ECE 428 MP2 Report

Shashank Saxena ssaxen4
Rahul Surti rsurti2

Algorithm Used
For detecting failures, we select K random nodes (in our case, 4 random nodes and the introducer node). We send our heartbeat to these nodes and our membership list along with it. Since the nodes we select are random each time, we can say that it scales to a large N since there is a high possibility that we select K-1 *new* nodes every time we select our nodes. The likelihood of this increases as we scale N.
Marshaled Message
We serialize our messages into a byte stream and send the byte stream across the network. Serializing and deserializing messages takes a bit of string parsing, but we are ensured at least the data sent over the network is marshaled.
Background Bandwidth
Background bandwidth usage for 4 machines can be calculated as follows:
Since our K value for selecting random nodes is 5, we must select all 3 other nodes to send heartbeats to. The max size of our UDP packet data is 1024 bytes, but we use less than 100 bytes per entry in the membership list. The period of heartbeating is 0.5 seconds. Total load on the network is 4 nodes * 3 nodes to heartbeat to * 80 bytes * 4 nodes in membership list / 0.5 seconds heartbeat time period is just under 8000 Bytes/sec.

We have no additional bandwidth usage for joining, leaving, and failing because of our selection algorithm for selecting nodes to send heartbeats to.
Joining:

1) When a new node joins the the network, it will send a heartbeat with a membership list of just itself and the introducer node to the introducer node.
2) Whenever an introducer node receives heartbeats, it keeps a priority queue of the most recent nodes to join the network.
3) The introducer node then sends its heartbeats to K most recent nodes, containing the most updated membership list, since every node must send a heartbeat to the introduce node by design.
    a) Our K value is 5 since the max number of nodes that can fail simultaneously is 4. This ensure the that at least one node will receive your heartbeat.
4) This ensure that within 2 successful heartbeats (1 second, assuming no packet loss) a new node will know of all the other nodes in the network. From here, the new node will choose K-1 random nodes and the introduce node to send its membership list (which includes the new node itself) to and spread the message of it joining the network in logarithmic time throughout the rest of the network.

The expected bandwidth usage for joining is just the bandwidth used to

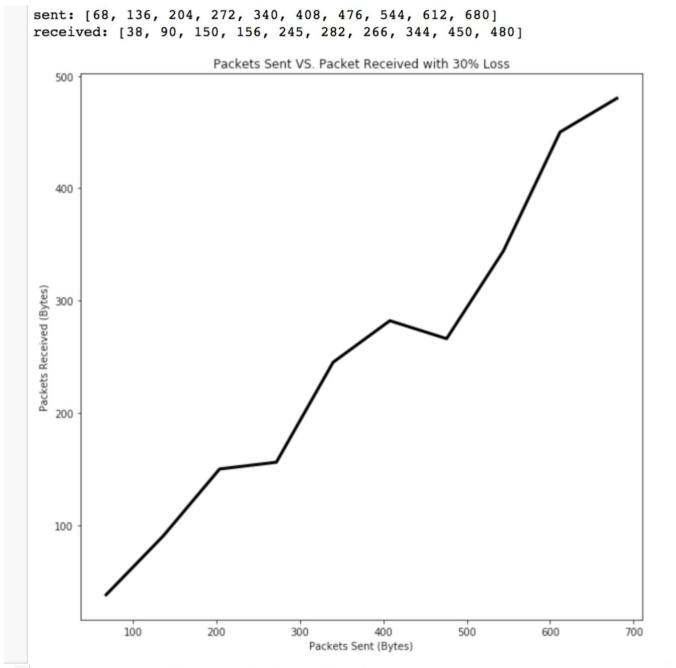
ECE 428 MP2 Report

Leaving/Failing:

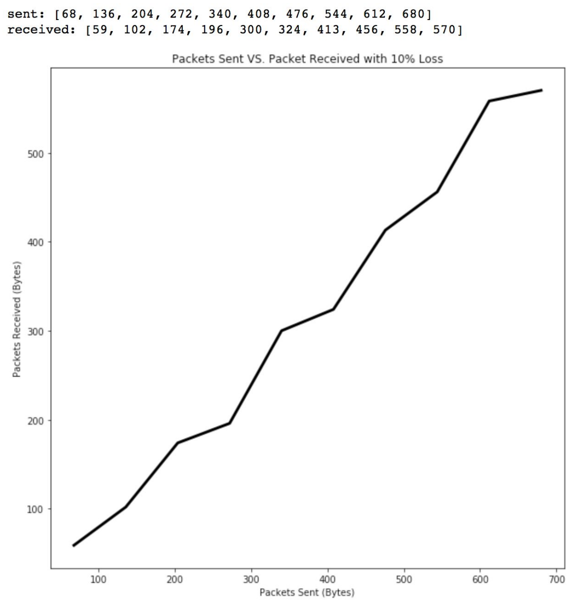
1) A node has left the network when another node has not received a heartbeat from said node for the duration of the timeout period (3 seconds).
2) When a Node leaves, the membership list size is *not* modified. Only the entry of that node is now set to not alive, and the local time of that node is invalidated and set to -1.

MP1 As Debugger

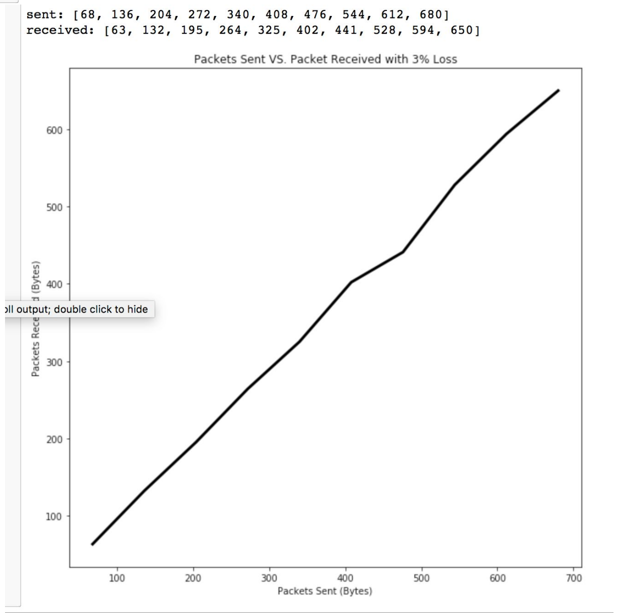
MP1 was very helpful as a debugger. Our MP2 code treated each vm as a servent and logged all node activity, ie. membership lists and node join/fail/rejoins. We then logged into the same vm's from a different set of terminals and ran the MP1 code we integrated into our MP2 codebase. We started nine servers and one client on the vm's and on the client we were able to grep our log files for node join/fail/rejoins



```
sent: [68, 136, 204, 272, 340, 408, 476, 544, 612, 680]
received: [38, 90, 150, 156, 245, 282, 266, 344, 450, 480]
```



```
sent: [68, 136, 204, 272, 340, 408, 476, 544, 612, 680]
received: [59, 102, 174, 196, 300, 324, 413, 456, 558, 570]
```



```
sent: [68, 136, 204, 272, 340, 408, 476, 544, 612, 680]
received: [63, 132, 195, 264, 325, 402, 441, 528, 594, 650]
```