PROJECT 2: GOSSIP ALGORITHM

HOW TO RUN THE PROGRAM

- Save the file on your system as project2.erl
- The latest version of Erlang/OTP 25.1 is recommended.
- Run the erl command on your system to ensure it is installed correctly.
- Move to the directory where the program is saved.
- In the terminal run the command **c(project2).**
- In the terminal run the command
  project2:start(Number_of_Nodes, Topology, Algorithm).
  ***Note: Execution for each given in the demo video.
  *Samples: project2:start(10,full,gossip).
  project2:start(10,full,push_sum).
  project2:start(10,line,gossip).
  project2:start(10,line,push_sum).
  project2:start(9,twodgrid,gossip). *Perfect Square
  project2:start(9,twodgrid,push_sum).
  project2:start(8,impthreed,gossip). *The Row Size and Plane Size are updated according to intput.
  project2:start(8,impthreed,push_sum).
- The terminal gives output.
  [Updated PID's of Actors in finished list]
  Finished List : [Actors whose PID is finished]
  Map#[Hash Map of Finished ID's with sum for pushsum]
  CodeTime

README REQUIREMENTS AND ANALYSIS

- What is working

Both the required algorithms i.e. Gossip algorithm for information propagation and Push Sum for sum computation across all topologies i.e. Full Network, 2D Grid, line, and Imperfect 3D.

Description

Actor Model

The actors are spawned in the 85th line of the code and after that work is assigned to them by the server.

Topology

Full - All the nodes are connected to each other and the random loop function picks the random node to share the rumor or push sum.

Line - The Neighbor Map keeps the track of the nodes that a node is allowed to share gossip or push sum and picks a random node from them. The neighbors are created using the Up and Down function for the nodes above and below it. Random Function selects the neighbor.

2D Grid - The Neighbor Map keeps the track of the nodes that a node is allowed to share gossip or push sum and picks a random node from them. The neighbors are created using the North, South, East and West functions for the nodes above and below it. Works for perfect squares.

Imperfect 3D Grid - The Neighbor Map keeps the track of the nodes that a node is allowed to share gossip or push sum and picks a random node from them. The neighbors are created using the North, South, East, and West functions for the nodes above and below it along with assigned to create 3D structure. The Plane Size and RowSize according to the input need to be added in lines 47 and 46. For e.g. for 8 nodes Plane size=4 and row =2 because Erlang does not have a cube root function

Gossip Algorithm

The gossip algorithm works parallelly or concurrently. Each node is part of the network in case of full. For line, 2d and imp3d a separate map is maintained that contains the PID's of the neighbor nodes or the nodes that are allowed to communicate with each other. The random loop picks or selects a random node from which the rumor is sent. A Rumour map keeps track of how many times a node has heard the rumor. As soon as the node hears a rumor 10 times it moves to the finished list and is no more an active participant. In the case of line, 2d, and imp3d the neighbor map is also updated for the active nodes that are ready to receive. We terminate the algorithm after convergence and measure the time taken to run the algorithm.

