Project 2

COP5612 Fall 2022

Implementation of Gossip & Push Sum Protocol

Team Members:

- Namita Namita (UFID- 48479313)
- Shriyans Nidhish (UFID- 19616510)

Problem Statement:

This project aims to explore the convergence of Gossip algorithms, which can be applied to both group communication and aggregate computation. The convergence will be analyzed using a simulator developed in Erlang, utilizing asynchronous actors. Specifically, the focus will be on the implementation of the Asynchronous Gossip algorithm, taking advantage of the asynchronous nature of Erlang's actors.

Implementation:

The implementation of the gossip algorithm in Erlang involves creating a network of actors across four different topologies. The objective is to analyze the convergence time of each topology based on the size of the network. The topologies are:

- Line: All the actors are placed in a linear order forming a line, where an actor at position 'i' has neighbors at positions i+1 and i-1. Maximum neighbors possible are 2 and minimum is 1.

- Full: Every actor is a neighbor of all other actors. That is, every actor can talk directly to any other actor.

- Random 2D: Actors are randomly position at (x, y) coordinates on a [0 - 1.0] x [0 - 1.0] square. Two actors are connected if they are within 0.1 distance to other actors. This is an interesting case which have lot of degree of randomness here with unsure maximum and minimum neighbors.

- Imperfect 3D: Actors form a 3D grid. An imperfect 3D topology refers to a network configuration where nodes are organized in a three-dimensional space, but the connections between nodes may not adhere to a perfectly structured pattern.

Run Instructions:

1. Start erlang compiler by typing "erl".
2. Compile the file "gossipPushsum.erl"
   Command: c(gossipPushsum).
3. Run the file one by one for gossip and pushsum.
   Command: gossipPushsum:main(100, line, gossip).
   Command: gossipPushsum:main(100, line, pushsum).

Algorithm:

Gossip Algorithm for information propagation

- Starting: A participant(actor) is told/sent a rumor(fact) by the main process.
- Step: Each actor selects a random neighbor and tells it the rumor.
- Termination: Each actor keeps track of rumors and how many times it has heard the rumor. It stops transmitting once it has heard the rumor 10 times (10 is arbitrary, you can select other values).

Push-Sum algorithm for sum computation

- State: Each actor A_i maintains two quantities: s and w. Initially, $s = x_i = i$ (that is actor number i has value i, play with other distribution if you so desire) and $w = 1$
- Starting: Ask one of the actors to start from the main process.
- Receive: Messages sent and received are pairs of the form (s; w). Upon receive, an actor should add received pair to its own corresponding values. Upon receive, each actor selects a random neighbor and sends it a message.
- Send: When sending a message to another actor, half of s and w is kept by the sending actor and half is placed in the message.
- Sum estimate: At any given moment of time, the sum estimate is $\frac{s}{w}$ where s and w are the current values of an actor.
- Termination: If an actor's ratio $\frac{s}{w}$ did not change more than 10^{10} in 3 consecutive rounds the actor terminates.

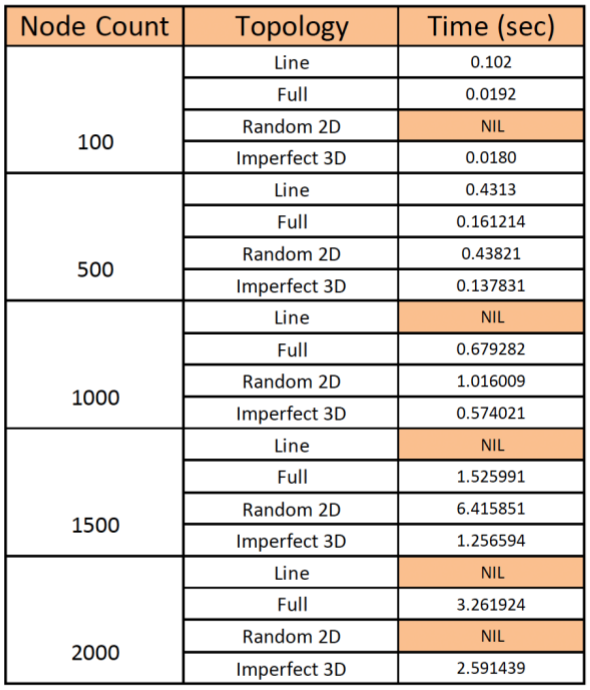
WARNING: the values s and w independently never converge, only the ratio does.

