# CS425, Distributed Systems: Fall 2019 Machine Programming 2 – Distributed Group Membership

## Design

For this MP, we designed a failure detection system with a virtual ring.

**Introduce and Join**

When the first node joins the ring, it will scan all introducers in a hard-coded introducer list which contains IP address and port for all possible machines in the system. It waits for one second and if it did not receive an ACK from any of the machine, it will join the ring with a membership list which only contains itself. For the rest of nodes, they will scan all machines and join the first introducer that replies. On the introducer side, when it receives a join message, it will send its membership list as well as a new id to the new node, and broadcast the new id, IP, port and join time to all other nodes in the ring.

**Heartbeat and Monitor**

In our ring, each node will send heartbeat to three successor every 1.5 seconds and monitors heartbeat from its three ancestors every 4 seconds. If the ring contains less than three nodes, it will monitor all nodes except itself. Whenever a node receives message for add new member, it will check if this node is inserted into three of its ancestors, if so, it will start monitor it. At this time, the fourth ancestor will stop sending heartbeat to it, and it will realize this is not a failure. Same applies to send heartbeat, if the monitor becomes its fourth successor, it will stop sending heartbeat. And if new node inserted into its successor, it will start send heartbeat to it. When a monitor did not receive heartbeat for 4 second, it will mark that id as failure. Then broadcast failed node id to all nodes in the ring. Then delete it from its own membership list.

**Timer**

We start a timer with call back function (NodeTimeOut) for each node. Every time we receive a heartbeat from a node, we reset the corresponding timer so that we can wait another full timeout period.

**Leave**

A node leaves when it receives a Linux signal (INT, kill -2 <PID>) sent by user. It will go into a signal handler and notify all nodes in the ring to delete it before it exits the program.

**Query Logger**

We use query logger to query logs on all machines. It helped us a lot when debugging because we can check all the logs and grep the keywords easily. However, its usefulness is also depending on our ability to write informative logs in our program. We found many critical logs in our code which might be problematic and logging it helped us a lot when debugging

**Marshaled Message Format**

Our marshaled message is a struct called Packet, it contains these fields:

Action ActionType; Id int; IP string; Port string; Map \*MemberList

ActionType includes ACTION\_JOIN, ACTION\_REPLY\_JOIN, ACTION\_NEW\_NODE, ACTION\_DELETE\_NODE, ACTION\_HEARTBEAT, ACTION\_PING, ACTION\_ACK.

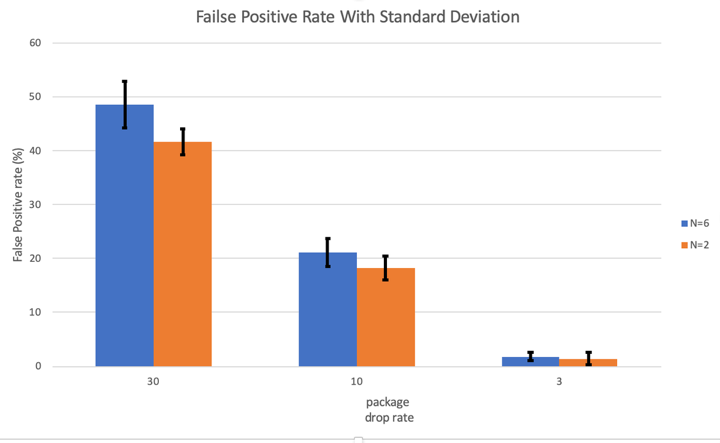
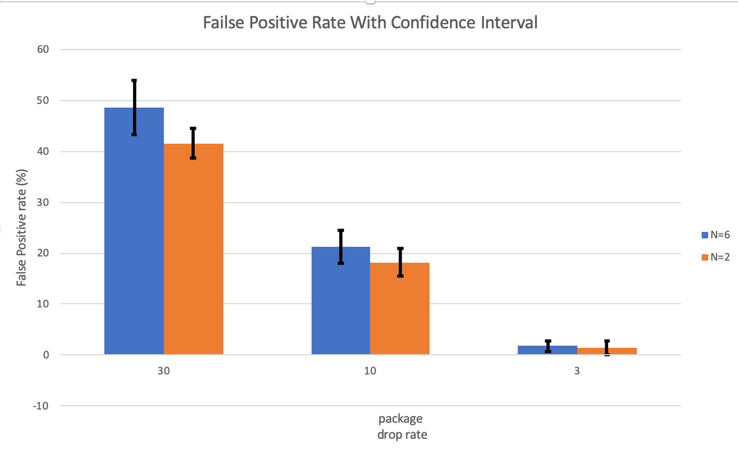
The most expensive packet includes a map, which contains membership list information sends from introducer to new node.

Our design is scalable to large N because for frequent communication, heartbeat, we only send to next three successor, and for broadcasting, we only do so when a new node joins or leave. This should not be very frequent. So, the average background bank width used will not be too high in normal situation.

**Bandwidth**

1. We used iptraf-ng to monitor udp traffic from our computer. It shows that every 1.5 second, a node sends three 77 bytes packet to other nodes, so the heartbeat bandwidth for one heartbeat sent is 77 bytes. Thus, the background bandwidth is 3 heartbeats \* 77 bytes \* 10 nodes / 1.5 second = 1540 B/s.
2. Measurement shows the average buf size for join is 59 Bytes. And reply membership list to a node takes 187 + 103\*N Bytes, So, totally 59 \* N nodes + 187 + 103 \* N = 162N + 187 Bytes (plus UDP overhead). N is the number of items in membership list.
3. The average packet size for leave and failure is the same. Delete node packet cost a payload 55 bytes from measurement of buffer size. Broadcasting the packet to k nodes takes 55\*k Bytes (plus UDP packet overhead).

Plot & Analysis

****

We set up the test by using 10 heartbeats for each test and check how many false positive happened. The decrease of package drop rate showed significant affect with the false positive rate. This is because false positive rate is related to package drop rate, when package dropped in accident for several times, the monitor did not receive message from that node for long enough time or 4 second in our case, it will treat that node as a failure node and broadcast this update to all other nodes. When we increase the number of nodes in the group to 6. All nodes send heartbeat to not just one but three other nodes, so either one of them did not receive heartbeat for long enough time will broadcast to everyone that the node failed. Thus N=6 have a slightly higher false positive rate, which is in our expectation. The standard deviation and confidence interval are very similar data both showing the distribution of our measurement we can see that it doesn’t vary a lot.