COP5615 – Fall 2019
PROJECT - 2 BONUS

Group Members

Name	UFID
Floura Angel Nadar	2303-6958

Steps to run:

1. Unzip the file and navigate inside the folder
2. Open Terminal (with elixir installed)
3. For running:
   Executable file supported : ./gossip_protocol <numNodes> <topology> <algorithm> <percentage of failure node>
   example: ./gossip_protocol 100 full gossip 10

   Otherwise, build mix project using mix escript.build and follow the above steps

What is working for Bonus Question

Topologies	Gossip	Push-Sum
Full (full)	Working	Working
Line (line)	Working	Working
3D Torus (3Dtorus)	Working	Working
Random 2D (rand2D)	Working	Working
HoneyComb (honeycomb)	Working	Working
Random Honeycomb (randhoneycomb)	Working	Working

To Run Topology : (instead of topology)

1. Full - full
2. Line - line
3. 3D Torus - 3Dtorus
4. Random 2D - rand2D
5. Honeycomb - honeycomb
6. Random HoneyComb - randhoneycomb

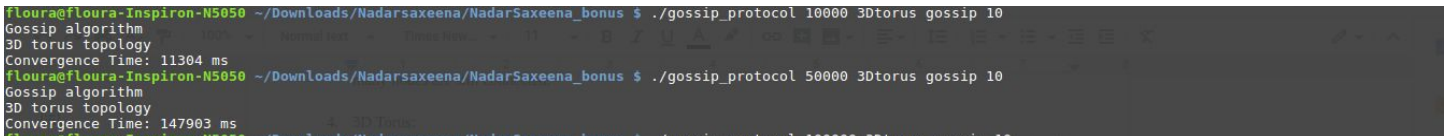
To Run Algorithm : (instead of algorithm)

1. Gossip - gossip
2. PushSum - push-sum

What is the largest network you managed to deal with for each type of topology and algorithm

Due to system constraint could not run for bigger values, but tried and tested for below with 10% failure:

Topologies	**Largest Achieved**
Full	10000
Line	10000
Random 2D	10000
3D torus	50000
Honey Comb	20000
Random HoneyComb	50000



Interesting Observations made on Topology Perspective:

1. Full:
   In this network, each node is connected to every other node in the network except itself. This algorithm has chances to converge faster if every node receives equal opportunity to receive a rumour, but if that fails then takes more time to converge. When failure percentage introduced, more the percentage more nodes were killed suddenly hence with remaining nodes the time and chances/ probability of equal distribution to receive a rumour increases.

   Eg. if probability of 1 node in 1000 = 1/1000, whereas 1 in 900 = 1/900 (faster)

2. Line:
   In line Topology, Each node is connected to before and after node except the first and the last node which is connected to only one node either before the node or after the node. An interesting observation made was that when the failure percentage was increased at some point the time taken to converge increased, whereas the other decreased. This may be because if a chain of nodes are removed then the other remaining chain can converge at a faster rate, or when there is just a single node left and its neighbors are dead then it can terminate itself because nobody to receive rumour from, whereas if only one node connected the process has to wait for that node to give all its rumour for 10 times, which the neighbor node can become inactive towards the particular node.

3. Random 2D:
   In Random 2D we place the coordinates of a node as x an y parameter and they have neighbours which are at a distance of 0.1. Similar to Line and Full, if the node is connected to many nodes then the chance of convergence increases on node failure of its neighbor, but in some cases it keeps waiting for rumour to be heard until its picked up by the neighbor. As failure node increases or decreases depending on the choice of failure neighbor and how many nodes are still connected.

