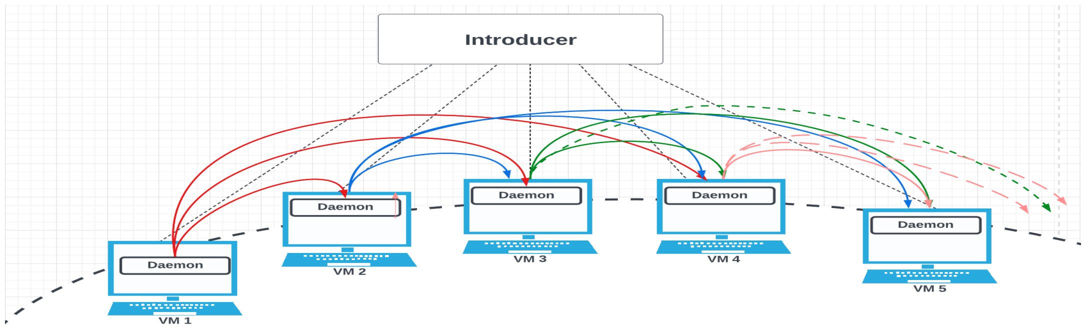
CS425 MP2 Report

Girija Manoj(gmk6), Santosh(santosh8)

Design:

Our failure detector organizes the processes into a ring structure. Every process in the ring will monitor the three active processes after its position in the ring and pings them every 1s. Every process will be monitored by its three active processes before it. As TCP is reliable, we use it for Introducer communication and UDP protocol for every other communications.
Note: This design is based on the assumption that at max 3 simultaneous failures occur.



Algorithm: We use the Ping-Ack mechanism, the Ping interval is 1s and Timeout is 2s. (SWIM-style but without any indirect pings).

Process Joins: When a process joins the ring, it first sends a **JOIN message** to Introducer, then Introducer then assigns **a process id (hostname + timestamp)** and shares the existing membership list to the new process and then notifies the rest of the system about this new process and topology gets readjusted.

Process Fails:
If process "A" does not get the **ACK** from its neighbour process "B" in the timeout period, then A marks B as fail. Then "A" sends a **fail message** to the Introducer and to 3 of its active predecessors, which then propagates to others and the topology gets reorganised.

Process Leaves: When a process leaves, It first sends a **leave message** to the introducer and then to its monitoring processes, which then disseminates. Each process updates its membership list accordingly and the topology gets adjusted accordingly.

Scalability:

This design can easily be scaled to N processes, at any point in time each process will be monitored by at max 3 processes and also each process can monitor at max 3 processes. So the load on a process is constant and independent of N.

Marshaled Message Format: We used JSON to marshall the message.

Message = { **"TYPE":** JOIN, PING, ACK, LEAVE, FAIL,
"DATA": Different payload as per the TYPE of the message. }

For join, fail and leave messages the data of the message includes the ids of the joined/failed/left processes.

Why it satisfies the three completeness requirements?
Let's say the processes A, B, C, D, E, F, G, H, I, and J(same order) are active in the ring.

a) **Meets 5-second completeness.**

Let's assume process D fails, Since A, B, and C sends pings to D every 1s. They will not be receiving the ack from D and as it timeout after 2s, they mark D as a failure. Hence the failure is detected first in 2s, which is less than 5s. **(time-bounded completeness)**

b) **Meets completeness up to 3 simultaneous failures**

If D, E, and F failed simultaneously, failure of D can be detected by process A/B/C, failure of E can be detected by B/C and failure of F can be detected by C. So every failure is still being detected. So the system is complete up to 3 simultaneous failures.

c) **Would violate completeness if 4 simultaneous failures.**

Let's assume D, E, F, G are failed simultaneously, then no active process can detect the failure of G, hence completeness is violated if 4 processes fail simultaneously.

Usage of MP1(Distributed grep) in MP2: We used logged the errors of MP2 programs into a file. We then used the distributed grep of MP1 while debugging our programs in MP2. It was very useful and saved us a lot of time rather than looking into each log file on each VM.

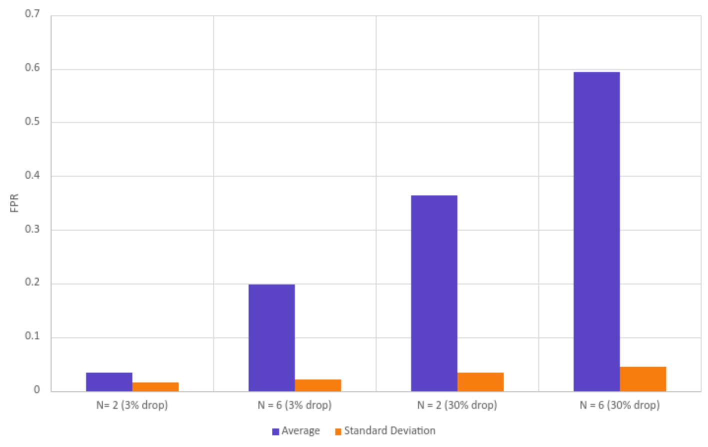
Measurements:

1. The background bandwidth usage for N=6 is **1800 Bytes/sec(B/s)**

2. The average bandwidth usage whenever there are 6 nodes in the system = 1800 B/s. During a Join it's 2552 B/s, during a Leave it's 2167.3 B/s, during a Failure is 2166 B/s and after Join it's 2100 B/s, after Leave it's 1500 B/s, after a Failure it's 1500 B/s. The average during Join, Leave and Failure will increase as expected since the extra information will be propagated in the system.

3. We measure the false positive rate by running the system and counting the number of times a machine is identified as failed eventhough it is still active. To simulate the message loss we randomly discarded ping messages (to match 3% and 30%). We calculated the following metric based on 5 observations.

	Packet Drop 3%			Packet Drop 30%		
	Average	Standard Deviation	Confidence intervals (95%)	Average	Standard Deviation	Confidence intervals (95%)
N = 2	0.03374	0.01615603	[0.0113, 0.0607]	0.36316172	0.034555422	[0.307, 0.339]
N = 6	0.19869486	0.0203421	[0.183, 0.214]	0.5933823299	0.044742856	[0.554, 0.633]



Observation:

- We can see from plot that, as the FPR increasing as the packet drop rate increases. This is expected, as ping and acks will be dropped and processes will marked as fail wrongly.
- As the number of processes increases, the no of pings and acks will increase as well as the packet drops. So for N=6 the FPR is high compared to N=2.