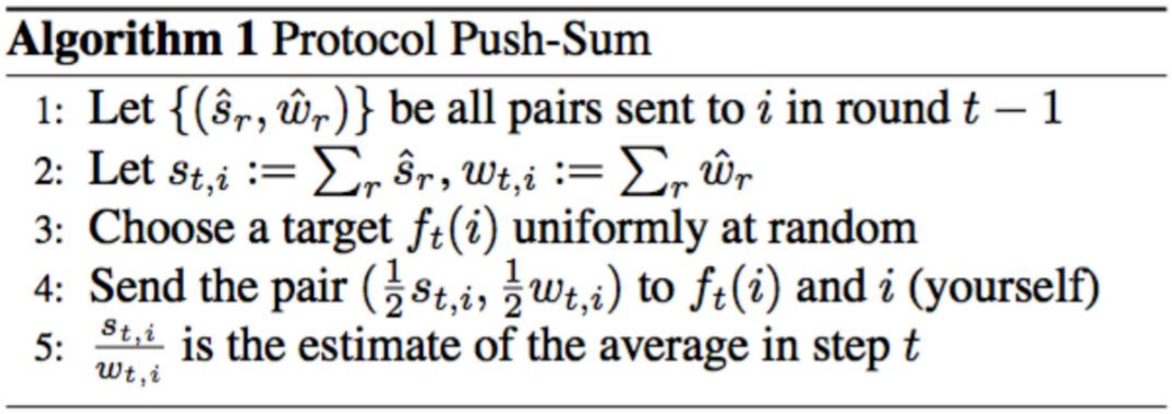
DOS Project 2

Aniketh Sukhtankar (UF ID 7819 9584)
Shikha Mehta (UF ID 4851 9256)

Brief Description:

This problem is aimed at implementing the Gossip and Push-Sum algorithms for information propagation. These algorithms are in turn implemented on various topologies like full, 2D, imperfect 2D and line.

- The Gossip protocol works by initiating the process from a single actor which forwards the message to the other actors. The point of convergence reached is when each of the actors listens to the message 10 times.
- The Push-Sum algorithm works by sending messages in the form of pairs(s,w) where s is the value of the actor number and w = 1 for each actor. The propagation converges when the s/w ratio doesn't change when compared to a pre-defined value (10^-10 in our case) for three consecutive times.



Algorithm 1 Protocol Push-Sum

```
1: Let $\{(\hat{s}_r, \hat{w}_r)\}$ be all pairs sent to $i$ in round $t-1$
2: Let $s_{t,i} := \sum_r \hat{s}_r, w_{t,i} := \sum_r \hat{w}_r$
3: Choose a target $f_t(i)$ uniformly at random
4: Send the pair $(\frac{1}{2}s_{t,i}, \frac{1}{2}w_{t,i})$ to $f_t(i)$ and $i$ (yourself)
5: $\frac{s_{t,i}}{w_{t,i}}$ is the estimate of the average in step $t$
```

A glimpse of the network topologies:

- A full network is a topology in which a message is sent from one actor to any other actor in a random fashion.
- 2D network is a topology in which an actor sends messages to its immediate matrix neighbours.
- Imperfect 2D behaves like a 2D network except that it sends an extra message to a random neighbour in the network.
- Line topology involves sending messages back and forth to an actor's front and back neighbours.

Running the program

GOSSIP

- enter the number of nodes (will be rounded up for 2d and imp2d topologies)

- enter the topology

- enter the algorithm
- (optional) enter the number of nodes triggered at initiation
- (optional) enter the stopping criteria (eg. 100 if there are a 1000 nodes in the network and a convergence rate of 90% i.e. 900 nodes converging is good enough)

BONUS

- enter the number of nodes (will be rounded up for 2d and imp2d topologies)
- enter the topology
- enter the algorithm
- enter the number of nodes to kill (used this feature for testing and analyzing various topologies and algorithms)
- (optional) enter the number of nodes triggered at initiation
- (optional) enter the stopping criteria (eg. 100 if there are a 1000 nodes in the network and a convergence rate of 90% i.e. 900 nodes converging is good enough)

Expected Results:

From the description of the Network topologies whether it be Gossip or Push-Sum the order in which the time taken to converge from less to higher time is given by *full < imperfect 2D < 2D < line*. The time taken to converge is calculated by the difference in system times when the process is initiated until all the actors terminate. This is in the order of milliseconds.

Implementation Details:

Main.ex: This elixir file is the entry point and hosts the main method. Our implementation takes 3 parameters in the order - NumNodes, Topology, Algorithm

- NumNodes: specifies the number of actors in the network
- Topology: stands for the name of the topology, i.e. line, full, 2d or imp2d
- Algorithm: stands for the algorithm used, i.e. gossip or pushsum

Implementation.ex: This file contains the code for distinguishing the combinations of the topology and the protocol. Depending on the input parameters, the topology and the protocol are implemented here. Apart from this, the logic for stopping the actors once they have reached their convergence criteria and printing the time taken can also be found in this file.

