Distributed Systems (CS 425) - Fall 2024 - MP 2 Report

Akhil Sundaram(akhils7), Anurag Choudhary (anuragc3)

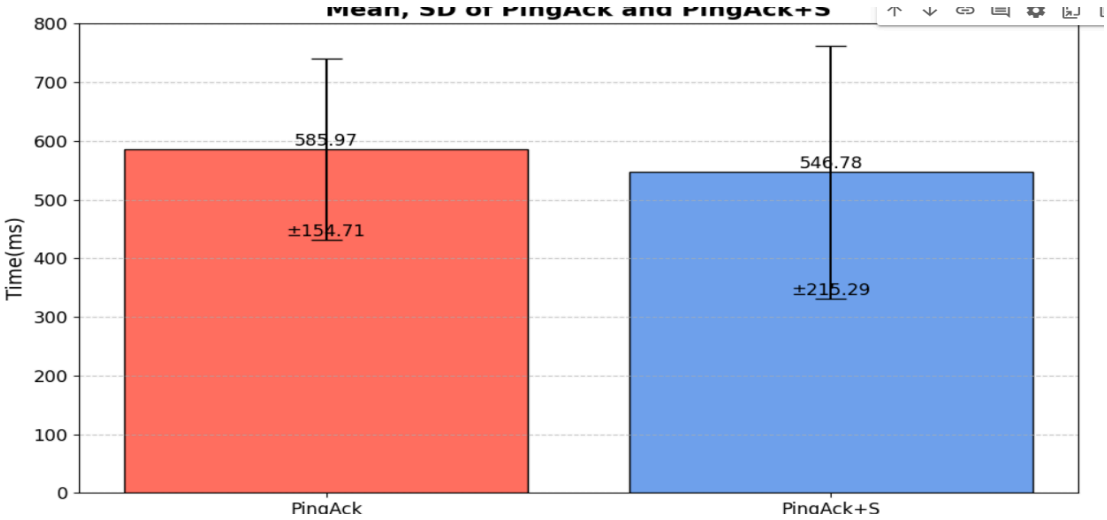
Design Overview: The distributed group membership algorithm implemented is based on the SWIM protocol. Our system consists of two main components: an introducer and a group membership program.
The introducer is a designated virtual machine (VMs) in the network. When a Machine comes up it checks its IP/hostname to determine if it's the introducer. All new joins to the network are through the Introducer. It has a TCP listener to accept join requests, which returns the current membership list. A newly joining node will send a TCP connection to the introducer. The introducer then (1)adds the node to its membership list, and (2) adds the new join information to its buffer. The values in this buffer are then disseminated to other members in the membership list.
We randomize the membership list, and iterate through it sending each node in it to send a UDP request. The request also carries some buffer data with membership information. It then waits for around 2 seconds (our timeout) for the node to reply. If the node timeouts or refuses the connection, we mark the node as failed. It is then put in the buffer, and also removed from the membership list. In the buffer/piggybacked data to its next ping, it sends this failure information. We also receive piggyback data from the response of the ping, which we process to update our membership list. We also selectively ignore buffer information based on current membership and buffer data. Once we iterate through the list, we again randomize membership list order and iterate through this newly random list.
Each Membership list element has hostname, memberID, and incarnation number. The memberID is a concatenation of host ip address, port, and timestamp. This is created at the node join request. Our Buffer with elements each also has a count denoting how many times it is disseminated in the network - times sent. It also contains the type of message, and target host/member. After a set amount, the buffer element will be deleted from the list and won't be disseminated again. Our current value is set at 4. To prevent network flooding, we implemented a check to avoid duplicate messages in the buffer, using node IDs as unique identifiers.

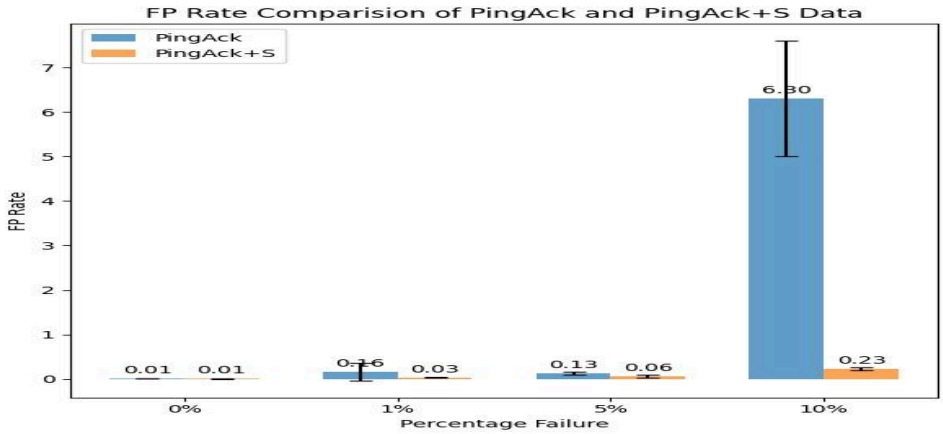
Suspicion Mechanism:

When suspicion is enabled, we manage the state through an adjacent membership list. We keep track of incarnation-number and states. When we receive a list of buffer elements, we process each element based on the SWIM priority. The states we maintain are mainly Alive and Suspicious, as we just remove the host at Faulty. Initially no member has any suspicion which can be treated as a State as well. On marking a node as suspicious, we called a timed function that executes after a timeout (5 secs). The function checks if the node is still in a suspicious state and not alive. If yes, it declares node as faulty and adds that to the dissemination buffer. If a node sees a suspicious state for itself it adds an alive state value to its buffer and sends that out. It also increases its incarnation number.
Important Note: Once a node is marked as "Failed", it cannot rejoin the network without a complete restart and reintroduction.This design ensures a more robust and fault-tolerant system, reducing false positives in failure detection while maintaining the efficiency of the SWIM protocol. Since we don't have to send the complete membership list every time, we reduce the bandwidth usage.

MP1 Usefulness: MP1 was used to grep the logs that are generated on the VM locally, and search for specific messages. This was useful to debug during development and to calculate the metrics below. Also, we reused parts of MP1 for introducer logic.

Metrics:





	Num fail 1	Num fail2	Num fail3	Num fail4	Num fail 5
Avg Det time with PA(ms)	1641	834	1192	789	657
Avg Det time with PS(ms)	5425	6893	5130	5916	5408

(2)
(i) The detection time is similar as we increase the number of failures,
(ii) False positive rates increase as we increase the number of message drops. It seen quicker as in just Ping as it fails instantly compared to Ping+S
(iii) Average bandwidth rate for Ping = 540 Bytes/sec, Average bandwidth for Ping+S = 568 Bytes/s