Report

Hang Yu and Feng Lu

I. Introduction

THIS project we finished the required assignments of two phases,one is SWIM protocol, another is communication through NATS.

II. Implementation and Tasks

A. Membership Discovery

Membership discovery, based on the implementation of Ping-Pong message exchanging, in this part we first create a local list for every node, which is used for storing the new neighbors. Then give ping and pong the ability of piggybacking message, in this way when a node knows the new one is coming this message will disseminated to others through pingpong rapidly, and updates their own member list. In order to avoiding this method such isolated, we merged membership discover into the Task4, which is an improvement for decreasing the messages load on internet.

B. Failure Detector

Failure detector is important for the system, in this level, we access to the Timeout control with pong message, which means after ping message sent out, if no any response back the target node will be treated as dead node, then put this declaration into both ping and pong messages disseminated to group members.

C. SWIM FD Contd

The sometimes network communication is unpredictable, considering the in reality we can not sentence a node to death just because no response, this is so cruel, every nodes should have mercy. In this task3, we create a suspected lists for temporarily store the suspected nodes, then add the suspect timeout control.After pong timeout we initiate the suspect timeout,and gossip to others that I am suspecting this node, during this time if get any message from target, we will cancel the suspected status, or else declare its death.All the update message should be disseminated.

D. Message Ordering

Considering that one node may be declared alive or suspected many times, how to handle these large number of messages has become a problem. In this case, we import the incarnation number which is similar to time stamp. The work flow is shown below[1]:

- $\{Alive\ M_l, inc = i\}$ overrides
  - $\{Suspect\ M_l, inc = j\}, i > j$
  - $\{Alive\ M_l, inc = j\}, i > j$
- $\{Suspect\ M_l, inc = i\}$ overrides
  - $\{Suspect\ M_l, inc = j\}, i > j$
  - $\{Alive\ M_l, inc = j\}, i \geq j$
- $\{Confirm\ M_l, inc = i\}$ overrides
  - $\{Alive\ M_l, inc = j\}$, any j
  - $\{Suspect\ M_l, inc = j\}$, any j

In order to let every node keeps others incarnation number, we make every node inject its own current incarnation number into ping and pong messages before sending. So this local incarnation number track list(incTrack) replaces the new node member update list. Finally for a good compatibility with message ordering, we change the data structure into Map<NatedAddress, Integer>.

E. K-Indirect Pings

In this task, after pong timeout we chose random K members which are all correct and open nodes.At first these K members ping this suspected nodes as normal, when get the response from target node, K members will notice this information to requesters. Then requester will execute task4.

III. Analysis

A. Analyze Failure Detect

In this part, we test the system with varying the amount of failures and different network size(50,100,200).

In this scenario, we first initiate 50 nodes and the ID starts from 6.After 1000ms only number 10 is killed. From Fig.1, it can be seen that 50 nodes are initiated at beginning, after 120 millisecond the dead node is detected by all the 49 alive nodes in the system.

In 100 nodes size test, we randomly kill node ID 20,10,44 and disconnect 16 later, after 2000ms of start deadlinks between 10-13 and 10-12. In order to see whether the convergence of system easily, we pick all the node ID20s log data from aggregater component to draw this figure2. In figure2, we can see that in a 100 nodes system when 4 nodes are randomly killed, it will take at least 240 milliseconds on average to

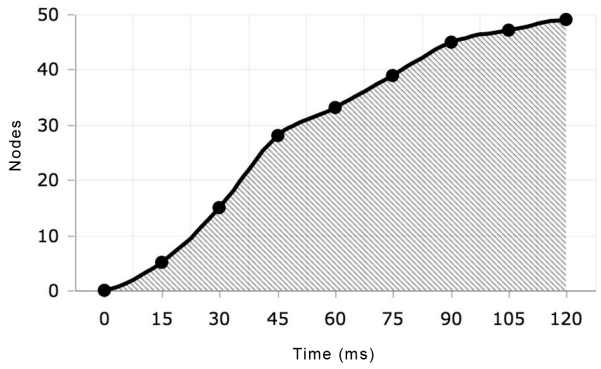

Fig. 1. Initiate 50 nodes in Failure detector system.

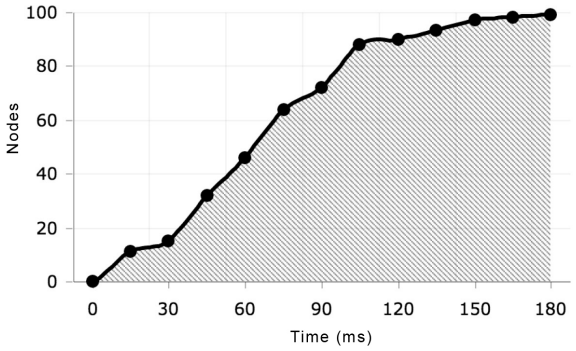

Fig. 2. Initiate 100 nodes in Failure detector system

converge, which means 4 fail nodes is detected 96 correct nodes.

In the third test, we first initiate 200 nodes, and kill only a node. From figure3, it can be seen that at least it takes 420 milliseconds to converge after initiating 200 nodes and kill a random node.Comparing with figure1, we can see with the size of network increasing, it takes longer time to converge.

