MP2 Report - Group#38 - yiteng3(Zhang) & maojunx2(Xu)

Project Design

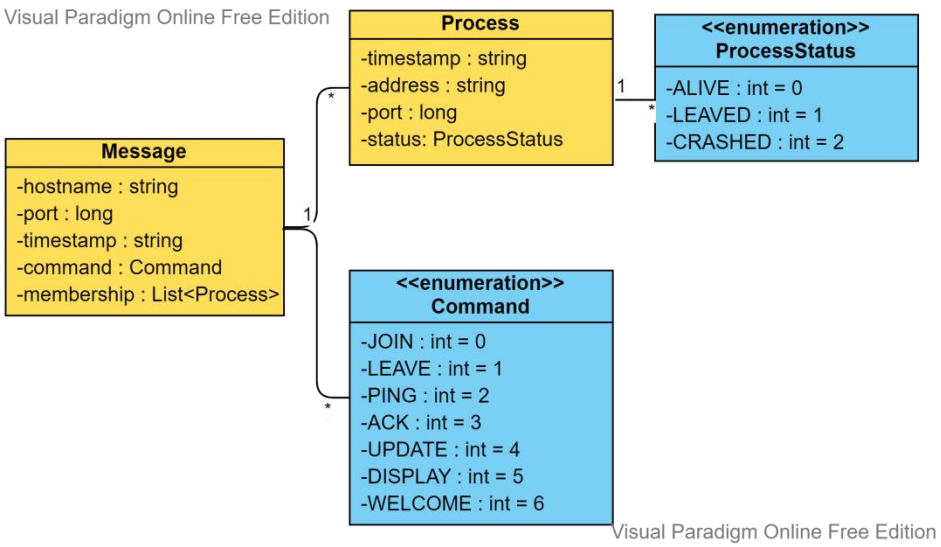
We build a ring topology connecting all nodes together. In addition to the introducer, each node can interact with up to four surrounding nodes(2 predecessors, 2 successors). In order to reduce the use of bandwidth, the message passing uses the UDP protocol.

Our program mainly includes five classes **Main**, **Sender**, **Receiver**, **Monitor** and **Updater**, and a formatted and serializable **Message**. **Main**, as the entry point of the program, can accept five instructions: "join", "leave", "list_mem", "list_self", "grep", which represent joining the distributed group, leaving the distributed group, and letting all members in the distributed group print their membership, printing its id , and query logs on all machines. **Sender** contains two static methods, which provide the interface for sending UDP datagrams of the entire program. The **Receiver** is responsible for monitoring the Socket port of the machine, parsing the received request, and performing corresponding processing according to the type of the request. **Monitor** detects if they are crashed by pinging up to 4 surrounding nodes(2 predecessors, 2 successors). After each round of ping, **Monitor** waits for 4.5 seconds, and then checks whether the **Receiver** receives the ACK responses from the corresponding nodes. If any node has not responded, **Monitor** will mark it as crashed. **Updater** is a utility, and almost all changes to membership on the machine are done with its static methods. It can complete the local update operation according to the incoming message from the Receiver. It is worth noting that the timestamp of any member that changes will be updated.

Scalability

The program maintains the topology through a linked list, which can hold a considerable number of nodes. This program can scale to large N, because each node only communicates with up to four surrounding nodes. Therefore, even if N is large, the bandwidth usage of each node is the same as the one when N=5.

Message Format



Completeness

Regarding these three completeness, **A. meets 5 second completeness:** the ping interval of our Monitor is 4 seconds, so if a machine crashes, at least one machine in the system can find this error within 5 seconds; **B. meets completeness up to 3 failures:** We can monitor a total of 4 neighbors(2 predecessors, 2 successors), so when any three processes fail, remaining processes can detect those failures, and the rest processes can converge to a new topological ring; **C. would violate completeness with at least one combination of 4 simultaneous failures:** If four processes crash at the same time, for example, machines 1, 2, 4, and 5 crash, then the rest processes cannot detect whether 3 is alive within 5s.

Use MP1 for debugging MP2

In MP2, membership updates are output to the logging.log file. We use MP1's grep function to query the updates on all machines. This extremely saved our debugging time.

Measurements

Background bandwidth usage (N = 6)

Our "Ping" type of Message does not carry membership, and the size is 42 bytes. When N=6, each node sends 4 "Pings" every 4.5s, so the average bandwidth is 6*4*42/4.5 = 224Bps.

Average Operations bandwidth usage (N = 6)

The size of each process is also 42 bytes, so the size of "Update" and “Welcome” are also (N+1)*42.

Join When a process Pi joins, it firstly sends a "Join" to introducer and introducer send back an "Welcome" to Pi which includes current membership list. Then Pi sends "Update" to its 4 neighbors and they will also send "Update" to their neighbors. So the bandwidth for "Join" is (6+1)*42*2 + 4*(6+1)*42 + 4*4*7*42 = 22*7*42 = 6.3 KBps.

Leave When a process Pi leaves, it firstly sends a "Leave" to its 4 neighbors, and they will also send a "Update" to their neighbors. So the bandwidth is 5.73 KBps.

Crash When a process crashed, its neighbors will detect the crash within 5s and send "Update" to their neighbors. So the bandwidth is 4.58KBps.

False positive rate

	N = 2		N = 6	
Message Loss Rate	3%	30%	3%	30%
Averages	1.976%	28.962%	2.877%	30.234%
Standard Deviations	0.0676	3.2300	0.0922	3.8240
Confidence Intervals	[1.846, 2.106]	[23.632, 35.292]	[2.696, 3.058]	[22.739, 37.729]

Discussion According to our test result, FP rate increases when UDP lost rate increases or the number of processes increases. For the former, it is reasonable that when lost rate increases, it can be more possible that a Ping doesn’t get ACK. Therefore FP rate increases. For the later, when N increases, the number of messages per second increase, is more likely to have 1 package lost, therefore FP rate increases.