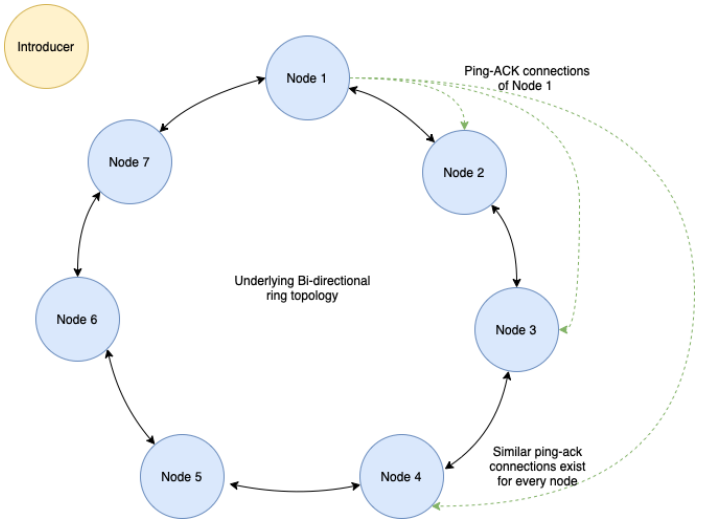
Machine Problem 2: Distributed Group Membership System

Group 65 : Aruna Parameswaran (netid: aruna2) ; Sanjit Kumar (netid: sanjitk3)

Design:

The distributed group membership algorithm is designed with 2 programs - an introducer program and a group membership program. The introducer program is a separate process running on one of the VMs to introduce new nodes to the network. The group membership program contains logic for the unidirectional ring topology of the network (for message dissemination) and the configuration (topology) of monitors for failure detection over the entire network. Each process's id is created from its ip, port and time of join.



Each node contains a full membership list. While socket connections are made only to its direct neighbors (ring network topology), failures detected are disseminated through the network ring. To account for up to 3 simultaneous failures, each node in the system monitors 3 of its neighboring nodes in the clockwise direction by directly pinging them and waiting for an ACK. Each node on receiving the ACK updates the last_updated_time in the membership_list and if it's from a new node, updates the membership list. Each node also sends its corresponding 4 monitors its own ACK for their pings and leaving_info at time of voluntary leave.

We have integrated the JAXB (Java Architecture for XML binding) format for marshaling membership data between two nodes. Before sending a packet, the sender node uses JAXB to marshal the member list Java object into a string of XML format, and the receiver node unmarshals the packet before reading it. This helps to ensure that the data is not erroneously modified as a result of differences in the underlying hardware between nodes. We chose XML as it is easy to parse, light weight and ubiquitous.

a) Our system meets 5 second completeness because each node takes a total timeout of 2s + cleanup of 1s (totally 3s) to detect a failure and disseminate to neighbors. The ping frequency is 1s. If there are 9 machines in the system, within 3 ping cycles, all of the members are notified of the failure.

b) The system can be easily upscaled to a higher number of nodes because the overall throughput in the network is pretty low due to ACK messages being relatively low weight compared to sharing membership lists.

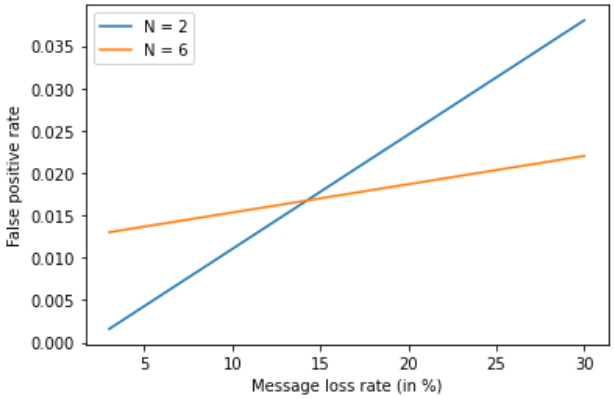
c) In the event that 4 neighboring nodes fail together (like node 1,2,3,4), the first clockwise node failure won't be detected by the system since all 3 of its monitors have also failed.

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.

Metrics: We used UDP's packet getLength() function to measure the average message length.
(i) Each member sends and receives PINGS from its 3 neighbors, and replies to each with an ACK, So in 1 unit of time, each member sends 6 messages in total, with a total of 36 messages being exchanged. The average size of each PING message was measured to be 51 Bytes. The total bandwidth usage is = (6*6*51) = 1836 Bps (~1.8 KBps). This is roughly 306 Bps/node.

(ii) When a new node joins, the full membership list is sent to the new node by the introducer (245B). In addition, the new node starts to communicate with its neighbors, with a total of 7*6 messages. B/w usage is (7*6*51) + 245 = 2387 Bps (~2.4 KBps) or 340Bps/node.
When a node fails, its failure is detected and the updated member list is disseminated by its neighbor (254B). The total b/w usage is (5*6*51) + (5*3*254) = 5.34KBps, ~1068 Bps/node.
When a node leaves, it handles spreading the leave message to neighbors. Total b/w usage is (6*6*51) + (6*3*239) = 6138 Bps, ~1023 Bps/node.

(iii) False positive rate $False\ Positive\ rate\ = \frac{FP}{FP + TN}$



	0	1	2	3	4	mean	std
n=3,3%	0.0014	0.005	0.0002	0.0005	0.0010	0.0016	0.002
n=3,30%	0.0260	0.010	0.0670	0.0208	0.0670	0.0380	0.027
n=6,3%	0.0150	0.015	0.0130	0.0130	0.0100	0.0130	0.002
n=6,30%	0.0278	0.011	0.0167	0.0278	0.0278	0.0220	0.008

The reason we believe why N = 2 shows worse FP rate is because in an highly unreliable medium (loss rate ≈ 30%), and when there are only 2 nodes the chance of both ACKs of the nodes being dropped for each other is very high. In such a case both nodes label each other to be failed and terminate the program. This we believe is the reason for bad behavior for low ranges of N and high ranges of Message loss rate. On the other hand, we think that is why higher N values show a more stable FP rate. The FP rate is acceptable and stable with a reasonable slope for N=6.