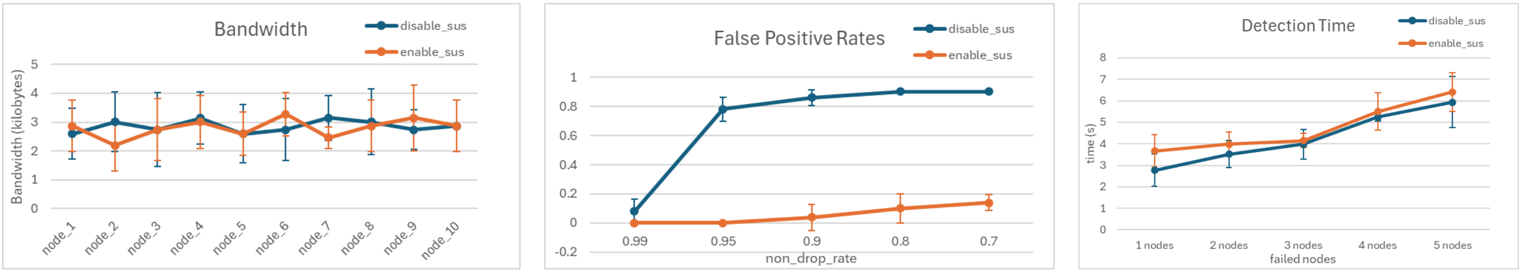
b. The graph below shows the false positive rates(FPR) at different non_drop_rate (0.99,0.95,0.9,0.8,0.7) for both disable_sus and enable_sus modes. The calculation represents the average rate of node failures over 20 seconds. The FPR with enable_sus is significantly lower than that with disable_sus.



c. When we determine the same detection time for sus and no_sus, we use the 3 failing nodes scenario to see whether the 2 have identical detection time. 3.976454128s (disable_sus), 4.154(enable_sus)



Q2. The bandwidth for ackping and ackping + sus in our implementation in the 0 message loss scenario is identical, as shown in the first question. So all results from Q1 apply to Q2. Bandwidth is measured using iftop.



Conclusion:

Suspicion mechanisms produce considerably lower false positive rates under same base bandwidth and similar detection times. However the load on suspicion includes at least one additional round of broadcasting, which is not reflected in the base bandwidth. The base bandwidth itself also exhibits high variance given the fact that a node can be simultaneously pinged by other nodes and pinging other nodes. Suggesting that we should leave some tolerance so max bandwidth is not reached, which could result in packet loss. Detection times also suffers from high variance, likely due to this random pinging and unforeseen suspicion events (for example, we are crashing nodes 2, 3, 4, but nodes are suspecting node 9 because there's a packet loss. This will increase message load and processing time). Poor synchronization practices like using only a few locks for lots of shared data could also be the culprit.