CS 425 - MP2 Report: Group 33

Design:

In this project, we implemented a distributed group membership protocol which maintains at each machine a list of all machines that are connected and up. We have a fixed contact machine called the "introducer". Every machine runs a server thread that continuously waits for incoming messages and client thread that begins by sending a JOIN request to the introducer and in turn receives a message containing its ID (Tuple of timestamp and IP) and current connected members list. The members keep on sending PING messages to maintain their membership list. Whenever they detect a timeout on another machine, they delete that node from their list. They also broadcast this information to all other nodes through FAIL messages. Also whenever a new node joins the group, the introducer assigns it an ID. It also sends NEWJ (new join) messages to nodes in its membership list so that they all can add the new node in their lists. Whenever a node wants to leave the group, it broadcasts a LEAV message so that others can delete it from their list. The code is fault tolerant - even if introducer leaves, other nodes keep on pinging and detecting failures. When introducer comes back, the other nodes originally present also become part of the same cluster. Introducer failure only limits joining new nodes in the cluster.

Algorithm Used

The nodes in every nodes' membership list are sorted by timestamps. Each node pings machines starting from their next node. For e.g. there are 7 machines, let us call them 1,2,3,4,5,6,7.
Node 1 pings machines in the order: 2->3->4->5->6->7
Node 2 pings machines in the order: 3->4->5->6->7->1
We have kept our ping interval to be 1.5 seconds. For example, Node 1 pings 2, and 2 pings 3. Now, if both nodes 2 and 3 die, then Node 1 can detect these 2 failures in twice of its timeout (ping) interval. We have to detect the failure by 3 seconds, so we kept our ping interval to be 3/2 = 1.5 s.

Scalability: Whenever a new node joins or leaves or fails, the number of NEWJ, LEAV, PING and FAIL messages would increase by just 1. So, it's a constant increase. The total messages vary linearly with number of nodes. Hence, our system is scalable. Our application is fast (detects failure within 3 seconds and disseminates within 6 sec), complete and consistent with very low false positive rate. We took into account various cases involving introducer failure, high network latencies and scalability while implementing the MP.

Logging: At every machine i, we log things to machine.i.log. Using MP1 program (distributed grep) we can query these logs across VMs. MP1 was very useful in verifying correctness, for measurement purposes & analysis, as we did not have to go to each VM to check the logs.

Measurement

The bandwidth of each node is about 34 bps. When there are 4 nodes in the membership list, whole bandwidth is 136 bps. The cost of the bandwidth is mainly from the ping & ack messages.

- When a node joins, the time taken from a new node sending a JOIN request to the introducer node till all nodes have updated its own membership list is 2 ms. The number of bytes transferred in this action on average is 207 bytes. Therefore, the average bandwidth usage is 103.5 bps.
- When a node leaves, the time taken for LEAVE process is 1 ms. The number of bytes transferred on average is 52 bytes. Therefore, the average bandwidth usage is 52 bps.
- When a node fails, the time taken to detect the failure of one node is 1.5s, and the number of bytes transferred on average is 52 bytes. Therefore, the average bandwidth usage is 34 bps.
- For the false positive rate measurement, we implemented a probability function that has x% of dropping the received message. We calculated the FTR as the percentage of node that is mistakenly detected as failure within 10s. We ran for 7 trials per each of six data points below.

We observed that the FTR increases as the package loss rate increases. It is reasonable since when the package loss increases, it is more likely that a node does not receive the ACK as the package dropped in between. The STDEV and CI have the similar trend as the average FRT.