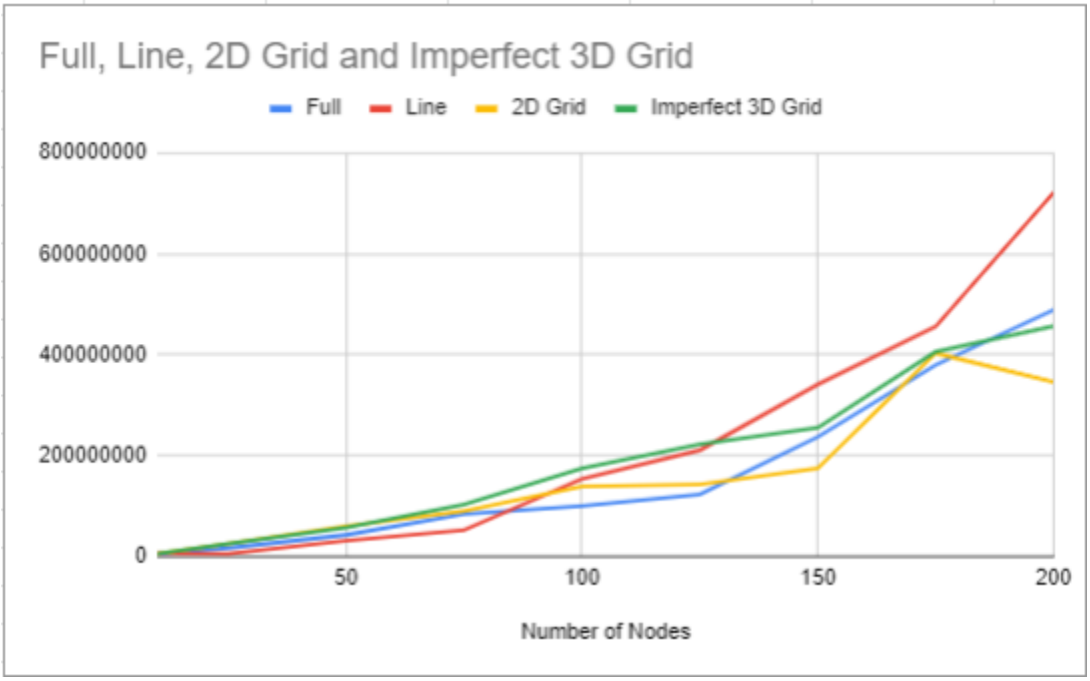
Push Sum

The push algorithm works parallelly or concurrently. Each node is part of the network in case of full. For line, 2d and imp3d a separate map is maintained that contains the PID's of the neighbor nodes or the nodes that are allowed to communicate with each other. The random loop picks or selects a random node from which the rumor is sent. A Weights map keeps track of sum, weight, and s/w. The node converges when the s/w ratio present at each node does not change more than 10 ^ -10 even after three rounds. In the case of line, 2d, and imp3d the neighbor map is also updated for the active nodes that are ready to receive. We terminate the algorithm after convergence and measure the time taken to run the algorithm

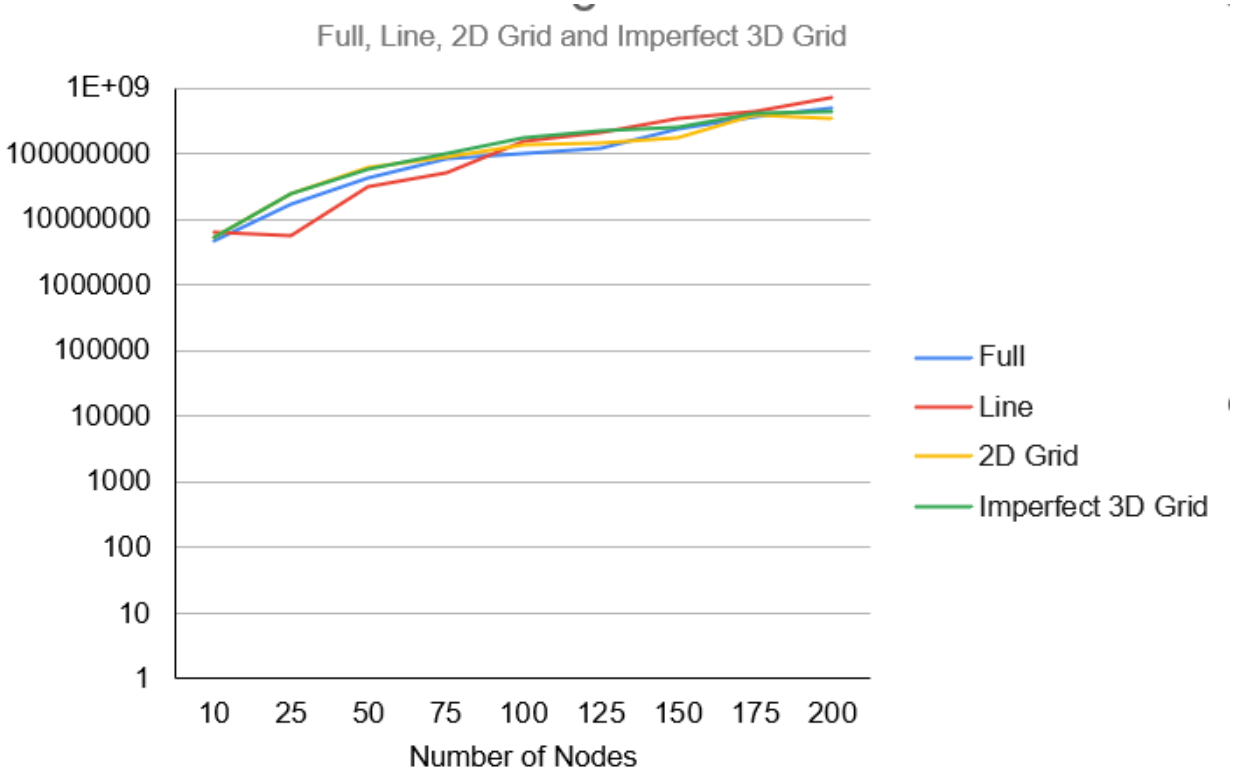
Gossip Algorithm

Time(in ms)