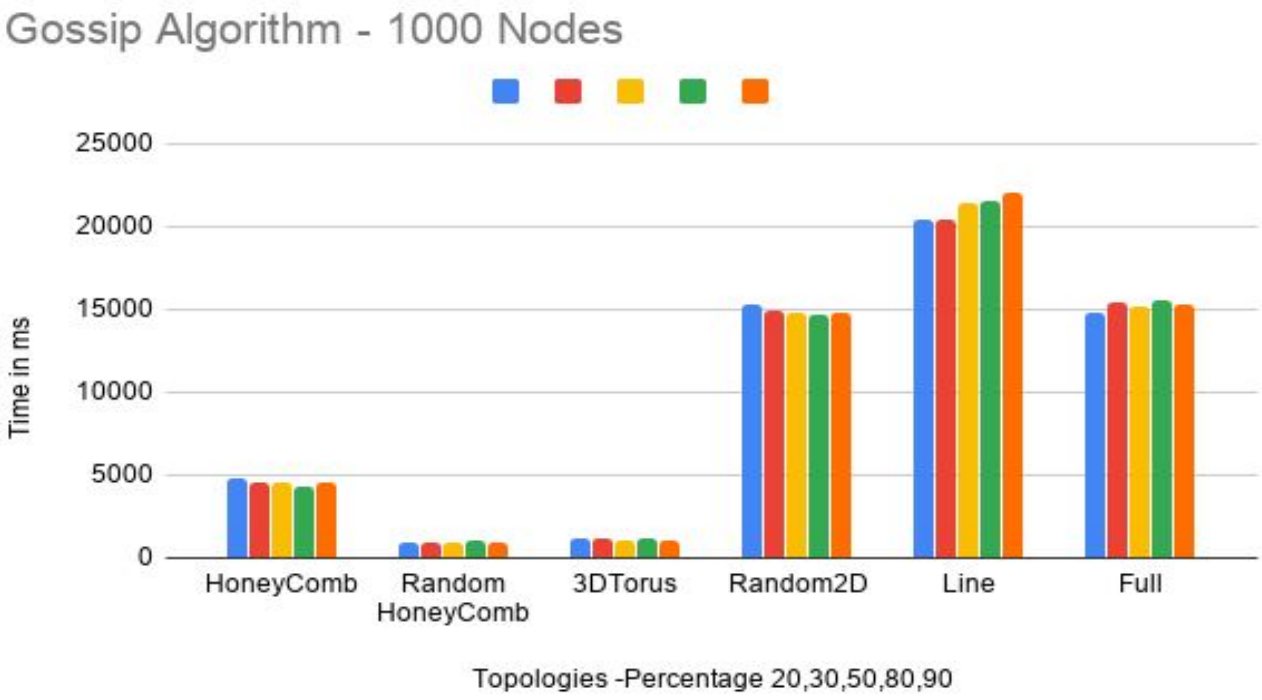
4. 3D Torus:
   In 3D Torus, the network is connected to 6 neighbors which can be represented as x+, x-, y+, y-, z+, z-. The network performs well as they converge faster because probability of rumour heard is ⅙, therefore they converge faster as compared to other nodes. When a percentage of failure is introduced, and if the nodes to fail are randomly picked then for more number of nodes failed, the network took more time to converge as the node has to hear from lesser number of nodes which may be connected to other nodes, hence if the priority of selection of a particular node is lesser than the time taken to converge is more.