Gossip Actors: This file consists of the information pertaining to a single gossip participant. This includes the information of its neighbours which is stored as a list (neighbourList) passed to it upon initiation. The underlying logic of maintaining process state is done by the process registry that keeps track of various actors. The implementation maintains a counter to track the number of times it received the rumor message and triggers termination when the counter hits 10. A separate process in the actor iteratively sends the rumour message to one of its neighbours every 100 milliseconds, from the time it received its first message up until the actor terminates.

Push-Sum Actors: This file consists of the information pertaining to a single push-sum participant. This includes the information of its neighbours which is stored as a list (neighbourList) passed to it upon initiation. The implementation maintains state as the pair (s,w), and iteratively sends half of it to one of its neighbours while updating its own state to the remaining half. Note that the nature of this implementation helps ensure convergence, as each node updates its state after waiting for a maximum of 100 milliseconds to receive any pairs from its neighbours.

Actual Results:

The time taken vs the number of actors is plotted as a graph for each of the network topologies. The expected ordering of timing with each of the topology has been fairly met. The values and the consolidated graphs have been disclosed as shown below:

Full Topology - Gossip

No. of Nodes	Convergence Time (milliseconds)
50	2188
150	2406
350	2625
550	2954
650	2844
850	2860
1000	3062

Imp 2D Topology – Gossip

No. of Nodes	Convergence Time (milliseconds)
50	2531
150	2734
350	2844
550	3297
650	2953
850	2844
1000	3718

2D Topology - Gossip

No. of Nodes	Convergence Time (milliseconds)
50	3172
150	3281
350	4500
550	4610
650	3843
850	5282
1000	5469

Line Topology - Gossip

No. of Nodes	Convergence Time (milliseconds)
50	6140
150	16422
350	23969
550	47812
650	48532
850	79750
1000	80718

