DOS Project 2 - Gossip Protocol

Rupayan Das - 63992113
Pirouz Naghavi - 30158432

Getting Started

Requirements

- **.NET 5 SDK on both machines**

Running the project

- *dotnet fsi ProjectPart2.fsx <numberOfNodes> <topology> <algorithm>*
- eg: dotnet fsi ProjectPart2.fsx 100 full gossip

Implementation / What is working / Interesting Findings

Topologies :

- full - every actor in the network is a neighbor of each other. An actor can randomly pick any other actor from the network to send a message.
- line - an actor can only have two neighbors (ID + 1, ID - 1). The actor with ID = 1 only has one neighbor, as well as the last actor with ID = numOfNodes.
- 3D - Each actor in the network can have a minimum of three neighbors and a maximum of six depending on its position on the grid.
- Imperfect 3D - Similar to 3D, except each actor has one additional random neighbor selected from the entire network.

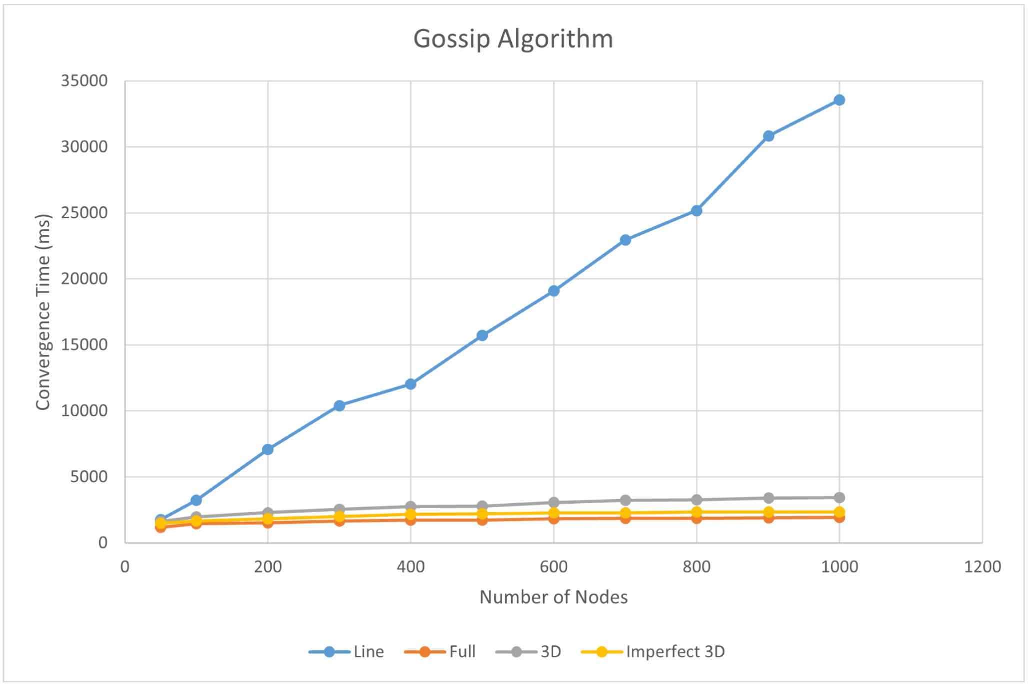
Initial common steps for both algorithms:

The dispatcher sends a message to each worker with the total number of nodes as -100 for all the nodes except the first node that is in charge of starting the communication receiving the correct number of nodes in the network, as well as the ID of each actor when spawned. In addition, the first message from the dispatcher includes the neighbors of each worker, excluding the line topology where the neighbors are selected as the worker ID plus one and minus one if possible, and full topology where the neighbors are clearly the entire network.