Results:

Gossip Protocol:

In our implementation, the Gossip Algorithm converges once all nodes in the network have heard the rumor 10 times. When all nodes hear the rumor 10 times, it stops broadcasting it until all nodes continue to disseminate it. When convergence is obtained, the program exits and reports the overall time necessary to reach convergence in milliseconds.

Below table shows the convergence time for each type of network with gradual increase in network size:



Node Count	Topology	Time (sec)
100	Line	0.102
	Full	0.0192
	Random 2D	NIL
	Imperfect 3D	0.0180
500	Line	0.4313
	Full	0.161214
	Random 2D	0.43821
	Imperfect 3D	0.137831
1000	Line	NIL
	Full	0.679282
	Random 2D	1.016009
	Imperfect 3D	0.574021
1500	Line	NIL
	Full	1.525991
	Random 2D	6.415851
	Imperfect 3D	1.256594
2000	Line	NIL
	Full	3.261924
	Random 2D	NIL
	Imperfect 3D	2.591439

Fig1: Table indicating network size, topology and convergence time.

The ‘NIL’ values denotes that we could not make the topologies converge and get an output. The reason being Gossip, as a protocol, performs best for the Full topology where all the nodes are interconnected to each other.

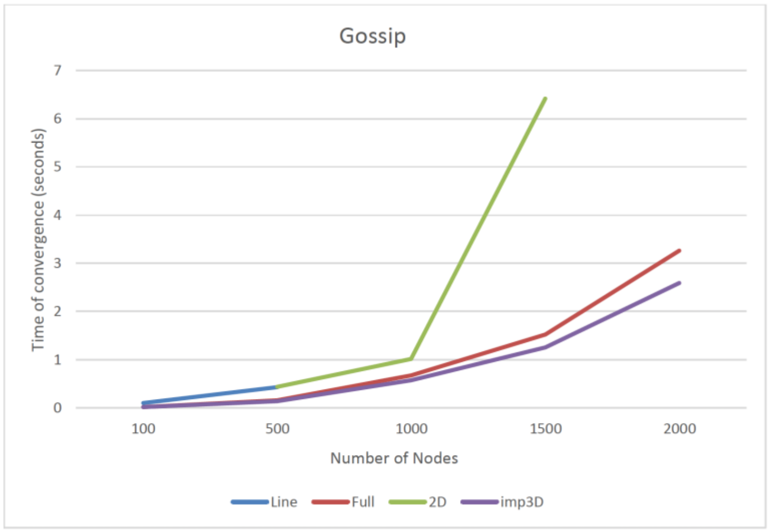


Fig2: Graph indicating network size, topology and convergence time.

The above graph plots the convergence time of different topologies vs the size of the network for Gossip protocol.
Inference from the graph:

- The imperfect 3D topology has the lowest convergence time for all topology networks.
- Full topology has little larger than imp3D convergence time for all types of networks whereas random 2D for larger networks its convergence time increases exponentially due to large memory occupied by the adjacent node lists for every node. Surprisingly, line topology didn't give better performance as adjacent nodes are connected and forms a long chain which takes long to compute and achieve convergence.

Algorithm	**Topology**	**Largest number of nodes attained**
Gossip	Line	500
	Full	2000
	Random 2D	1500
	Imp3D	2000

Fig3: Table indicating largest network size attained in each topology for gossip protocol.

