CS425 MP2 Group 44 Failure detector

Shichu Zhu(szhu28), Fan Shi(fanshi2)

Overview and Protocol Design:

We build our Failure detector based on the insight of SWIM failure detector and Google's protocol buffer. The detector is embedded into the structure of our distributed system.

We implement the protocol on an extended ring. Each process (including introducer) has its own location corresponding to his "IP:port" identification number. Each process will select three successor sending Ping message and will mark a target failed once it doesn't send 'Ack' message within timeout. We use Gossip method to disseminate the failure information. When new node join the network, the introducer will applying the same Gossip method to disseminate the joining information.

The protocol buffer is well designed with clearly structured. Each UDP message will include a message type so that we can handle all type of incoming information at one place. For Failure detection message, it will include a header to specify the functionality of this message. Therefore, we can encode Ping, Ack, Delete and Join information in the same protocol message type.

The failure detector module is split into sender and receiver module. Sender will be responsible for sending Ping message and marking failure member. Receiver will deal with all kind of incoming UDP message, and execute corresponding action.

Details of Protocol implementation

[1] Join:

1.1 The new join will call introducer for full membership list

1.2 The introducer then disseminate the joining information

[2] Ping-ack:

2.1 Each process sends ping message to three successors and collects response.

2.2 Receiver get Ping message and return Ack message. (For each call)

2.3 Set timeout of 1.5 second. Mark failure for process with Not Ack message.

2.4 Disseminate the failure message if any failure (both real and fake) detected.

[3] Volunteer Leave:

3.1 Mark failure for the process itself

3.2 Disseminate the failure message.

[4] Ping Targets chosen:

4.1 Pick three successor from membership list and send ping.

4.2 Select three random member for gossip style dissemination.

Scalability and UDP handle:

The protocol can be easily scaled. Each non-introducer member will contact reasonable amount of ping targets. If we want to add more members to the system, we simplly add member to each process' member list through gossip style dissemination. Even if with large number of nodes in the system, the background bandwidth is scoped into a considerable range. Each member will be pinging and pinged by three other members. Failure message will be sent in Gossip style so that there is no single node that has to send lots of failure detected message.

UDP connection can be unstable so that we send each message for multiple time to ensure that at least one message can arrive at target.

MP1 Usage in MP2

MP1 is very useful for this MP. Firstly, we use the same protocol buffer as in the MP1. Also, we can generate helpful log information to debug the system. MP1 's log generator can help us find the corner cases and the failure. In the future MPs, we will incorporate MP2 failure detector into the first MP so that the gRPC framework in MP1 will have a sense of dynamic topology. Due to the fact that TCP is banned for MP2, we did not merge the failure detect within the gRPC framework for now.

Measurement:

[1] Background bandwidth for 4 machine

133.47 bytes/s (Average for four machine). The bandwidth measurement was achieved by wrapping the UDP sender with a byte length counter. Each second the byte counter is reset and at the end of the simulation the average bandwidth of the specific node was computed by averaging. Further average over all nodes was done to compute the presented results.

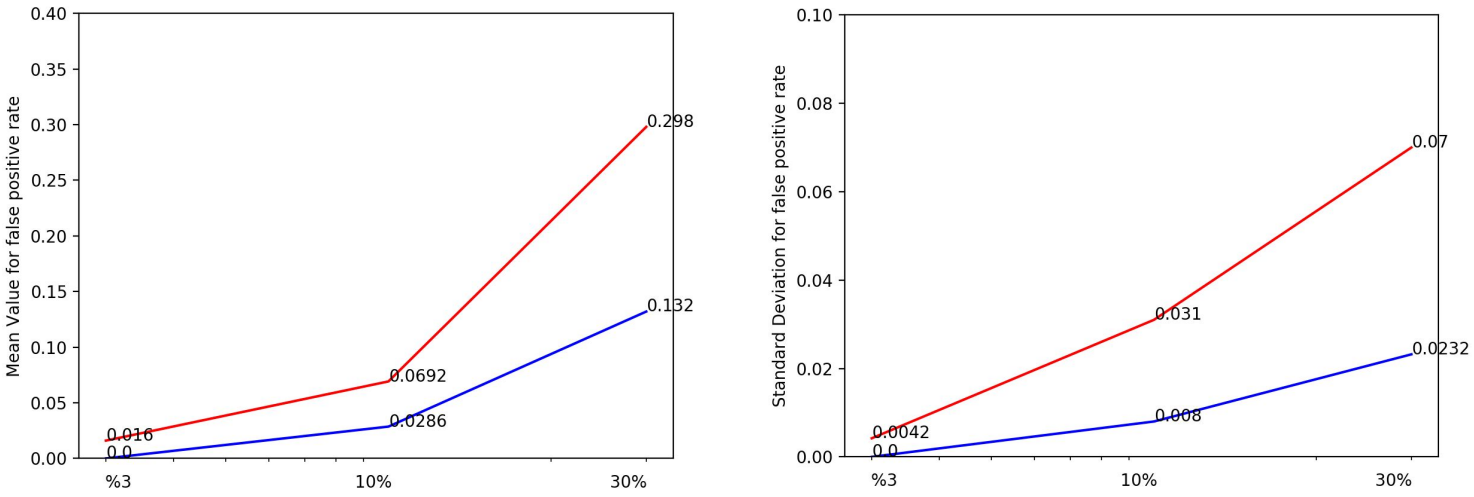
[2] Expected bandwidth for join, leave, fail

For this measurement, we measured the average bandwidth for a single node join. Bandwidth for leave and join include background ping-ack and failure gossip message. The bandwidth for join includes transmitting full membership list for new node.

	Join	Leave	fail
Bandwidth count	475 bytes/s	341 bytes/s	325 bytes/s

[3] False positive rate at different status

We measure the false positive rate for 2, 4 machine at drop rate %3, %10, 30%. Each experiment has been conducted for 5 times. The mean and standard deviation is calculated and plotted below. (3% failure rate for two machine is too small to measure, we mark it zero here).



Analysis for the false positive rate:

Our measure method is that we open the numbers of machine at once, wait until the first false positive detection. In this way, we will mark false positive rate as (1 / timeToFirstFailure). The meaning the rate is “how many false detections can happen within one second of the system running”.

We can see the trend of false positive rate is increasing as the chance of packet loss is higher. This is obvious as the connection is less reliable. The other trend is that false positive rate also increases as the size of the system is growing in size. However, this does not imply that the SWIM algorithm is not scalable, due to the following two reasons: 1. The rate measures any node fail in the entire system instead of normalized by number of nodes. 2. We did not implement the suspicion mechanism in SWIM (i.e. the indirect ping).

The standard deviation is plotted in the right panel. It’s generally proportional the the average value. For confidence interval, we simply use the double standard deviation [u-2s, u+2s] as this provides the 96% confidence of sample falling into the interval. The standard deviation is an order of magnitude smaller than the mean, which implies that our implement is stable and robust.