Number of Node	Full	Line	2D Grid	Imperfect 3D Grid
10	4813000	6562000	5187000	5234000
25	16906000	5688000	25250000	25375000
50	42860000	31407000	60562000	57015000
75	84250000	52265000	90047000	102672000
100	100015000	154109000	139141000	174781000
125	122969000	210937000	143281000	222328000
150	236719000	340937000	174781000	254987852
175	379250000	455812000	402875000	406234000
200	488844000	721671000	346484000	456234000

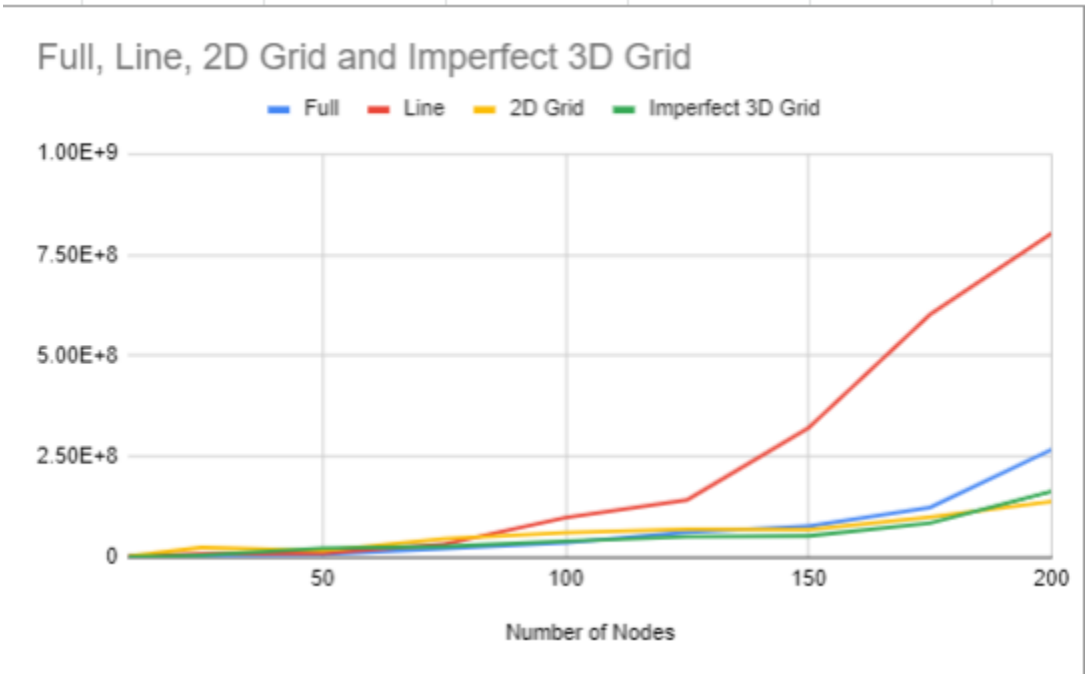


Logarithmic Time

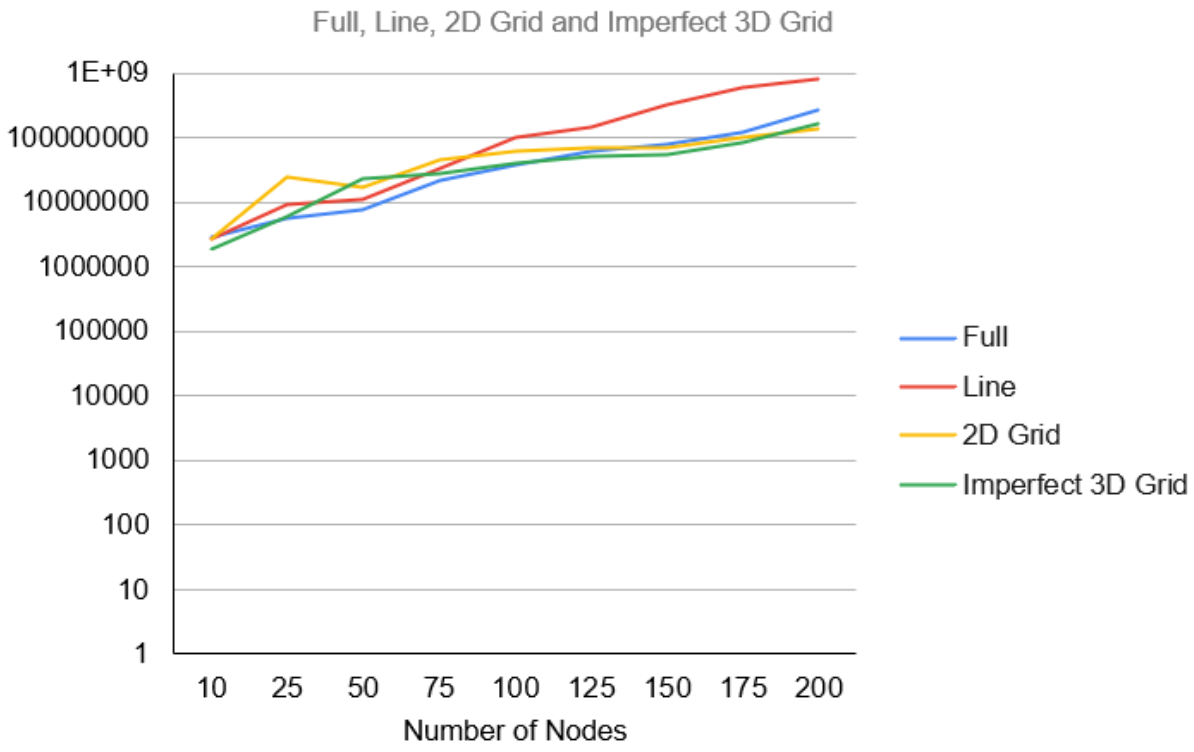


Push Sum Algorithm

Number of Node	Full	Line	2D Grid	Imperfect 3D Grid
10	2932000	2747000	2693000	1903000
25	5561000	9535000	24828000	5850000
50	7837000	11342000	16662000	23234000
75	21499000	32848000	46622000	27529000
100	37065000	99434000	61857000	40922000
125	62876000	142891000	69922000	51416000
150	77779000	320990000	68720000	53467000
175	123776000	602441000	100399000	84844000
200	267515000	802441000	138447000	164210000



Logarithmic Time



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

	Full	**Line**	**2D**	**Imperfect 3D**
Gossip	2500	1500	2000	3000
Push Sum	1500	900	2500	3000

We stopped at these limits because for those combinations the system had reached its limit. Note *The values were not included in the graph as they were affecting the scale due to large runtimes.

Observations

- For Gossip - When there were more nodes, the running time of the line topology was noticeably longer and eventually stopped converging. The fact that each actor in a line topology only transmits rumors to two of its neighbors—on the right and left—could be one explanation for this apparent delay in convergence. When there are fewer nodes, the full topology functions as intended, but when there are more nodes, it runs really slowly, which may be because there are so many neighbors for the rumor to reach.
- For Push Sum - According to the findings of the Gossip implementation, 2D and imperfect 3D performed nearly equally well. The running time for the line topology was substantially shorter than for the other topologies, which is an improvement. When there were fewer nodes, the full topology performed well, but as there were more nodes, it stopped converging. This may have been because there were more neighbors for the rumor to propagate to, as was the case with the Gossip implementation, which had a high number of nodes.
- The highest convergence times for both techniques were found in line topology. The fact that each node has a maximum of two neighbors must be the cause of this. As a result, the rumor spreads through the network slowly.
- In the cases of line, 2D, and Full Topology, convergence times for both algorithms rise exponentially, however in the case of improper 3D, convergence times rise gradually as the number of nodes rises. This behavior was entirely contrary to what we expected because full topology selects random nodes and sends messages to them, but in the event of a big network, memory restrictions enter the picture. The adjacency list of each node in imperfect 3D, which is a perfect combination of randomization and a 3D network, contains just eight nodes and is therefore memory-free, making it the quickest for huge networks.