Machine Programming 2 – Distributed Group Membership

Yunsheng Wei (wei29) Neha Chaube (nchaub2)

In MP2, we developed a distributed group membership service based on Gossip Membership Protocol. Some design decisions are as follows:

1. Each member maintains a local membership list, the membership list is implemented as a sorted list, so as to support linear time merge, update and random choice. The membership list contains one item for each member in the group, each item contains 4 fields [id, heartbeat counter, last update time, state].

2. There is an introducer, which is a machine whose IP address is known to every one. Whenever a new member wants to join the group, it will send its membership list (containing itself as the only member) to the introducer, and the introducer will know its existence, and gossip to other members in the group.

3. Whenever a member wants to voluntarily leave the group, it will gossip a membership list containing itself with state marked as “LEAVE”. Once others know its leave, they will mark its state as “LEAVE”, and continue to gossip its leave to others, until it is deleted from their membership lists.

4. Once one member hasn’t been heard for T_Fail time, its state will be marked as “FAIL” by others and stop being gossiped. But an additional time T_Cleanup is used to decide when to delete it from others’ membership lists, which can prevent failed members from appearing as new ones.

5. Every member periodically sends their membership list to introducer at a relatively low frequency, which enables introducer to function normally after restoring from crash, and can also prevent partitioning.

6. There is only one kind of message in this design -- membership list, so the protocol is quite simple and uniform. All membership lists are all sent over UDP, and marshalled using JAVA’s serialization mechanism.

7. We set the frequency to gossip membership list to once every 300 milliseconds, and the frequency to send membership list to introducer to once every 2000 milliseconds so the bandwidth usage is about 4 messages per second, which is not relevant to number of machines N, so it scales to large N. Other key parameters are T_Fail: 1500 milliseconds, T_Cleanup: 4000 milliseconds.

Experiments

Bandwidth Usage

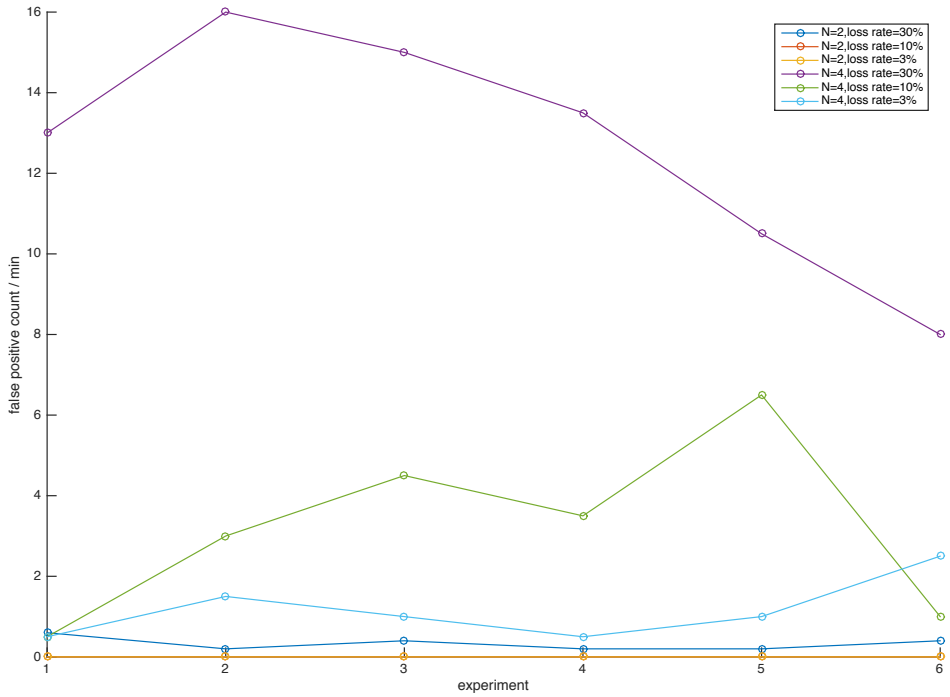
Measurements of background bandwidth usage for 4 machines:
We use *iftop* to measure the background bandwidth usage.