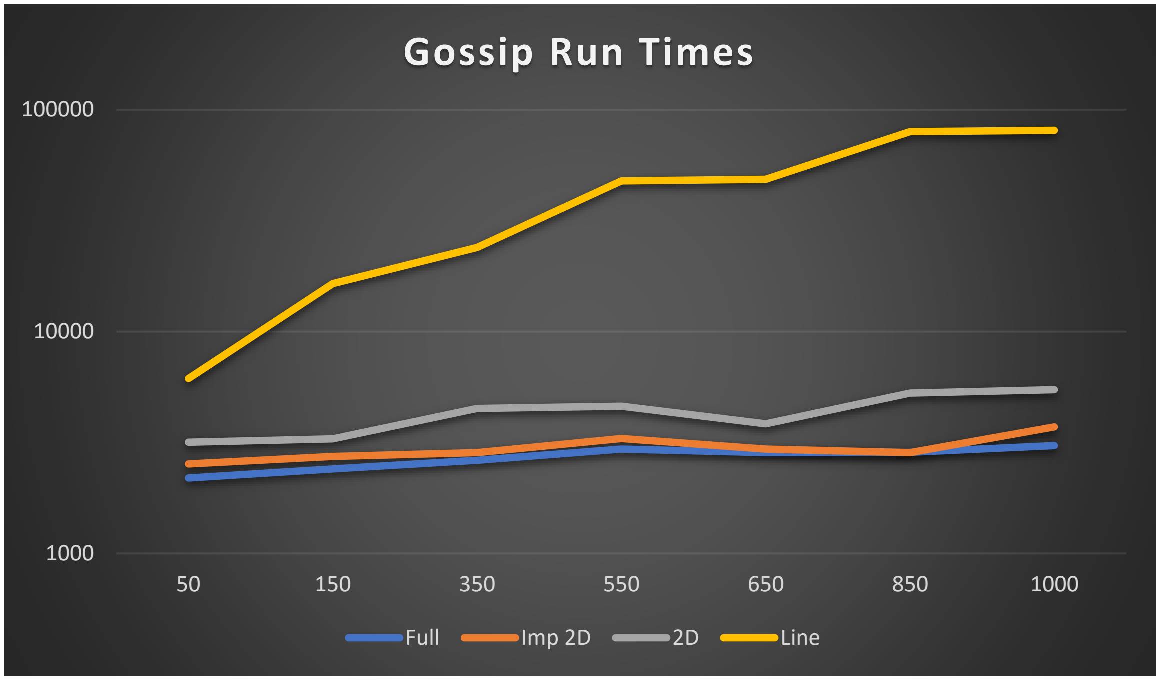


Figure 1: **Run Time for Gossip Protocol for Different Topologies**

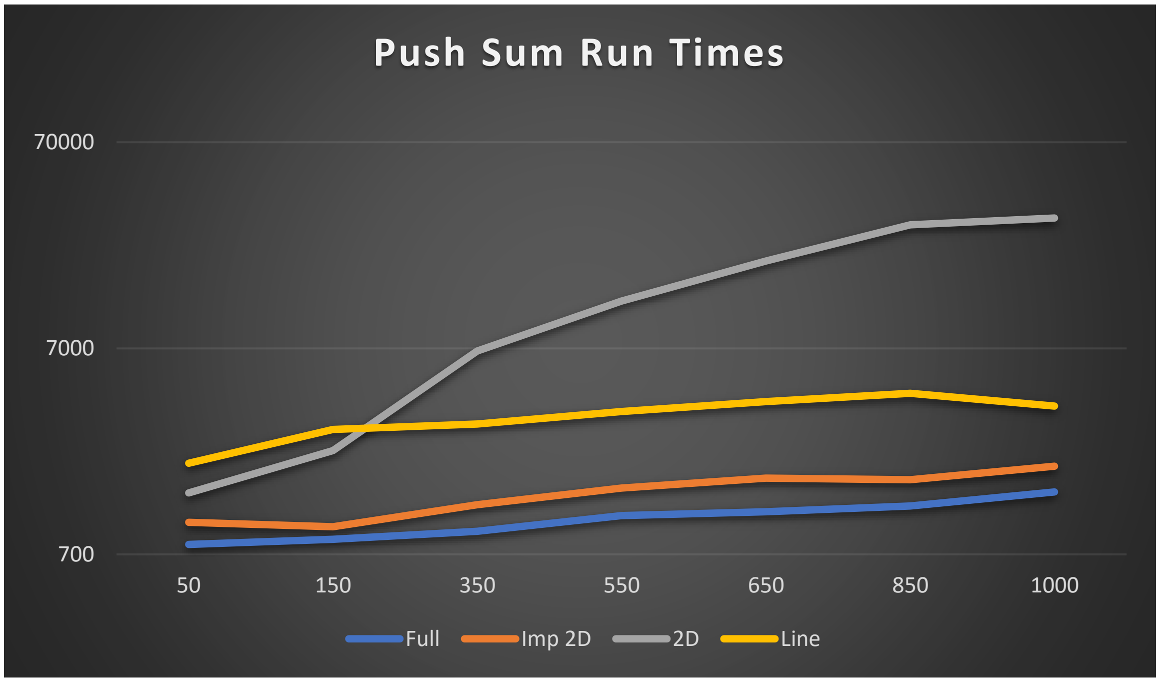


Figure 2: **Run Time for Push Sum Protocol for Different Topologies**

Full Topology Push Sum

No. of Nodes	Convergence Time (milliseconds)
50	782
150	828
350	906
550	1078
650	1125
850	1203
1000	1406

Imp 2D Topology Push Sum

No. of Nodes	Convergence Time (milliseconds)
50	1000
150	953
350	1219
550	1468
650	1641
850	1609
1000	1875

2D Topology Push Sum

No. of Nodes	Convergence Time (milliseconds)
50	1391
150	2234
350	6781
550	11875
650	18578
850	27828
1000	30016

Line Topology Push Sum

No. of Nodes	Convergence Time (milliseconds)
50	1938
150	2828
350	3000
550	3453
650	3864*
850	4234*
1000	3675*

*Algorithm had to be re-run as it failed multiple times in converging

Interesting Observations:

From the above experiments, one interesting observation we had was that after crossing a certain threshold (around 200 nodes), the Push-Sum algorithm consistently converged faster using a Line topology as compared to using a 2D topology. Apart from this deviation, the order of convergence of the network topologies from the results obtained can be fairly shown as

T(full) <T(imperfect 2D) < T(2D) < T(line)

Another observation we made while preparing the README file is how the maximum number of nodes (in a network) vary based on the topology and algorithm in use.

The kind of convergence issues that emerged while implementing this project have inculcated a good understanding of the gossip and push-sum protocols. Line topology has been the most difficult one to converge as the number of neighbors is limited for an actor and the message propagation is not that frequent as it is with the other networks. In the Bonus part of our project, we have implemented 2 approaches based on the following parameters to handle such convergence issues.

1. the number of nodes that the main process initially invokes
2. the criteria for convergence, i.e. the number of nodes left to converge before which we may declare convergence

Apart from the above parameters, the Bonus project implementation is also capable of trapping node failures and/or loss of connection among nodes.

References:

- https://en.wikipedia.org/wiki/Gossip_protocol
- http://www.inf.fu-berlin.de/lehre/WS11/Wireless/Lectures/lecture13.pdf
- https://github.com/nkkosuri/Stimulation-of-Gossip-and-Push-Sum-Algorithms/blob/master/projectReport.pdf
- https://elixir-lang.org/