COP5615 - Distributed Operating System Principles
Project 2: Gossip Simulator

Project Contributors:

1. ***Saksham Bathla** UFID: 82967553*
2. ***Arnav Gaur** UFID: 83249716*

Running the Code:

```
dotnet fsi --langversion:preview proj_2.fsx 1000 line push-sum
```

Here, 1000 is the number of nodes, line is the topology and push-sum is the algorithm

Topology

- Full Network: Every actor can directly communicate with any of the other actors.
- Line: An actor can talk to only two actors, directly adjacent to it in the line topology
- 3D:
  An actor can only talk to grid neighbours in 3d. We have simulated this using a single list in the following way.
  1. We choose the next highest perfect cube nearest to the number of nodes, N
  2. We assume there are N^⅓ layers of N*N size.
  3. So, for example, for N=27, 1-9 is a 3*3 layer, 10-18 is the second layer and 19-27 is the third layer
  4. Now, to select neighbours, the neighbors in the same layer are i+1, i-1, i+N^⅓, i-N^⅓ (if these numbers lie on the same layer and within the range 1..N)
  5. Neighbours in the upper and the lower layer are i+N^⅔ and i-N^⅔ (if there are upper and lower layers)
- Imperfect 3D Grid: An imperfect 3D grid has one extra random neighbor selected from all the actors added to each node

Results

Gossip

- Full Network

Number of Nodes	100	500	1000	2000	10000
Time (ms)	991	1100	1550	2166	Out of memory

- 3D Grid

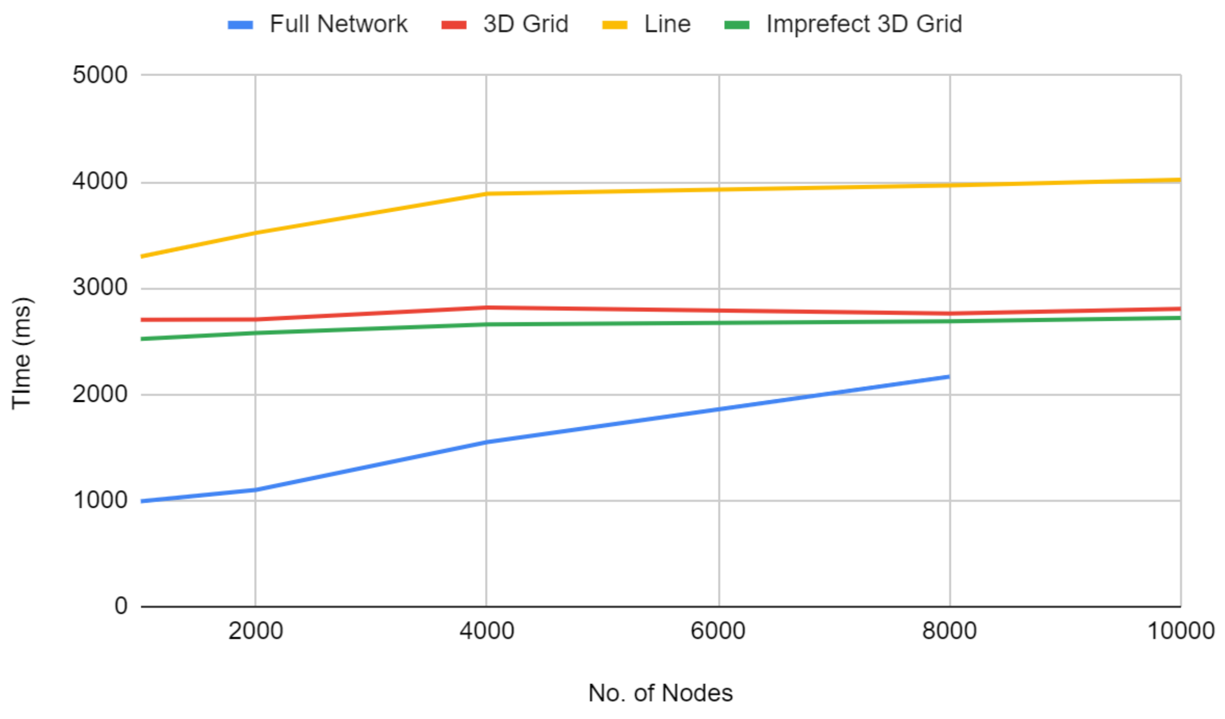
Number of Nodes	1000	2000	4000	8000	10000
Time (ms)	2702	2705	2817	2760	2805

- Line

Number of Nodes	1000	2000	4000	8000	10000
Time (ms)	3296	3520	3887	3966	4021

- Imperfect 3D Grid

Number of Nodes	1000	2000	4000	8000	10000
Time (ms)	2520	2576	2658	2687	2720



Push-Sum

Full Network

Number of Nodes	100	500	1000	2000	10000
Time (ms)	7557	8102	10524	10762	Out of memory

