Alex Shen
Tiancheng Wu

CS 425 MP2 Report

Design

The distributed group membership is modeled in a ring where each node in that ring sends heartbeats to two successors and two predecessors. This ensures that up to 4 simultaneous failures with over 5 active machines can be detected completely. Heartbeats are sent out every 0.5 seconds to reduce the potential for false positives, but also not frequent enough to spam the network. To facilitate joining the group membership, an introducer is dedicated to assigning a unique ID to the new node and then disseminates the presence of the new node to all current members through the gossip protocol. Nodes that leave or crash will also have their status disseminated to other nodes quickly.

The algorithm described above scales to large N because each node only heartbeats its surrounding 4 neighbors, and each of those neighbors will do the same such that the entire group will be monitored without having each node communicate directly with all N nodes. The use of gossip to disseminate information is also very efficient because it has a logarithmic runtime complexity and each node only needs to know a few other nodes to gossip to. The total bandwidth needed to disseminate this gossip efficiently is linear in the number of nodes. Since the entire system uses UDP, there's very little overhead in the interactions between node, so that also enables the algorithm to scale to large N efficiently.

The messages are marshalled using the JSON format. This format was chosen because it is very simple to work with and easy to diagnose. The serialization of the data with JSON allows the data to be platform-independent, and the ease of use makes our lives easier.

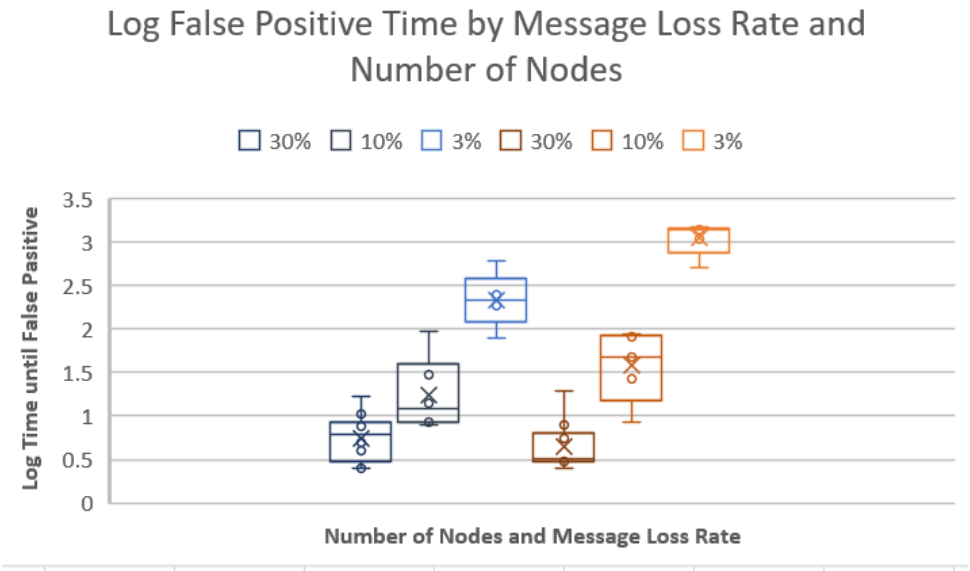
The distributed log querier from MP1 was useful for being able to filter and find errors or failed nodes from all the logs.

Measurements

(i) The background bandwidth for 4 machines is approximately 8KB/second, which was measured as the size of the data sent to 4 other machines twice per second for 4 machines.

(ii) The expected bandwidth usage for the following node operations is as follows:
Join: 67 byte message * 4 neighbors * N machines ≈ 1KB
Leave: 67 byte message * N machines ≈ 268 Bytes
Fail: 67 byte message * 4 neighbors * N machines ≈ 1KB

(iii) The nonzero false positive rates were calculated as the amount of time needed for a false positive failure to occur given the number of nodes and the message loss rate. Due to the exponential scaling of the time it takes to get a false positive failure, the time was scaled logarithmically to get a better sense for the distribution.



As we can see, the blue colors are the message loss rates for a system with two nodes, and the orange is for a system with 4 nodes. The values are as expected because the lower the message loss rate, the longer it takes for enough packets to be dropped consecutively so that a node is considered a failure. We also see that adding more nodes makes it more difficult to run into false positives because there's more heartbeating going on to prevent a node from being mistakenly marked as a failure.

A Closer look at some of the values in seconds of approximately how long it takes for a false positive failure to occur:

	2 Machines			4 Machines		
	30%	10%	3%	30%	10%	3%
Mean	6.5752231	27.740016	266.75466	5.529556	51.16592	1292.659
StdDev	3.9901524	31.191309	178.06661	4.435865	31.24239	383.392
95% Confidence	2.2575997	27.339885	156.0794	2.509781	27.38466	336.0518

Notice that more time spent can lead to uncertain results, but overall having more nodes and a lower message loss rate will make false positives extremely unlikely.