B. Limiting the information exchanged

In this part the, in order to figure out the performance with message limitation, we limit the size when ping and pong piggybacking message(1,2,3). In the code, we randomly pick n(1,2,3) numbers of messages from the local list then set

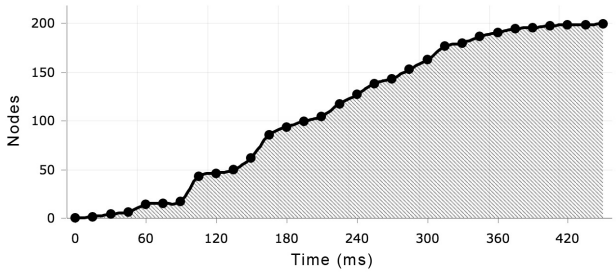

Fig. 3. Initiate 200 nodes in Failure detector system

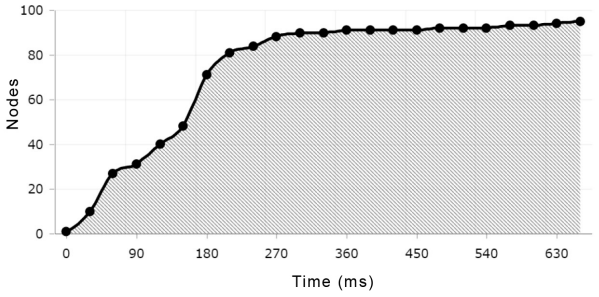

Fig. 4. The size of piggyback message is 1

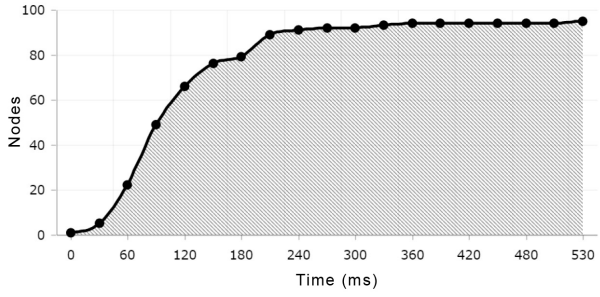

Fig. 5. The size of piggyback message is 2

them into ping and pong message.We initiate 100 nodes in the system, kill 5 random open nodes after 1000ms to see how long it takes converge. In every test, we only pick the log data of one node from aggregater component.(Dead node will be dected by 95 nodes)

From figure4, it can be seen that before 180ms the dead node is detected very fast by other nodes.Due to only piggyback a random message every time, the system takes long time to get converge.

From figure5, we can see that the converge time is 530ms, it is faster than Figure1. In this test, we update the limit to size 2.

Figure6 shows that system takes 360ms to converge. In this test the piggyback message size is 3.

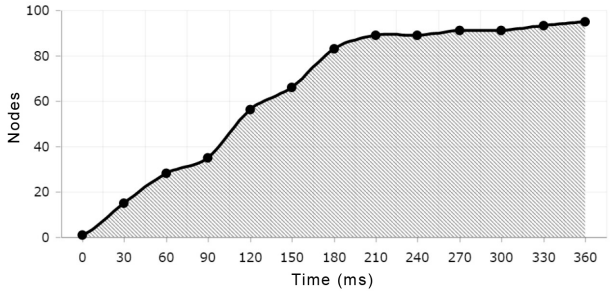

Fig. 6. The size of piggyback message is 3

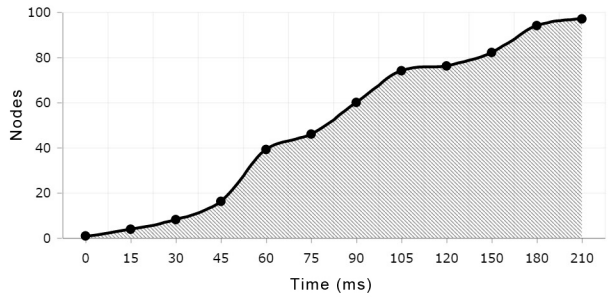

Fig. 7. Ratio of nated nodes to open nodes is 4:1

C. Nated vs Open

In the previous test, the ratio of nated node and open node is 1. Now we are going to change this ratio, first let us figure out the result of when the nated node is more than open nodes.we make the ratio of nated to open is 4:1, and kill only one node after 1000ms. From figure7 we can see the system is becoming stable at 97 nodes have detected the dead node, but not all,the convergence time is 210ms.When we change this ratio to 5:1, we find that only 95 nodes can detected the fail nodes.With the ratio increasing we find out that system becomes unstable, when the ratio greater than 6:1,no nodes can be detected.

IV. SWIM Through the NATs

We first start and connect the Croupier component in the NatTraversal component, then use the heartbeat mechanism to detect the dead nodes among the parents of a node, which is the same way with SWIM component. We monitor every Nated node parents, if parents size is less than 2, we will add the new public nodes which come from croupier component port to the parent list.Meanwhile, in order to let Swim component to know the publicSample from the croupier, we use NatTraversal port to forward these updated parents to the SWIM component, then inject them into ping messages.

V. Improvement

During implementation we found that there is a situation should be considered,which is when we receive ping messages from a suspected node, we should declare it is alive immediately.So we add this mechanism into the NetPing event.

References

[1] Abhinandan Das, An Das, Indranil Gupta, and Ashish Motivala. Swim: Scalable weakly-consistent infection-style process group membership protocol. In In Proc. 2002 Intnl. Conf. Dependable Systems and Networks (DSN 02, pages 303312, 2002).