MP2 Report - Group 57 - Jiazheng Li (jl46) & Yabo Li (yaboli2)

System Design:

First of all, to identify each process, we use the combination of its IP address and the timestamp of joining the group as its unique ID. All the processes know the IP and port of the introducer of the group (for now hardcode). When initializing the group, each process sends its information to the introducer, and the introducer sends back its current membership list to the new member, and the new member can start pinging. We use a ring topology: each member only pings its 3 successors, (because the maximum number of simultaneous failures is 3, this would make our design complete, with the optimal networking resource consumption). We combine pinging and gossipping: the message in pinging may contain certain information about a node's status (with codes: 0 as failed, 1 as new join and 2 as leave), or just a very short ping containing very few bytes. We also use encoding/json to make our message platform-independent. All the processes marshal the messages into json bytes and unmarshal the json bytes back to their original type. Each process is holding a messageQueue, as a buffer to store the information it detects by itself, or receives from its predecessors. Each process periodically pings its successors with the first message in its messageQueue, if exists. The pinging period could be used as an adjustive parameter. We can ensure all failures could be detected within 1 second in our design. We also implemented a logger that could write important information about status update into log files, and the log files could be queried by our querier in MP1. When there is some output on the terminal out of our expectation, we use the querier to check the actually written information in the log file, and compare it with the terminal output. As for its scalability, under that assumption of 3 simultaneous failures, the message overhead for each node should still be O(1) on average when N changes, however the worst case would be proportional to N in terms of the overhead of transmission of group status update.

Background Bandwidth Usage for 4 VMs:

	VM01	VM02	VM03	VM04
Avg Receive	58 Bps	42 Bps	46 Bps	50 Bps

The way we measure our bandwidth is using *tcpdump* lib on Linux, to capture all the communication among VMs on the port "12345". In and out packages of one VM are calculated together. After adding up the communication bytes size, we divide this sum by the time period, to calculate the avg bps during this period. The bandwidth for the 4 VMs are under normal condition (no join, no leave and no fail; just normal pinging). We can witness very similar bandwidth for all the VMs.

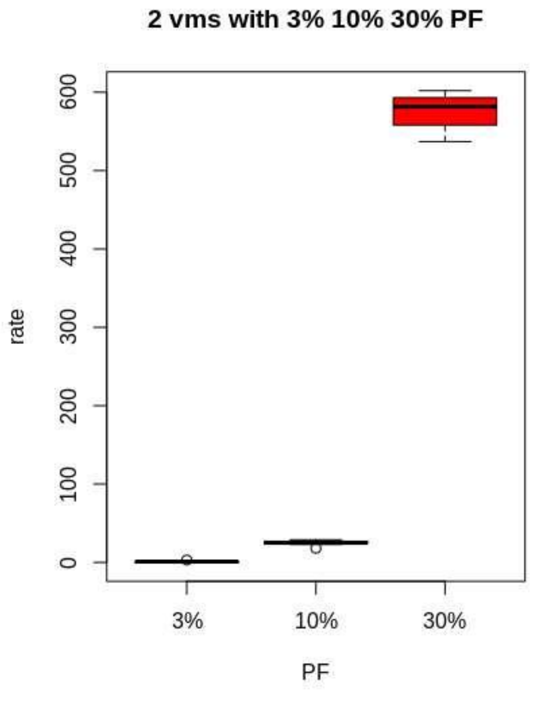
Background Bandwidth Usage for 4 VMs:

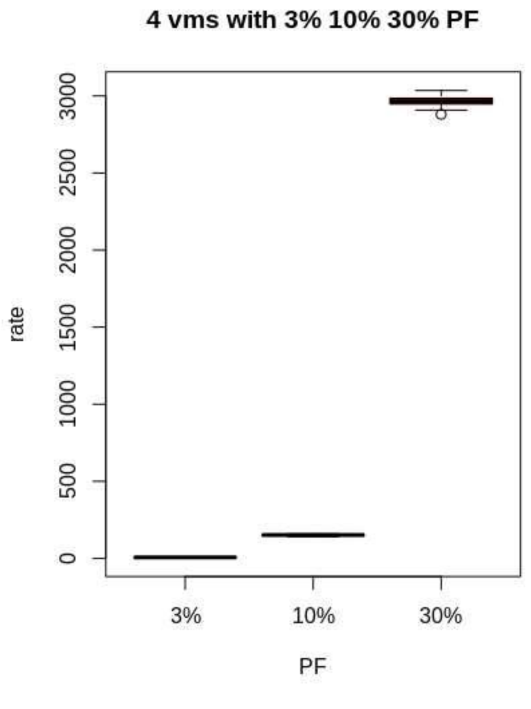
	Join	Leave	Fail
Introducer Avg Bandwidth	921 Bps	593 Bps	544 Bps

Non-Introducer Avg Bandwidth	593 Bps	606 Bps	588 Bps

When special events (join, leave or fail) happen, the bandwidth during the message transmission phrase would become much higher, the avg payload size in terms of bytes increase from 4 to about 40; the information contains the process to whom the event is happening, its unique ID, its status and the TTL value for message transmission. When a new node joins, the bandwidth of the introducer becomes very high because it needs to send it membership list to the new process (the larger the group size is, the larger this payload would be). Leave and fail show a similar situation for all VMs.

False Positive Rate:
The way we avoid false positive to improve our accuracy is when a pinged target doesn't give ack, it tries to ping it two more times. If none of the three consecutive pings receives ack (each read deadline is 30ms, and time difference between each attempt is 20ms; so in total if the sender never receives the ack from the recipient in 150ms, it marks it as failed). To simulate network package loss, we assign a probability *p* at the pinged target not to reply to the coming message. So the probability of all three pings receive no ack is p^3, and the false positive probability should be $1 - p^3$. Because we use no suspicion mechanism, we assume this number is the same for both 2 VMs and 4 VMs. We calculate the probability of there is at least one false positive in the group when each member finishes one ping.





The left figure shows the mean, standard deviations and confidence interval of 2 VMs, The right one shows the mean, standard deviations and confidence interval of 4 VMs. We can see because in 2 VMs there would have two message channels, but when there are 4 VMs there would be 12 channels, so the probability of at least getting one false positive of 4 VMs is 6 times of that of 2, which corresponds to our theory.