Bonus - Project 2

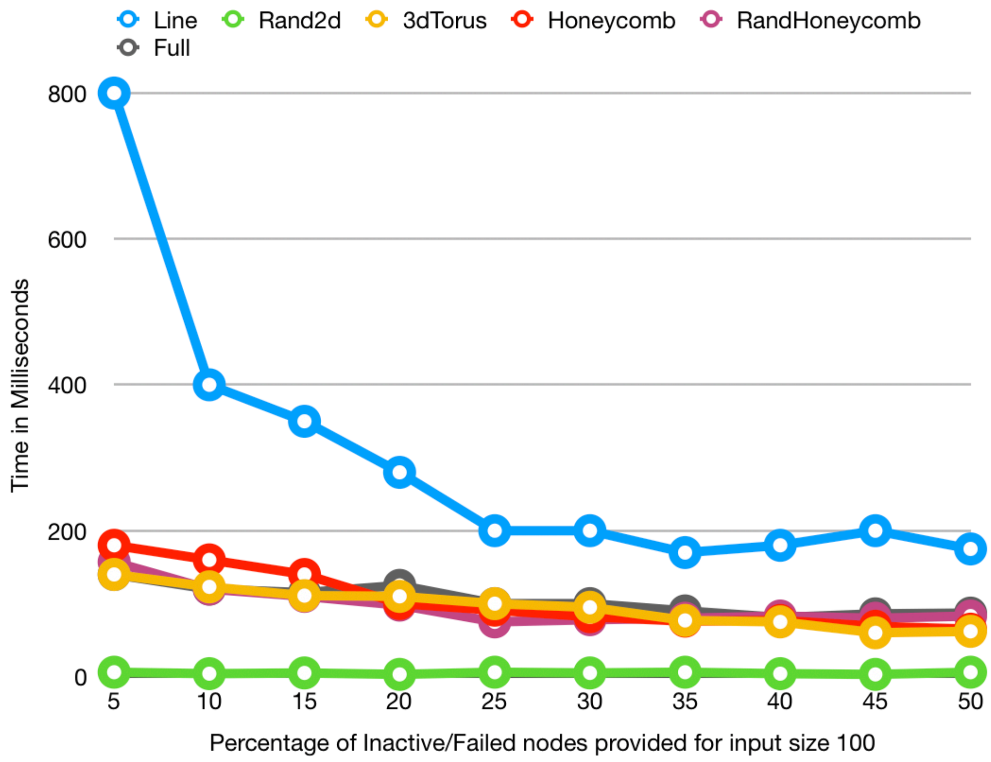
Group members -

Abhishek Jaimini - 7365-2858

Yash Kandalam - 3515-4584

Algorithm

We modified our code to also accept inputs with percentage of failed nodes too. We randomly selected the percentage nodes from all nodes and deactivated them. When a deactivated neighbor comes up, we replace with a random active node as a neighbor. We then calculated the convergence time for failure percentage ranging from 5 -50. Following plot was generated using the collected data. The working and steps to run can be found in Readme file in the same folder.

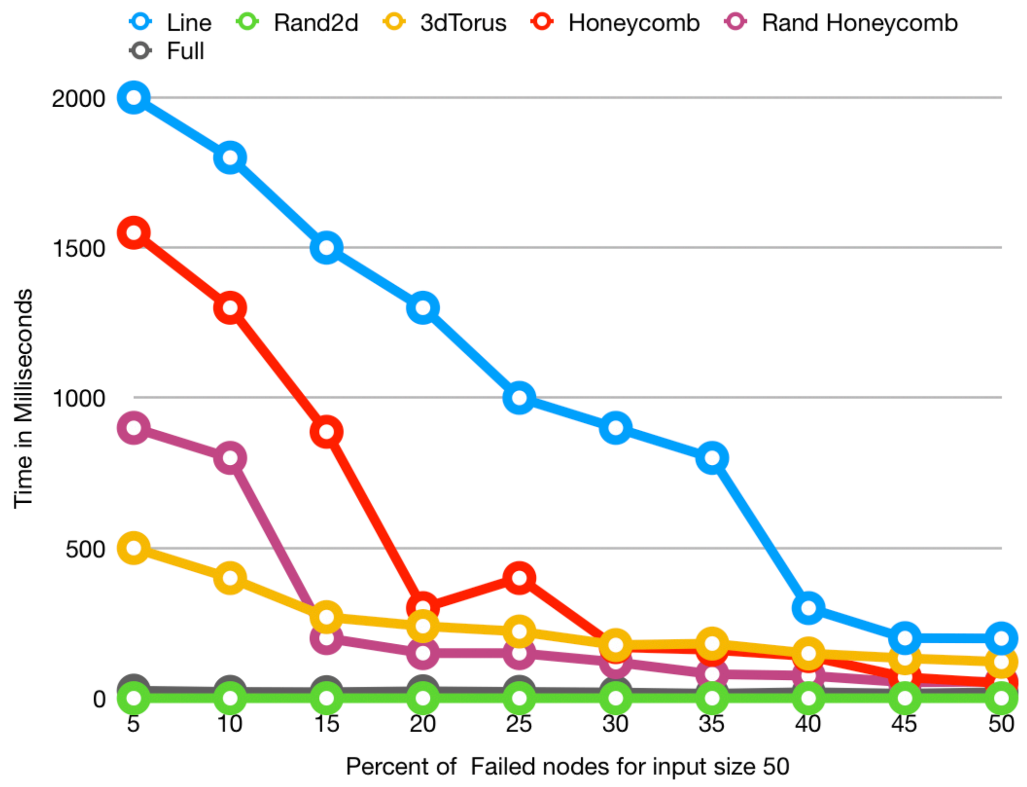


GOSSIP WITH FAILURE INPUTS

Observation - Using the plot, we observed that as Failure inputs increases, convergence time decreases. We are assuming that this happens because overall number of active nodes decreases, therefore, decreasing the input perceived. Line takes higher time than others. Rand Honeycomb performs slightly better than honeycomb.

PushSum Algorithm

We collected data for pushsum also and plotted on a graph. The input size was 50 nodes and we increased the failure nodes from 5 to 50 percent.



PUSHSUM WITH FAILURE INPUTS

Similar to Gossip, Line performed slowest. The overall performance increased with increase in failure nodes as they were replaced by neighbors remaining active nodes. Interesting thing to note was that Rand Honeycomb was better than Honeycomb. Adding one extra neighbor increased the performance for honeycomb topology as message could disperse faster using more neighbors.