CS 425 MP2 Report

Karan Sodhi (ksodhi2) & Nashwaan Ahmad (nahmad31)

Design:

For our distributed membership nodes, each node held the full membership list, implemented as a linked list, consisting of member IDs {hostname:timestamp}. Whenever a new node joined the group, its membership ID would be added to the end of the list of all the other nodes. Each node also holds a hash map, mapping its member ID to its corresponding node on the linked list allowing for O(1) deletions on the list.

When a node wants to join the membership group, it sends a TCP request to the primary introducer which would add it to the group and send back the full membership list. The node would then send a Join message over UDP to all the other nodes in the group notifying them that it has joined. Similarly, a node wanting to leave the membership group, sends a leave message over UDP to all other nodes in its list notifying them that it has left. In both these cases the nodes receiving the join/leave message will update their membership list accordingly.

The way the group kept track of failures was that each node would ping its subsequent 3 neighbors, if possible, in the membership list every 5 seconds, looping around if necessary, thus forming a virtual ring. When a node is pinged it sends back an acknowledgement message over UDP notifying the sender that it is still alive. If the pinging node did not receive the acknowledgment within 1 second of sending the ping it would mark the node as failed and send a UDP message to all other nodes in the group notifying them of the failure which would result in the node being removed from all of the membership lists. Since the node is being pinged every 5 second a machine failure will be reflected in at least one membership list within 5 seconds.

This meets completeness upto 3 failures because if 3 adjacent nodes failed simultaneously, the failure of all nodes would be detected because there is at least 1 monitoring node that is still running for each failed node. However, if 4 adjacent nodes failed simultaneously then 1 node may not be detected as having failed because all nodes monitoring it have failed.

Additionally, if the primary introducer goes down, it has the ability to rejoin the group because it stores logs of the 3 most recent nodes that have joined and will treat one of these nodes as the introducer in order to rejoin the group and re-populate its membership list.

The algorithm scales to large N because even if there are many members, we are able to delete and add members in the local membership list in constant time. Additionally,

each node only has to receive and send pings from at most 3 nodes and when a new node leaves or joins only the information about that node is sent through the network instead of the entire list.

Marshaled message format:

The messages sent are platform-independent because they are sent using ASCII strings in this format {Letter Corresponding to message}/n{Payload}. For example, a leave message is L/n{Membership ID of leaving node}

MP1 Integration:

Whenever a node joined, left, or a failure was detected we logged that information in a file. Using the MP1 querier we were able to easily check the logs to make sure every node joined and left as we expected it to which helped us a lot when debugging.

Average Bandwidth Usage:

For N=6 machines, the background bandwidth usage for a single node is 924 bytes/second and for the whole group is 5.544 kb/second, measured using iftop. Whenever a node joins, leaves, or fails bandwidth increases by 320 bytes in each node for a total of 1.244 kb/second for the node or a total of 7.464 kb/second for the whole system. This makes sense because in each case an extra packet will be sent with the same number of bytes detailing what happened to the node. Additionally when a node joins there an additional 906 bytes/s of bandwidth is sent through the introducer since it has to set up a TCP connection with the joining node and send it the entire membership list, therefore a total of 8.37 kb/seconds is sent through the group during a join.

False Positive Rate:

Note: The error bars mark the standard deviation and the 97.5% high and low confidence interval. For N = 2, the false positive rate for a node roughly corresponds with the Packet Loss rate which makes sense because if a packet is lost that means the machine will not receive an acknowledgment from its neighbor and will falsely mark it as failed. Additionally, the false positive rate for a node is much higher when N = 6, which also was expected because now the node has to send pings and receive back acknowledgments from three machines which means there is a greater chance of one of those packets being lost and thus marked as a false positive.