Qixin Zhu(qzhu3) Dongming Lei(dlei5)

Design

We treat the machines to form a virtual ring, each connected to its neighbors, specifically two successors and two predecessors. This design scales to large N because for each machine, it connects to its 4 neighbors no matter the value of N. To Marshal message, we use python to serialize the object to string, then use base64 to encode before sending through the udp protocol. Once received the message via udp, we again use base64 to decode the message and then use python's eval to deserialize the string to message.

Each node has one thread running a udp server listening for messages in this MP including heartbeats, leaving notification, joining request and so on, one thread running a tcp based RPC protocol from MP1 just for grep service, one thread periodically send heartbeats to neighbors, and finally one thread waiting for user command.

We use 01 node as introducer. When a new node join, it will send a message to the introducer ask for join. Once 01 received the join request, it will add this node to its member list and, subsequently, the 01 node notifies all member in its member list of this join, in the meanwhile, the heartbeating service running on node 01 is also able to pick this up and send it to its 4 neighbors, and the 4 neighbors will propagate the information (in their member list) to their neighbors. Each neighbor, once received the message via udp, iterates through the member list from remote node and current node and merge the two. Each machine's member list consists of three component, one for machine_id, one for heart_beat_count, and one for local timestamp. In the merge process, if the remote member list's heart_beat_count is great than local's, we update that entry with remote heat_beat_count and current local timestamp. On the other hand, if we see one entry's last updated time was 2s ago, we mark it dead. Eventually, all machines online are able to get this update. When a node voluntarily leaves the group, it will send request_to_leave message to other nodes in the member list and stop sending its heartbeats to its neighbors. Similarly, when a node fail, it no longer sends heartbeats to its neighbors. Once its neighbors noticed that it no longer receive update from this node's heartbeat count, its neighbors will mark this node dead, removed from the member list and push it into a recently_removed list to prevent the odds that we have this machine rejoined as new. Once this node stayed 12s in the recent removed list, we remove it from the list to enable this node to rejoin.

Usefulness of MP1

MP1 is very useful for us. It provides us a great convenience to get local log messages without having to log into the machine and type in multiple commands to retrieve and grep the log.

Measurement

All measurements below are shown for the entire system. In 1 second, 1 message sent by a machine and received by another one is count as 1 mps (message per second).
Background bandwidth for 4 machines

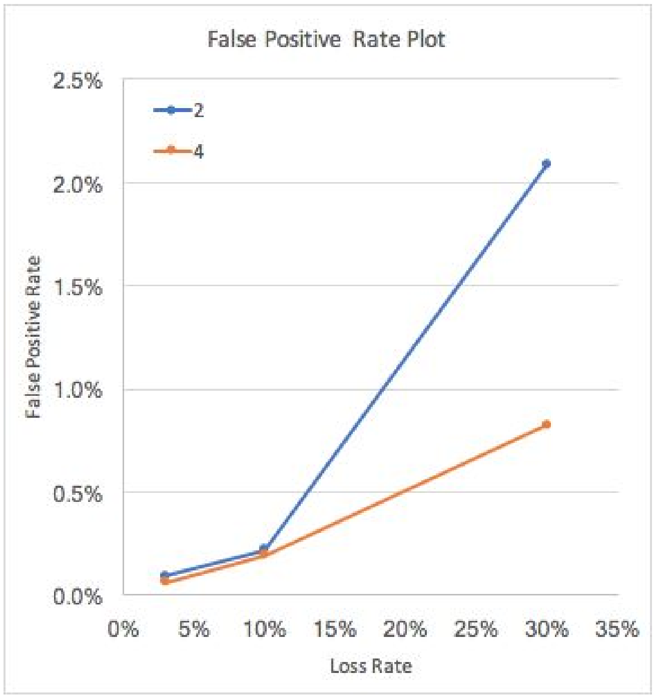
Each machine send 1 message to its 4 neighbors per second while each machine also receive 4 messages from its 4 neighbors per second. Therefore, the bandwidth for one machine is 4 and for 4 machines, it's 16 messages per second.

Expected bandwidth usage when a node (Assume 10 machines in the system)
Background: 40 mps
Joins: 1 message to introducer in addition to background bandwidth, total 41 mps
Leaves: 1 message to each member in member list in addition to background bandwidth. The leaver no longer send heartbeats. Total is 40 - 4 + 10 = 46 mps
Fails: the failed machine stop sending heartbeats, total is 40 - 4 = 36 mps

	Loss_rate=0.03			loss_rate=0.1			loss_rate=0.3		
	Avg	St.Dev	Conf.Int	Avg	St.Dev	Conf.Int	Avg	St.Dev	Conf.Int
N=2	0.0920%	0.0495%	0.0434%	0.2163%	0.2010%	0.1762%	2.0876%	0.4673%	0.4096%
N=4	0.0609%	0.0277%	0.0243%	0.1899%	0.0745%	0.0653%	0.8225%	0.5863%	0.5139%

Plot discussion:

(1) N=2 has higher false positive rate then N=4 case. The potential reason is that when a package is loss in a 4-machine system, other neighbors still receive the heartbeat from the same machine, and hence propagate the information "it's still alive" to other machines, resulting in less false positive detection. In a 2-machine system, there is no such "backup plan", so it will have higher false positive rate.



(2) False positive rate increases when loss rate gets higher. This is as expected. When more packages are lost, the probability of a machine missing other's heartbeat is higher, thus marking it as "offline".

(3) When the package loss rate is small (3% and 10%), the false positive rate is very small and don't affect much in real use. When the loss rate is significantly high (30%), the false positive rate starts increasing rapidly (for both 2-machine and 4-machine system). This is as expected. We conjecture that as long as loss rate is within acceptable range, the system is robust.