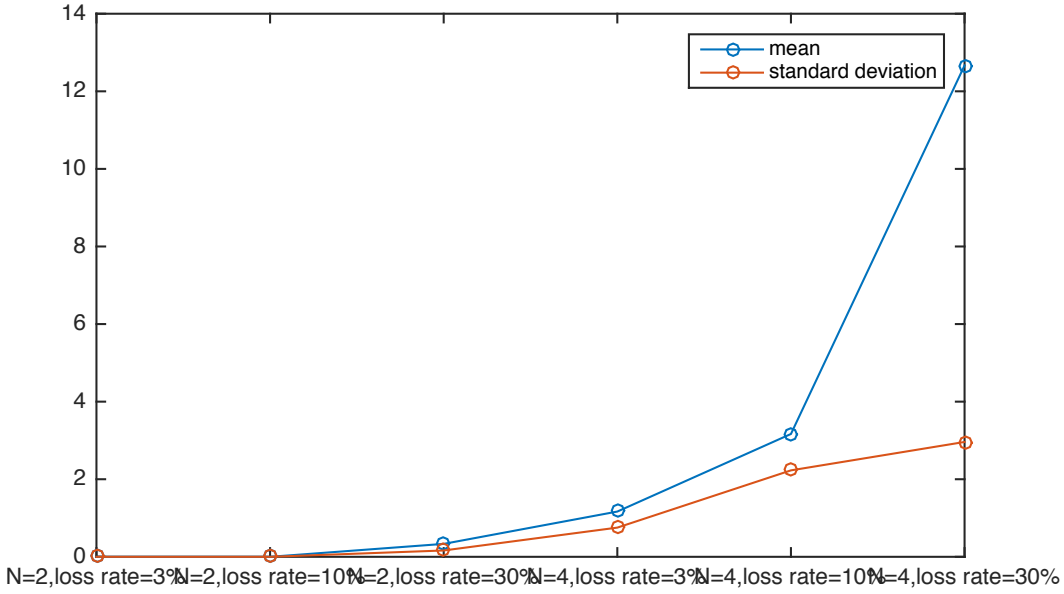
The transmitted traffic is around 22.5Kb / s.
The received traffic is around 22.5Kb /s for ordinary members, and 31.2Kb / s for introducer.

Expected bandwidth usage whenever a node joins, leaves or fails:
Since our design is based on gossip, so there is no difference between joins, leaves and regular messages, so the bandwidth usage doesn't change much. What increases is just one item in membership list, and it increases about 2Kb/s. For join and leave, it takes O(logN) (N is number of machines) gossip periods to propagate one member's joining and leaving to every other member with high probability.
In the gossip membership protocol, failure is based on timeout, so no special messages are gossiped about failure, so no extra bandwidth is required.

False positive rate

We use *iptables* to simulate packet loss. The following plots are our experiment results.





The first plot is the false positive counts per minute for every experiment we conducted. We did 6 experiments for each setting. In our experiment, for N=2, and loss rate = 3%, 10%, we did not observe any false positive failures, because theoretically, the probability of false positive failure will be 0.03^5 and 0.1^5, it is too low for us to observe in a reasonably short time.
The second plot is the average and standard deviation for each setting. We did not plot confidence interval, because we only have 6 data points for each setting, thus confidence interval is not easy to compute, and can just be approximated by ± 2 standard deviations.

From the plot we can see with N and packet loss rate increasing, average false positive counts are increasing, as well as standard deviations. The result is the same as expected, because with N increasing, the expected time to gossip a member's heartbeat to all other members increases, and the randomness of gossip also increases, so this explains the results well.

In MP2, we use MP1's distributed log querier to query the logs for each local membership list, and get the results for our experiments.