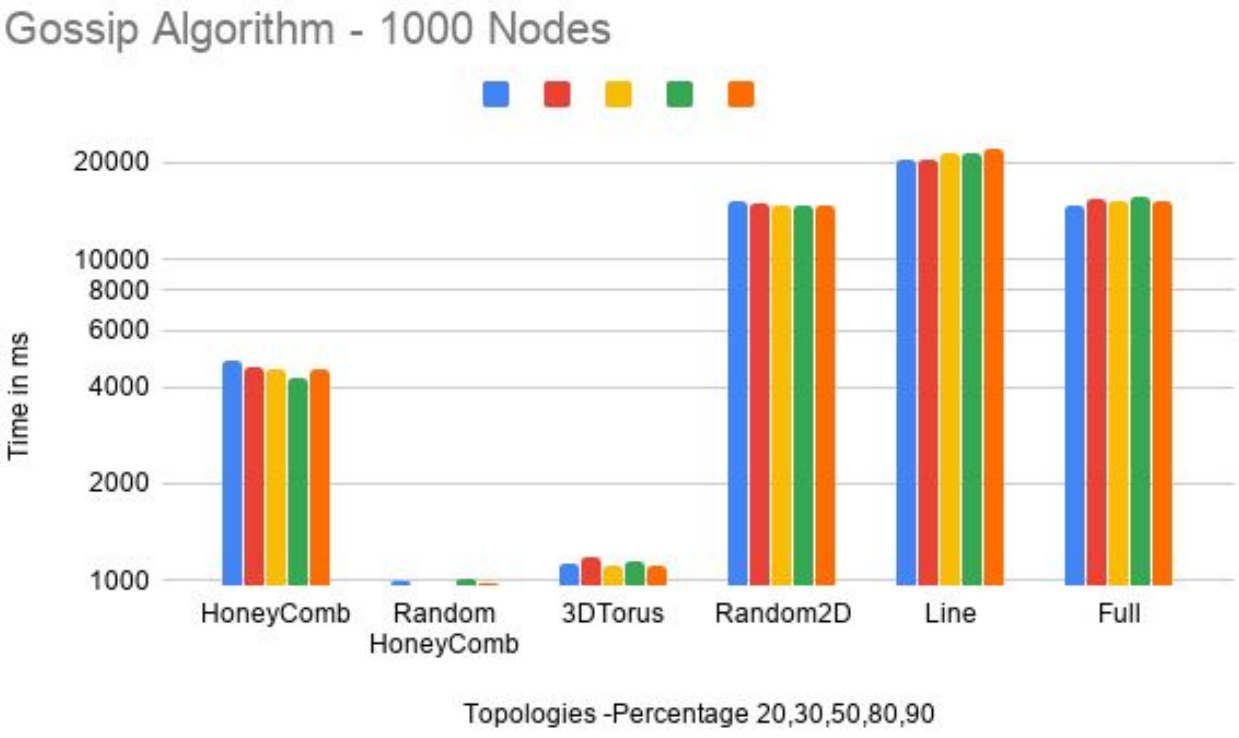
5. HoneyComb:
   HoneyComb networks have a maximum of 3 neighbors and can have 1 or 2 neighbors as well. The built network converges faster but not that relative to random honeycomb and 3D torus. Because it is connected in a sequential flow the time taken for a rumor to initially reach a node is more for honeycomb. Interesting observation made was as number of nodes failed increased the convergence time decreased.

6. Random HoneyComb:
   Like honeyComb network random honeycomb is connected to at most 3 neighbors and has an extra 1 neighbor which is randomly picked from the network other than itself or the ones who are already a neighbor. Random HoneyComb performs on par with 3Dtorus sometimes even better, because the chances of failure nodes all to fail at one time is not that frequent. An interesting observation made in both Random HoneyComb and honeyComb network is that it sometimes converges faster as if the nodes fails in a line then the node is left with no neighbor or the node is left with no parent to send a rumor and hence converges faster, whereas at other times it takes longer when it is left with just one parent to receive rumor from.

Charts for Observation:

