Project 2 Bonus Report

Group Members

Balaji Balasubramani - 98763981

Divyareddy Surapa Reddy - 13727408

GOSSIP

Introduction:

Implemented the failure model by deleting randomly the number of nodes which is provided by the user as one of the inputs. Observed how much time the remaining nodes have taken to converge.

Maximum Number of Nodes Considered: 1000

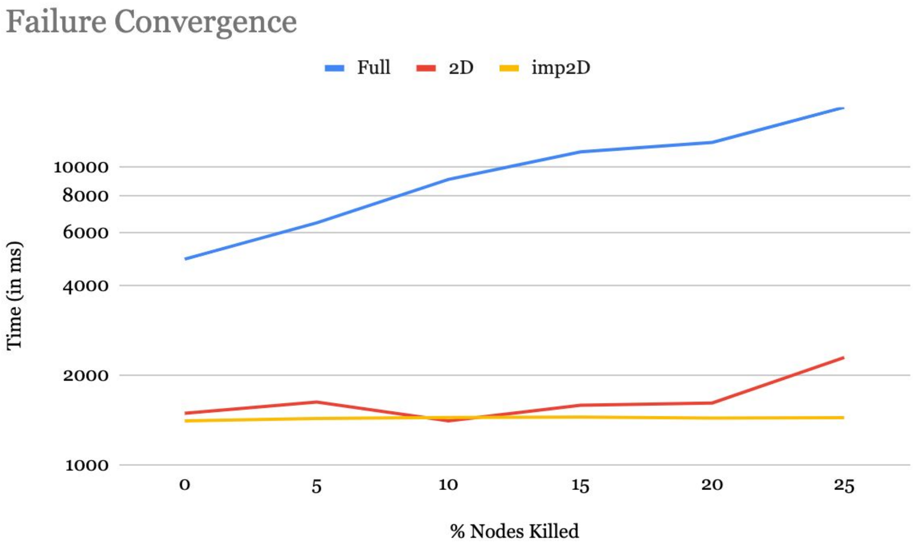
Graphs:

- The consolidated graph containing results from 3 (Full, 2D, Imperfect 2D) network topologies with different numbers of nodes is constructed.
- The graphs are plotted with the number of nodes in X axis vs time taken in Y axis.

Failure Model Convergence:

% Nodes Killed	Full	2D	Imperfect 2D	Line
0	4893	1490	1405	8341
5	6472	1624	1431	*
10	9049	1405	1441	*
15	11190	1583	1446	*
20	12037	1612	1434	*
25	15791	2290	1440	*

** - Line Does Not Converge*



Detailed Findings:

1. Convergence was noticed for the remaining nodes present in the 2D, Imperfect 2D and Full network topologies.
2. But, it **failed to converge for Line** topology. For instance, we had initially taken 200 nodes and removed 1 node. Once, the random node that was picked to be deleted was 100, it is observed that 1-99 had converged and the remaining didn't as it did not receive any gossip message.
3. Even Killing more than 10% of the nodes, the line topology didn't converge
4. Full, 2D, Imperfect 2D topologies handled the node failures correctly and converged even when the number of nodes killed were extremely large.
5. When the number of failure nodes percentage increased above 50%, we observed that the algorithm did not converge.
6. Imp2D was converging the fastest then followed by 2D and then Full.(imp2D < 2D < Full)

Implementation:

Input:

1. numNodes - Number of nodes(actors) in participation, 1000 always in this case
2. topology - One of the networks: full, 2D, imp2D and line
3. algorithm - Either gossip or push-sum
4. removeNodes - Remove the nodes from the participation List.

Command to run:

```
dotnet fsi --langversion:preview project2Bonus.fsx 1000 <topology>  <algorithm> <remNodes>
```

Initial Steps

1. Built the topology by constructing neighbors list for each actor during its initiation. The neighbors list consists of only the remaining nodes present in the network.
2. Now, will start the given algorithm by randomly selecting a single actor/participant.

Algorithm

3. The information of the actor's neighbors are stored as a list (neighborsList) which is constructed during its initiation
4. Upon external rumour message arrival (from one of its neighbors actors), the actor sends the rumour message to one of its neighbours and also maintains its own state (external message count) of tracking the number of times rumour message is received by incrementing it by 1.
5. Also at a periodic interval it sends the rumour message to itself (self message) from the time the first message to it is received until it terminates.

Termination

6. The program terminates when all of its actors receive the rumour message exhaustion number of times.