Push Sum Protocol:

We assumed that a node's s/w ratio (Average Estimate) did not fluctuate by more than a factor of 10-10 throughout three consecutive message receive rounds while using the Push-sum approach. When all the nodes have attained convergence, we terminate the program and calculate the total time required to reach convergence in milliseconds. We do this because if we terminate the program after the first node has converged, the average estimate of the remaining nodes in the network will differ from that of the converged node in a large network. We halted the program when all nodes in the network had converged to boost fault tolerance.

Below table shows the convergence time for each type of network with gradual increase in network size:

Node Count	Topology	Time (sec)
100	Line	3.704356
	Full	0.094953
	Random 2D	0.746507
	Imperfect 3D	0.0797
500	Line	16.413078
	Full	2.853087
	Random 2D	126.252277
	Imperfect 3D	2.767447
1000	Line	NIL
	Full	12.718222
	Random 2D	733.01403
	Imperfect 3D	11.420195
1500	Line	NIL
	Full	28.290662
	Random 2D	1405.67189
	Imperfect 3D	27.161208
2000	Line	NIL
	Full	53.267333
	Random 2D	NIL
	Imperfect 3D	48.96682

Fig4: Table indicating network size, topology and convergence time.

The 'NIL' values denote that we could not make the topologies converge and get an output. When there are 2000 nodes in the random2D topology, there may be some nodes who won't get neighbors assigned and therefore the nodes may not get messages so that they converge.

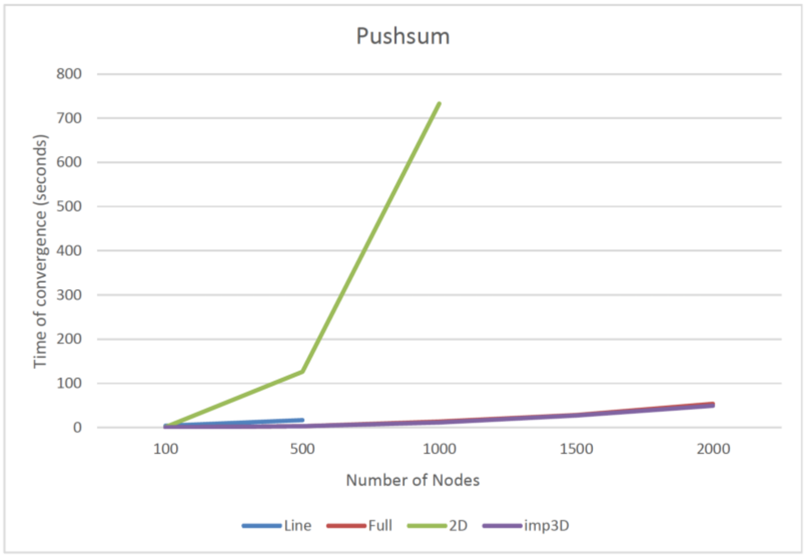


Fig5: Graph indicating network size, topology and convergence time.

The above graph plots the convergence time of different topologies vs the size of the network for Push-sum algorithm.
Inference from the graph:

- The 2D topology has the largest convergence time out of all.
- Imperfect 3D and full has least convergence time for all kind of networks whereas line again lacks behind in most of the cases and fails to achieve convergence even in large amount of time.

Algorithm	**Topology**	**Largest number of nodes attained**
Pushsum	Line	500
	Full	2000
	Random 2D	1500
	Imp3D	2000

Fig6: Table indicating largest network size attained in each topology for pushsum protocol.

Observations:

Convergence time for both algorithms in the case of Full Topology increases gradually as the number of nodes increase. This behavior was completely opposite to what we thought as Full topology is choosing random nodes and sending them the messages but here memory constraints come into picture in the case of large network. Whereas in the case of imperfect 3D which is a perfect blend of randomization with 3D network, the adjacency list of each node just has 8 nodes which does not pose any memory constraints, making imperfect 3D the fastest in the case of large networks.