MP2: Distributed Group Membership
Group 38: Rahul (rahulsk2) & Venkat (vn7)

1) Design:

a. Algorithm Used:
The group membership protocol works by organising the nodes in a pseudo ring topology. Each Node pings its two predecessors and one successor. Because it is guaranteed that we can have at the most 3 simultaneous failures, it can be guaranteed that every failure will be detected by at least one node. Each node maintains a membership list of all nodes in the network which is sorted by the Node ID (Port_IP_Timestamp). Following is how the major operations work.

Node Joins:
The node sends a JOIN message to the Introducer (hardcoded to machine1), The introducer sends an INTRODUCE message to every node in its membership list. On receiving this message, every node in the n/w adds the new node to its sorted membership list. Introducer then adds the new node to its own membership list and sends back a JOINACK message which contains the current membership list in the Payload. The new node uses this to update its own membership list and decide its neighbours.

Node fails:
The Node is being PINGed to by 3 of its neighbours, any or all of these can detect the failure when the node does not send an ACK back to a PING message. When this happens, the detecting node sends a FAILURE message to its connected neighbours, which gets propagated along the network. On receiving a failure message, a node will remove the failed node from its local membership list.

Node leaves:
Node leaves is modelled just like a Node Fails, except that in this case the leaving node sends a LEAVE message to its neighbours which then functions in the same way as a FAILURE message.

Additional cases handled:
The introducer can fail and restart. In order for the introducer to build its own membership list during restart, it sends an INTRODUCE for itself to each node in the N/W and adds any node that sends back an INTRODUCEACK

b. Why it scales to large N
The network dynamically reorganises according on failures and additions, moreover any node is only pinging and getting pinged from 3 other nodes irrespective of the size N. Moreover, on joins and failures, the entire membership list is not propagated across the nodes, but just the ID of the concerned node in the payload.

c. Marshalled Message format
We use a JSON message format
Message{
    ID: “PortNumber_IP_Timestamp”,
    MessageType: PING/ACK/JOIN/JOINACK/INTRODUCE/INTRODUCEACK/FAILURE/LEAVE
    Payload: Depends on message type
    Additional Data: Used to send membership list to a newly joined node
}

d. MP1 in MP2
During the development and debugging phase it was really useful to have integrated the distributed log querrier from MP1 into this MP. We had run with some off by 1 and edge case errors that would have been non-trivial to catch as they would manifest only when a larger number of nodes were connected to the network and the MP1 grep helped us a lot in this debug phase to figure out the error.

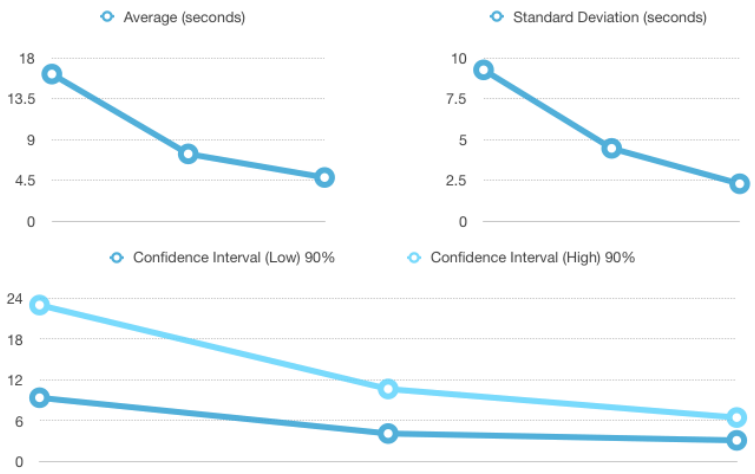
MP2: Distributed Group Membership

2) Measurements

a. N = 2 machines

Stable bandwidth: 300 bps
Join: +270 bytes
Leave: +131 bytes
Failure: +103 bytes

False Positive Rates are plotted in the graphs

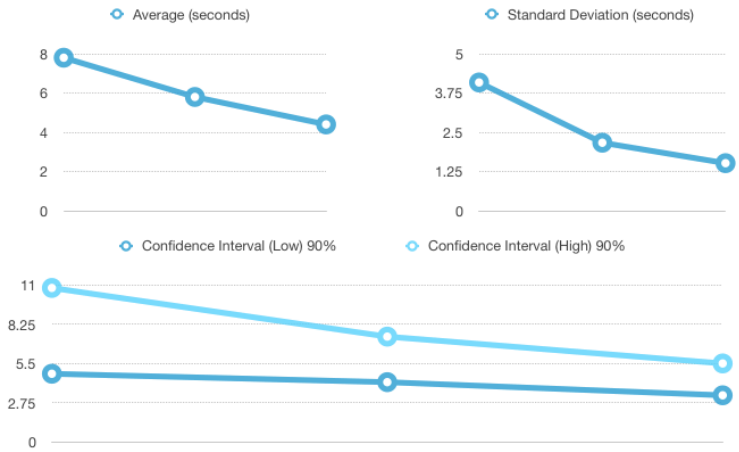


N=2 Machines

Packet Loss	R1	R2	R3	R4	R5	Average (seconds)	Standard Deviation (seconds)	Confidence Interval (Low) 90%	Confidence Interval (High) 90%
3%	30	20	9	7	15	16.2	9.25742944882649	9.38962053186462	23.0103794681354
10%	10	13	2	4	8	7.4	4.4497190922574	4.12649133191922	10.6735086680808
30%	6	4	2	8	4	4.8	2.28035085019828	3.12242258002797	6.47757741997203

b. N = 4 machines
Stable bandwidth: 1700 bps
Join: +250 bytes
Leave: +360 bytes
Failure: +330 bytes

False Positive Rates are plotted in the graphs



N=4 Machines

Packet Loss	R1	R2	R3	R4	R5	Average (seconds)	Standard Deviation (seconds)	Confidence Interval (Low) 90%	Confidence Interval (High) 90%
3%	15	6	7	5	6	7.8	4.08656334834051	4.79365279782923	10.8063472021708
10%	7	9	4	5	4	5.8	2.16794833886788	4.20511332691003	7.39488667308997
30%	5	4	2	6	5	4.4	1.51657508881031	3.28430671777589	5.51569328222411

From these data points, one can see the following:

* As the packet drop rate increases, the false positive happens earlier. This makes sense as a larger packet loss will cause a false positive to become more likely
* The time for false positive to be detected decreases with increase in machines in the network, this is intuitive as more nodes are now ping-acking each other and the probability of such a detection increases.