MP2 Report from Haorong Sun (Haorong4) and Haoyu Zhang (haoyuz3)

Introducer:

the introducer holds a hash table which stores all the members inside the group, we are calculating hash values by IP addresses, so the position for every node in virtual ring is relative fixed.

Member:

Normal member will store in a total of four neighbors’ IP addresses, two successors and two predecessors. The first three in the list is its members (receiving heartbeat from them), and the last three is its listeners (sending heartbeat to them).

Eg:

If [1,2,4,5] is the neighbor list store by node 3.

Then node 3 is receiving heartbeats from node 1,2,4

And node 3 is sending heartbeats to node 2,4,5

Since member will have at most 4 IP addresses storing locally, it has nice scalability for large number of machines.

The member will also keep a Fail list which records failures and the failure time stamp. Once a failure is detected it will be spreading through the virtual ring, so every node will be aware for all the failures.

Join: when a node wants to join the group, it will send request to the introducer. Introducer will send back its neighbors’ IP addresses, and this node will try to communicate with these neighbors and join the virtual ring.

Leave: when voluntary quit happens, the leaving node will first communicate with its neighbors, establishing new membership without this node. It will also inform the introducer that it is leaving, and then quit the group.

Fail: if the listener has not receiving heartbeat from its members for three times than heartbeat period, that member will be marked as fail. If a node fails, we designed that its successor and predecessor will exchange information and fix the virtual ring.

Our message form is “request \n info \n info \n info \n info”

Membership update:

“FullUpdate \n IP1 \n IP2 \n IP3 \n IP4”

In order to save message size, we only send IP addresses which is needed by the destination machine, averaging 1 or 2 IP addresses per message. Averaging around 40 bytes per message

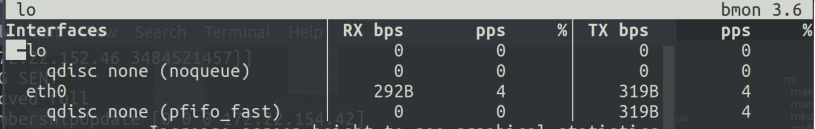
Failure update:

“Fail” \n “IP” \n “time stamp” Averaging around 20 bytes per message

Normal Heartbeat is “1”, so it is 1 byte per message.

While debugging the system, MP1 is kind of helpful, since it could help us to gather all the information from 10 vms to one terminal, saving lots of time and effert.

We used bmon to measure bandwidth usage. First measurement: the background bandwidth usage for one machine when six machines are connected is: 280 Bps on average for receiving and 300 Bps one average for sending. We used bmon to measure it.



Second measurement:

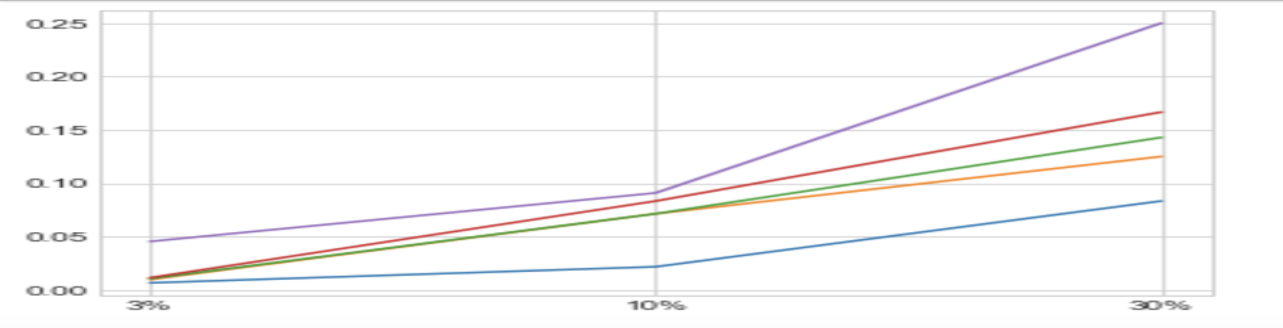
When a node joins, the average bandwidth usage for each of the new node’s neighbors is: 400Bps for receiving and 700Bps for sending. The bandwidth usage for the nodes who are not connected to the new node is almost the same as the first measurement.

Leave: Average bandwidth for each neighbor is: 350Bps for receiving and 450Bps for sending. Average bandwidth for each non-neighbor is almost the same as the first measurement.

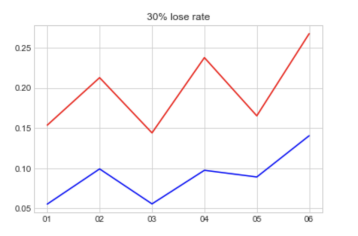
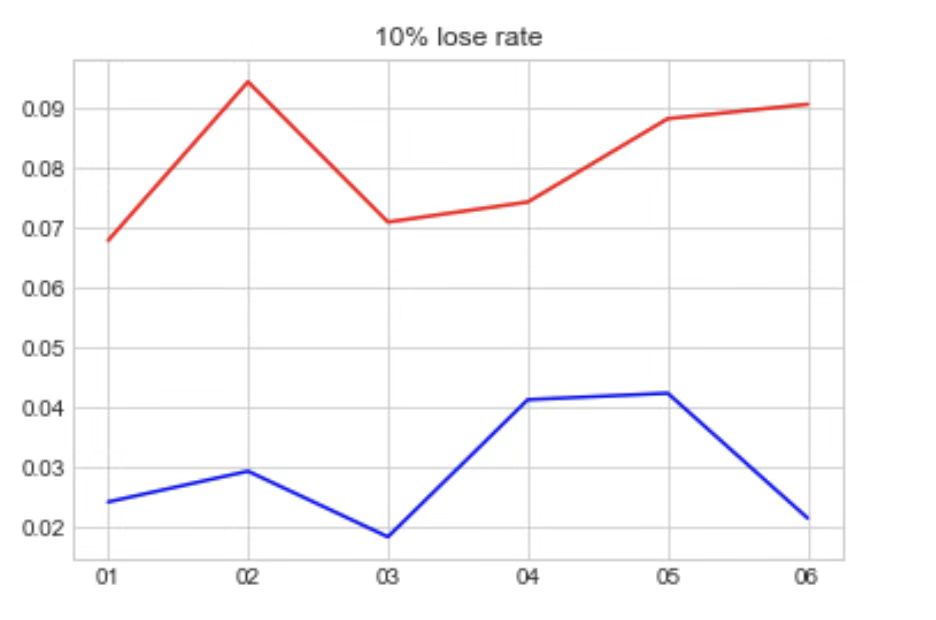
Fail: Average bandwidth for each node is: 300Bps for receiving and 400Bps for sending.

Third measurement:

We use Netem to set packet drop rate.



The above graph is false-positive-rate vs drop-rate. Each line represents a independent test.



In the above three graphs, each graph represent mean value and standard deviation of false-positive-rate on six machines with drop rate 3%, 10% and 30%. The red lines represent mean values and the blue lines represent standard deviations.

The false positive rate increases as the packet drop rate increases, just as we expected, because when the probability of packet loss increases, the heartbeats are more likely to be dropped, which makes a listener more likely to timeout and treat a member as failure.

The average values on different machines with same drop rate are nearly the same, because when the drop rates are the same, and number of trials is large, there should not be big differences among them.