COP5615 – Fall 2019
Project 2 – Gossip Simulator Bonus

Subham Agrawal | UFID - 79497379
Pranav Puranik | UFID - 72038540

Aim - Implementing Gossip and Push Sum Algorithms in Elixir.

Implementation-

To run bonus,
$ ./pushthegossip numNodes topology algorithm nodes_to_fail timeout

For this part, we are removing a certain number of “nodes_to_fail” nodes that will be given as input. The other parameter, “timeout”is the maximum amount of time we want the algorithm to run. Timeout should be more than 5000.

Working of failure model-

Initially, we remove nodes_to_fail from the network. We then start the algorithm by sending a message from the main process to a random node. We show the convergence reached by the algorithm at timeout. Note that the nodes in the network adjust their neighbours (cancel it from neighbour list), if they find that their neighbour does not exists.

Results:

Form 100 nodes and 10000 as timeout, the following **(% of nodes did not converge)** out of (numNodes gives as input) or (approx numNodes).

Points to note -

- For random 2D we are using 500 nodes as rand2D usually does not converge on 100 nodes.
- For 3D Torus, we are approximating the numNodes to the nearest cube. So, if input is 100, we are actually creating 125 node torus. So 8% of nodes not converged (value in cell of the table below) meaning 8% of 125 nodes did not converge. And we are killing only 10 nodes (10% of 100).
- Similar to 3D, in honeycombs we are approximating the numNodes to make a perfect honeycomb structure.

Gossip Algorithm -

% failed	Full	Line	Rand2D	3DTorus	Honeycomb	Random Honeycomb
0	0	0	0	0	0	0
10	10	86	10	8	10	10
20	20	94	20	16	21	20
30	30	94	30	24	57	31
40	40	96	40	32	86	46
50	50	99	50	40	99	55

Push Sum Algorithm -

% failed	Full	Line	Rand2D	3DTorus	Honeycomb	Random Honeycomb
0	0	0	0	0	0	0
10	10	82	11	12	28	25
20	20	86	23	27	36	35
30	30	97	35	32	63	49
40	40	95	42	43	88	100
50	50	100	52	46	92	85

Interpretations:

- Even if a single node fails in line topology, the other half side of the nodes cannot converge.
  ----o------o----o-----o-----x---o-----o------o-------o------- …
  The number of nodes converged for line will always be different for same % nodes failed depending on where the network breaks. Sometimes, we have a situation where both the neighbours of a node has failed, and we select that node from main process.
  -----o-----x----**O**-----x-----o-----
  In this situation, we have 0% convergence.

- Full always converges(excluding the failed nodes). Random 2d almost always converges(excluding the failed nodes). Here, since many nodes are connected to each other, there's always a neighbour for some node. That makes these networks a good choice, though full network may be difficult to implement.

- Also, the 3D torus converges completely (excluding the failed nodes). That means it is difficult to break a 3D torus and make two or more disconnected sets. 3D torus is a robust structure, which also converges quickly.

- Honeycombs are similar to line. If nodes in between shut down, the isolated ones do not converge. Although the results would vary every time, this network really has a low convergence.

- Random honeycomb has a connection to one different node. This sometimes saves the network, and performs better than simple honeycomb.