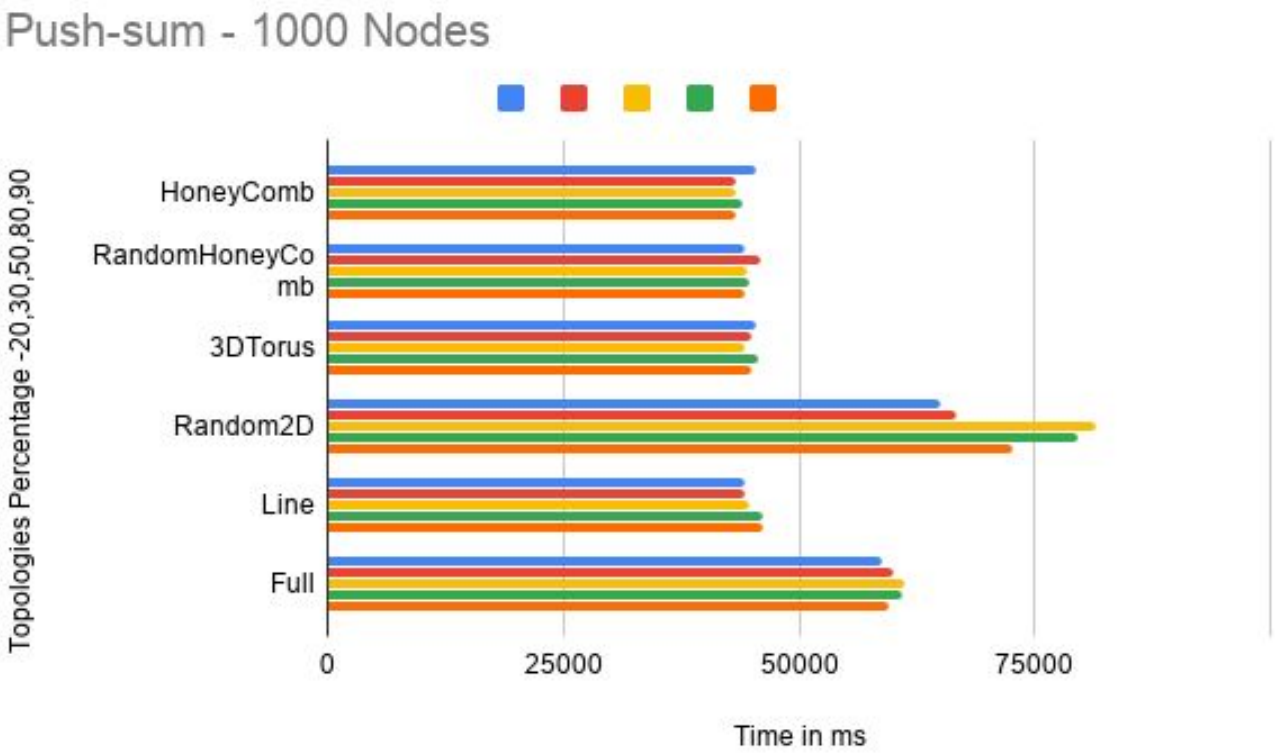
Gossip

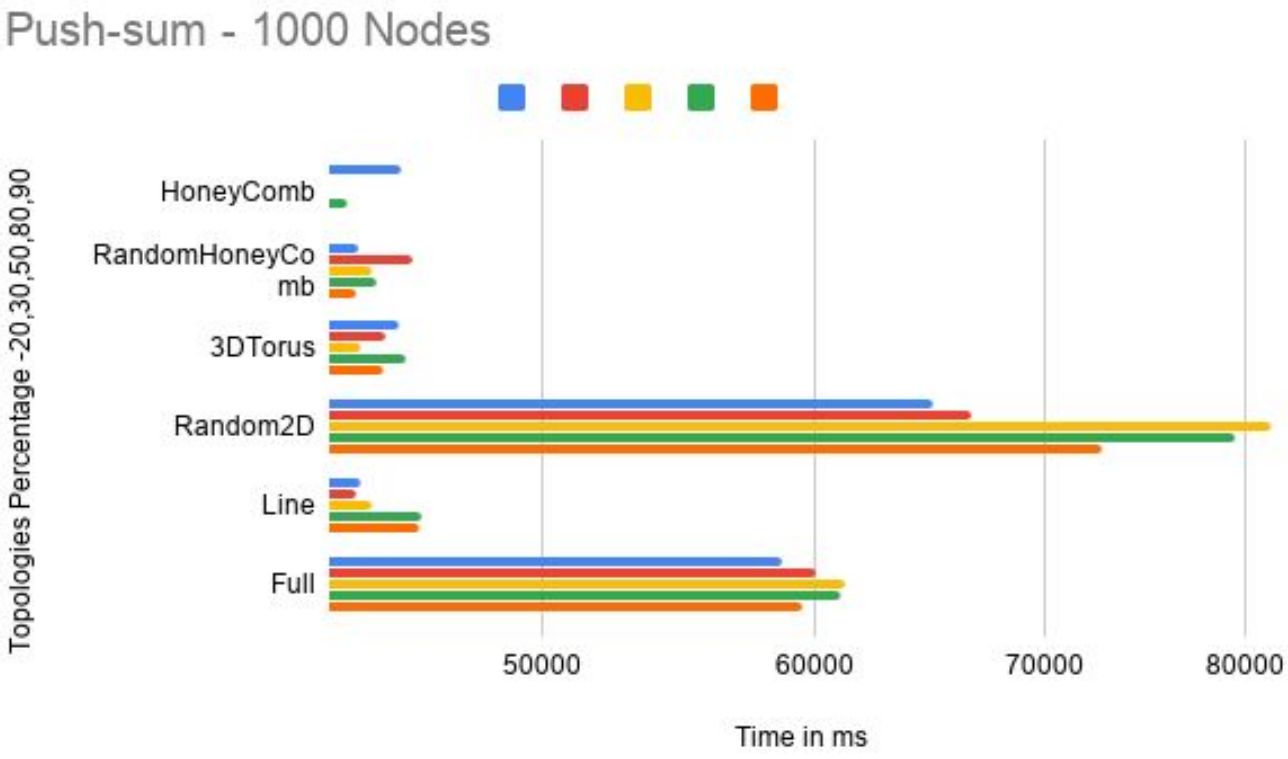




Logarithmic Scale

Push-Sum





Logarithmic Scale

Interesting Observations made on Topology Perspective:

1. Gossip Algorithm:
   Gossip Algorithm which is used to propagate information on a network. On trying to implement convergence of nodes when the number of times rumor is heard is 10, when failure model was introduced it took more time to converge in some cases whereas less in another, depending on the topology used.

2. Push-Sum Algorithm:
   Push-Sum was designed using the fact that if the average sum/weight ratio of a node is not changed(10^-10) for more than 3 times then it converges. On more number of failure nodes or on increase in percentage of failure, the algorithm in general took more time to converge.

How it was designed

1. Failure Model:
   A variable parameter with varying percentage of failure for each node. So when the node algorithm attains a gossip spread of count = numnodes/5, then the percentage(input) of nodes = number of nodes, is stopped abruptly. But it is handled but informing it's neighbor its no more active, hence the topology algorithm still takes time to converge remaining node.

2. Convergence Parameter:
   Convergence is achieved if the condition such as for gossip protocol when count > 10 or for push-sum when the difference in s/w < (10^-10) is three times. Also the algorithm is designed to inform it's neighbor that it would not be active anymore, hence if its neighbors are dead, the node is terminated.

3. Tested:
   The failure model was tested by varying the percentage of failure (20, 30, 50, 80, 90). The time for convergence was recorded and the above inferences were made.

4. Convergence Time:
