Design:

We create a udp listener thread for each server. Also a thread for sending ping and ack is created, and a 'cmdline thread' is created for monitering inputs from keyboard. We maintain a global member list variant in each server.py file. And we have different functions for different commands received from keyboards. For example, 'lsm' means to list the member list of this server, 'join means to join this server into group. 'leave' means this sever leaves the group. These commands will trigger specific functions to send socket message. The format of message is [[<command>, <sender_ip>], receiver_ip]. The commands keywords are like 'join, leave, ping, ack'. When udp listenser of the receiver machine hears, it will implement different functions like ask_for_join, handle_leave_requests, update_ping_ack_count and merge_member_list. We use VM1 as introducer, when it received the message asking for join, it will add the new server to its member list and notify its neighbors that a new member has joined. In the member list, every row is a triple tuple (id, ping_ack_count, timestamp). Id is the domain name of server, ping_ack_count maintains how many times this server respond to other servers' ping. When merging list, the brand new member would be added. The existing member's ping_ack_count would be compared with the one from receiving member list, maintaining the ack count always larger. In failure detecting function, if the timestamp of a member in member list has not been updated for a period, mark it as offline.

No matter how many N is, each server.py would run independently. And each server's neighbors would always be the previous two and succeeding two members in its member list. When member list change, such as a server leaves, the current server would turn to hear next server which is nearby on member list.

Usefulness of MP1

MP1 lets us conveniently get local log messages without logging into the machine and type in many commands to retrieve and grep the log.

Measurements

assuming the ping-ack period is P, each period would send 2 messages between two machines. then on average for a single machine, the unit bandwidth usage is 1/P.

(i) the background bandwidth usage (bandwidth is always in Bps) for 4 machines:

Each machine would send 4/p messages per seconds, therefore the background bandwidth usage for 4 machines is 16/P bps.

(ii) the average bandwidth usage whenever a node joins, leaves or fails :

Assuming there are 10 machines in the group, the background usage is 40/P bps.

For join, an additional message would send to introducer, and introducer will send its updated member list to other machines. So in total there would be (40 + 1 + 10)/P = 51/P bps.

For leave, 9 leave message would be sent. And the left machine will no longer send message. Therefore the bandwidth is (40 – 4 + 9)/P = 45/P bps

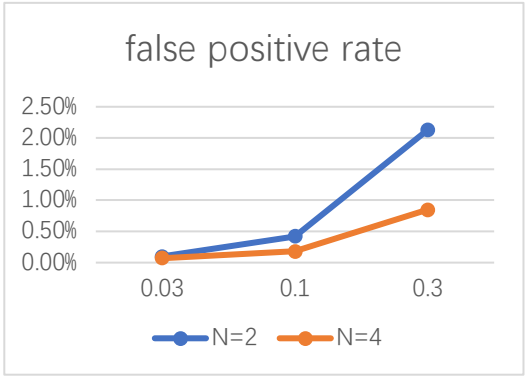
For fail, the failed server would not send message. Therefore the bandwidth is (40 – 4)/P = 36/P bps

(iii) Suppose the confidence is 0.95.

	Loss = 3%			Loss = 10%			Loss = 30%		
	avg	Std	Con_int	avg	std	Con_int	avg	std	Con_int
N=2	0. 10%	0. 04%	0. 05%, 0. 09%	0. 43%	0. 09%	0. 34%, 0. 47%	2. 13%	0. 46%	2. 00%, 2. 31%
N=4	0. 05%	0. 03%	0. 02%, 0. 08%	0. 19%	0. 08%	0. 13%, 0. 23%	0. 84%	0. 59%	0. 74%, 1. 28%

Plot discussion:

(1) N=2 has higher false positive rate then N=4 case. The possible reason is that when a package is lost in a 4-machine system, other neighbors still receive the ping-ack from the same machine, and hence the information “alive” will be propagated to other machines, resulting in less false positive detection. In a 2-machine system, when a package lost, the other one will never receive it. So it will have higher false positive rate.



(2) False positive rate increases when loss rate gets higher. This is as expected. When more packages are lost, the probability of a machine missing other’s ping-ack is higher, thus marked as “offline”.

(3) When the package loss rate is small (3% and 10%), the false positive rate is very small and don’t affect much in real-world system. When the loss rate is significantly high (30%), the false positive rate starts increasing rapidly for both 2-machines and 4-machines. This is as expected. Therefore as long as the loss rate is within acceptable range, the system is robust.