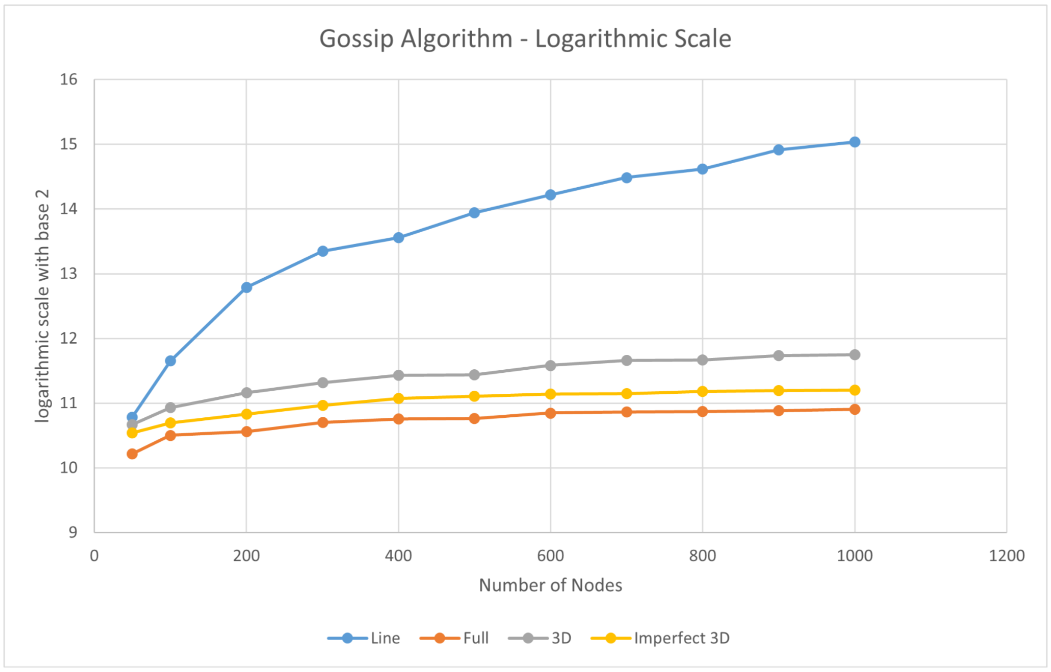
Gossip :

We have implemented a Gossip model where the model converges or terminates when all the actors or nodes present in the system have heard the rumor at least once. So, the dispatcher keeps track of which actors have received a message and increases the count when the actors send “Have heard the rumor” message back to the dispatcher. However, an individual actor will keep on sending messages to other actors until it has itself received the message a total of 10 times. To not fall into the trap of random walk, we make each actor which has received a message wake up after 100 milliseconds and send the rumour to a random neighbor dependent on the topology. This implementation of the gossip algorithm works for all given topologies of full, 3D, imperfect 3D and line.