3D Grid

Number of Nodes	1000	2000	4000	8000	10000

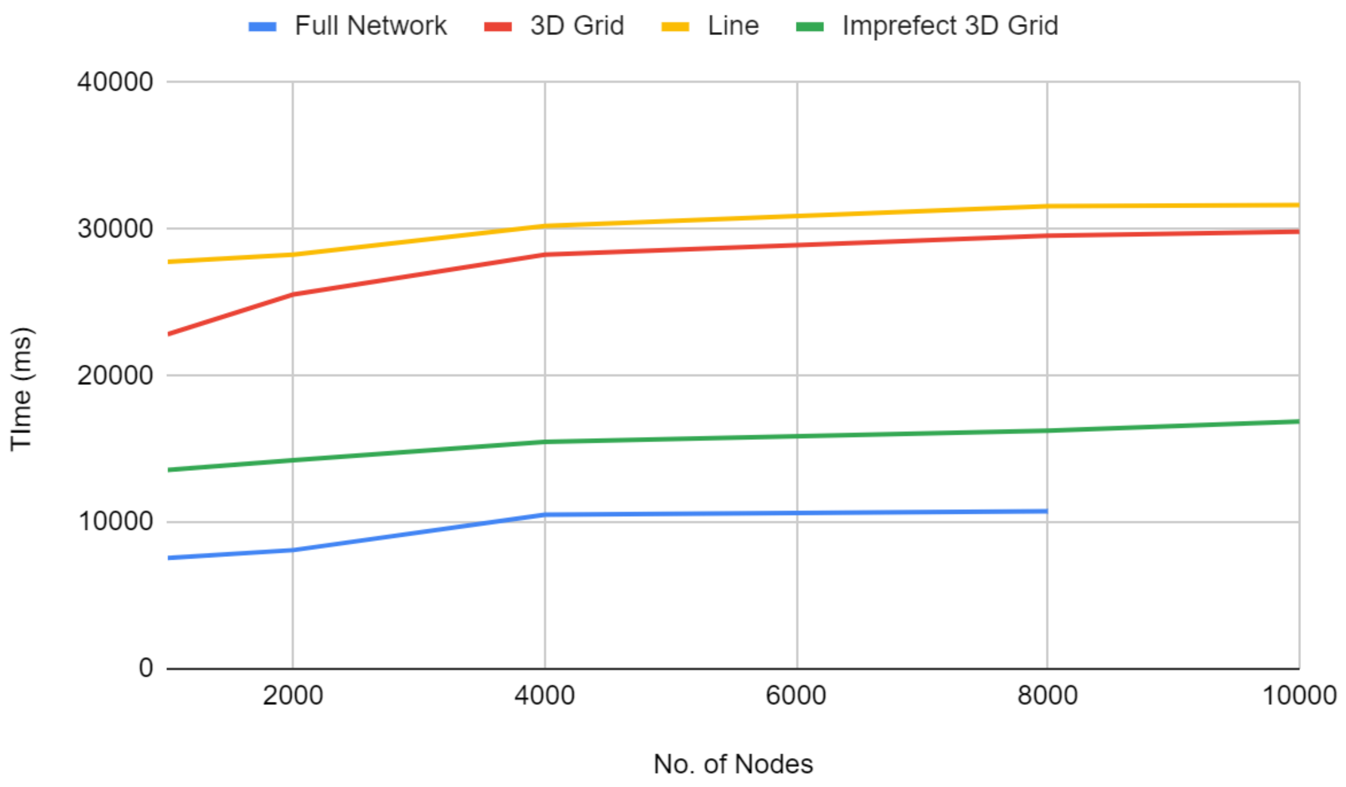
Time (ms)	22837	25550	28269	29549	29841

Line

Number of Nodes	1000	2000	4000	8000	10000
Time (ms)	27773	28274	30223	31568	31649

Imperfect 3D Grid

Number of Nodes	1000	2000	4000	8000	10000
Time (ms)	13579	14235	15489	16252	16874



Discussions

a) Gossip

- We noted that with the given parameters, the algorithm performs differently on the four topologies.
- In terms of convergence time we note that: **full < imp3D < 3D < line.** This is because convergence time is inversely proportional to the number of neighbours for each node
- This is because a higher number of neighbours means faster propagation of the gossip.
- But, a network like a fully connected network is very expensive in terms of memory.
- To conclude, **an imp3D network stands to be the most efficient** considering the memory space and speed trade-off.

b) Push-Sum

- We noted that with the given parameters, the algorithm performs differently on the four topologies.
- It is safe to say that in terms of convergence time we note that: **full < imp3D < 3D < line.** This is because in line each node can talk to any other node in full network, hence spreading the rumor faster. The algorithms prove to be the most efficient and take lesser time in a full network.
- We note that the push-sum algorithm takes more time compared to the gossip.
- This can be attributed to the fact that each actor on average has to receive more messages in order to converge when compared to gossip.

<u>Bonus</u>

c) Implementation

For Bonus implementation, we simulated the node failure by taking a command line argument to specify the ratio of nodes to fail.

Now to run the code, we run the command like:

```
dotnet fsi --langversion:preview project_2_bonus.fsx 1000 line push-sum 0.1
```

We see that the convergence times increase as the number of failing nodes increase. Also, for both Gossip and Push-Sum algorithms, there is a possibility of network partitioning if all the neighbours of some nodes fail.

Also, in the push sum algorithm, we cannot guarantee the information correctness in case of node failure after starting the transmission. This is the case because some information can be lost.

To combat the partitioning, we can do one basic change to make sure that the disconnected components of the networks converge within themselves. This can be done by sending the message to the same actor / only the non converged / non failed neighbours.