   Convergence time is calculated by the difference of the start of node time to that of its end when it is converged in milliseconds.

Overall Observations made:

1. Gossip propagation takes less time to propagate as number of nodes increases for even its worst topology which takes more time as compared to others as compared to push-sum. Ie. Line Topology
2. HoneyComb Random have the highest probability of converging at a faster rate, followed by 3D torus and HoneyComb topology.
3. Line Topology failure is mainly because that the neighbors of a particular node are deleted quickly as they are in sequential order, which results in failing to converge at a faster time
4. Since we used the concept of node to terminate if its neighbor node is not active the converging was achieved faster, rather than waiting for all nodes to converge.
5. General convergence time sequence Random HoneyComb □ 3D Torus < HoneyComb < Random 2D < Full □ Line.
6. In Gossip Algorithm, Random 2D showed a decreasing time required to merge as percentage of failure increased
7. HoneyComb takes least amount of time as compared to other algorithms for Push-Sum with failure.

Table used for Chart:

Gossip Algorithm						
Percentage of Failure	HoneyComb	Random HoneyComb	3DTorus	Random2D	Line	Full
20	4831	991	1128	15281	20391	14821
30	4606	958	1188	14887	20463	15467
50	4526	970	1105	14781	21467	15168
80	4249	1012	1155	14644	21561	15561
90	4546	984	1113	14774	22017	15263

Push-Sum Algorithm						
Percentage of Failure	HoneyComb	RandomHoneyComb	3DTorus	Random2D	Line	Full
20	45534	44225	45410	64956	44287	58752
30	43356	45857	45059	66652	44148	60022
50	43389	44599	44272	81510	44629	61265
80	43914	44783	45614	79517	46106	61035
90	43342	44177	44925	72703	46068	59513