Normal Scale



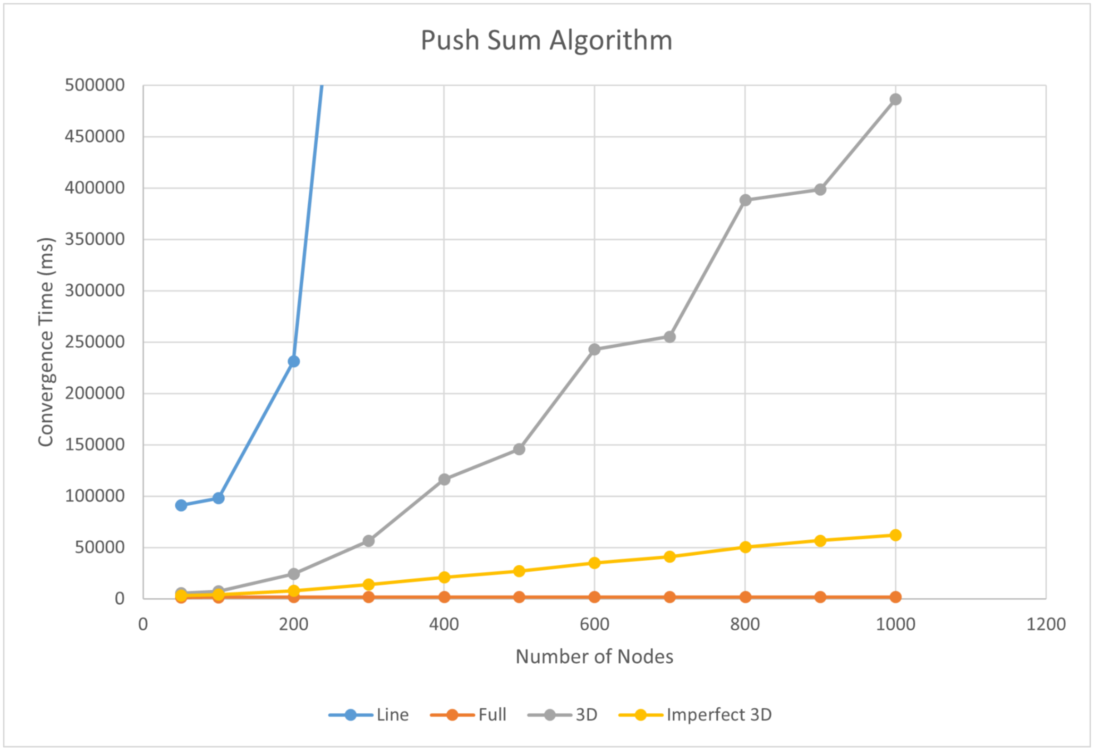
Log Scale with base = 2



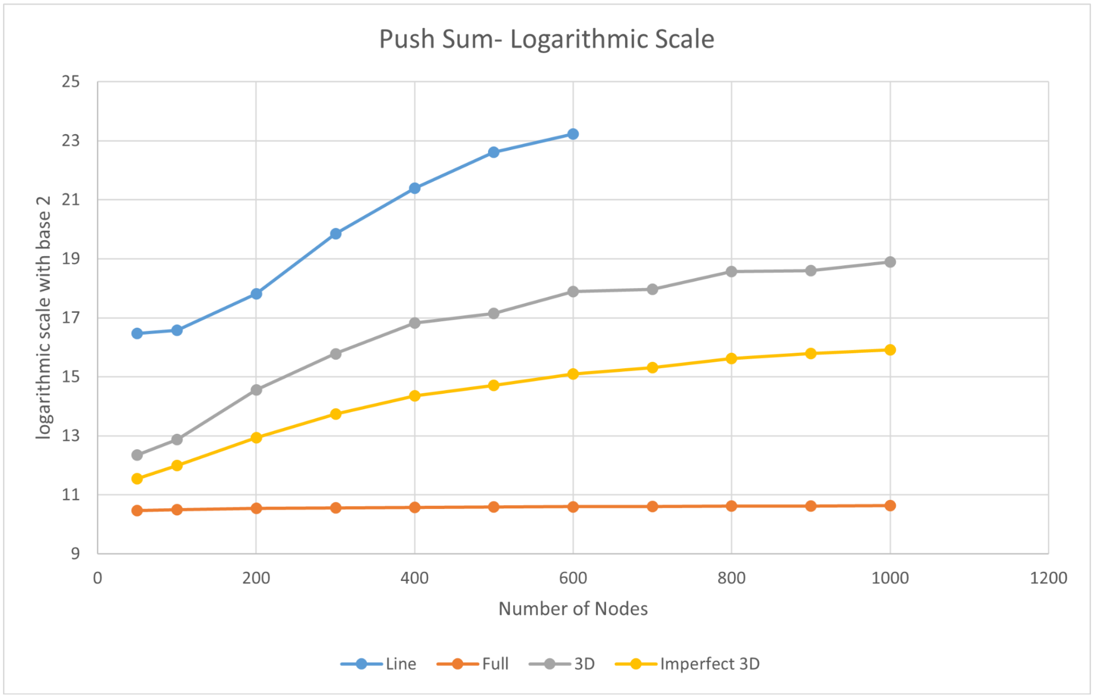
Push Sum:

For Push Sum, convergence occurs or the system terminates only when each actor in the network has its current ratio of sum to weight less than the previous ratio by less than ten to the power of negative ten for three consecutive rounds. To implement this algorithm, the dispatcher first sends each actor a sum (for each actor, the initial sum = ID; this was the design choice because it greatly simplifies troubleshooting of the implementation, but the algorithm will work when each worker selects a random value as the initial sum) and a weight (initialized to 1 for all) along with the number of nodes and the actor ID. Once an actor has it’s ratio converged, it sends a finished message to the dispatcher and stops sending any messages and stops updating its values for sum and weight.

Normal Scale



Log Scale with base = 2



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

Gossip :

- Full - 40,000 (17789 ms)
- Line - 1000 (33552 ms)
- 3D - 20,000 (10109 ms)
- Imperfect 3D - 20,000 (5376 ms)

Push Sum :

- Full - 20,000 (2535048 ms ~ 42 mins)
- Line - 600 (9883489 ms ~ 164 mins)
- 3D - 1000 (1516907 ms~ 25 mins)
- Imperfect 3D - 10000 (1215088 ms ~ 20 mins)

Interesting Finds

- The order of convergence time for both Gossip and Push Sum algorithm is the same
  Full < Imperfect 3D < 3D < Line

- Full topology is the best topology for both algorithms. Having randomness along with the most possible number of neighbors makes it easy to converge in almost all cases. There are almost never any nodes left alone that are not able to converge because of the lack of messages received. Even as the number of nodes are increased, the convergence time increases in what seems to be logarithmic time.

- Line topology is the most inefficient in both algorithms. This is evident from our experimentation and the reason for this is that an actor only has two neighbors to select from, making the time complexity of the algorithm exponential.

- Imperfect 3D is better than 3D because of it’s one extra random neighbor detailing the importance of randomness in such algorithms meant for distributed systems. It allows the message to find new pathways to traverse through.

- So one conclusion we can make is that, the more the number of neighbors available for selection as well as more the amount of randomness, the better is the propagation time of rumor or message through the network.

- For the gossip algorithm, we experimented with the parameter of message received count for each actor. We kept changing it for each topology between the values of 5 and 100. It was noted that the best results usually came in the 10 - 20 range especially when the number of nodes started becoming greater than 10,000.