Project 2: Project Report

Group Members

- Aniket Dash UFID: 7549-9549
- Avik Khamaru UFID: 3699-1467

Topologies

- **Line**: Each and every actor has two neighbors i.e for actor(i) neighbors are actor(i+1) & actor(i-1). Except for the first and last node where there is only one adjacent neighbor.
- **2D**: Actors have four adjacent neighbors where they can spread their rumors except the edges of the grid where they have only two adjacent neighbors.
- **Imperfect 2D**: All actors have four adjacent neighbors just like 2D and in addition to that a random neighbor is chosen from the entire grid. In this topology, every non edge actors have 5 neighbors and the edge actors have 3 neighbors to spread the rumor.
- **Full**: In this topology, each and every actor are connected with each other.

Implementation of Code

- Each and every actor are represented by the name of "NODE". Each NODE has its own state property like the number of times it heard the rumor, an array having reference to its neighbor nodes, etc.
- Each Node(actors) has an onReceive function which is overridden from parent type "Actor" and this function receives the rumors from the queue and processes it sequentially. On the type of message, it executes various functionalities, for example: for gossiping it invokes a "StartGossip" matcher.
- A one time reference of Listener(another type of actor) is created, the work of the listener actor is to keep track of the startTime, endTime, calculating convergence time, etc. There is only one listener actor which keeps track of the starting and ending result accumulated by all the actors.

- In the main thread user-defined values (no of nodes, protocol name, and the network type) are taken. Also, the ActorSystem object (which is the entry point of all the actors that will be created) is created in the main thread. According to the topology that the user-provided various actors (Nodes in our case) are created and neighbors are assigned. It is after this process that the exact algorithm (Gossip and Push) is invoked. The starting node(actor) for the invocation is selected randomly. Also, before the invocation starting time of the algorithm is noted by the listener actor which is also created in the main thread.

Gossip Protocol

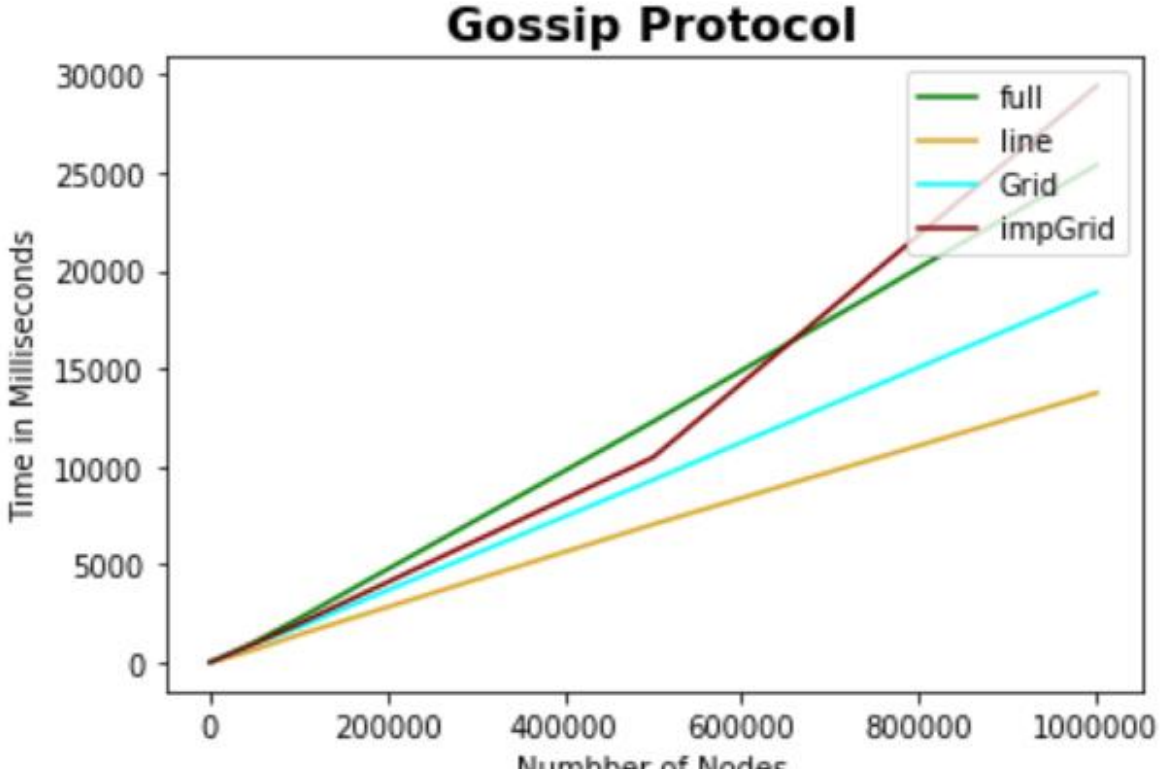
1. For Gossip protocol, at first, a leader (a leader is a kind of actor) is selected randomly in the main thread and it is invoked with a simple message that it needs to propagate (We have hardcoded the message that the actors will propagate, it can also be taken from the user).
2. After the starting point, each node(actor) invokes its neighbors and the same process continues until and unless convergence is reached.
3. Now each time an actor is invoked its number of messages heard count increases by one.
4. If an actor receives 10 messages, then that specific actor is stopped from propagating further. For doing this a global array having information of every node (actor) is defined whenever an actor is getting messages more than 10 times, its neighbors are not directly selected, rather a random integer which is not equal to that actor is selected.
5. Meanwhile, the listener is keeping track of the nodes (actors) which has received exactly 10 messages. When the listener finds that the number of nodes that received 10 messages is equal to the number of the node that the user-provided, the algorithm converges, and the time is reported.

Push-Sum Protocol

1. For push sum protocol each node (actors) has sum, weight, and termRound as its state property. Each node starts with sum as “node’s number” or “i” , weight as “1.0” and termRound as “1”.
2. Here also at first a leader is selected at random and it is invoked and each actor will invoke it's neighbor with a half value and when the value gets less than delta (10 to the power -10) the termRound is getting incremented.
3. When one actor reaches termRound value as 3 that is for 3 consecutive values that the actor calculated were below delta value. We have assumed that since the average reported for once actor for three consecutive rounds is below delta the other actors in the network would have similar values and we converge the network.

Observation and reasoning:

Graph showing gossip protocol variation with respect to the number of nodes and time.

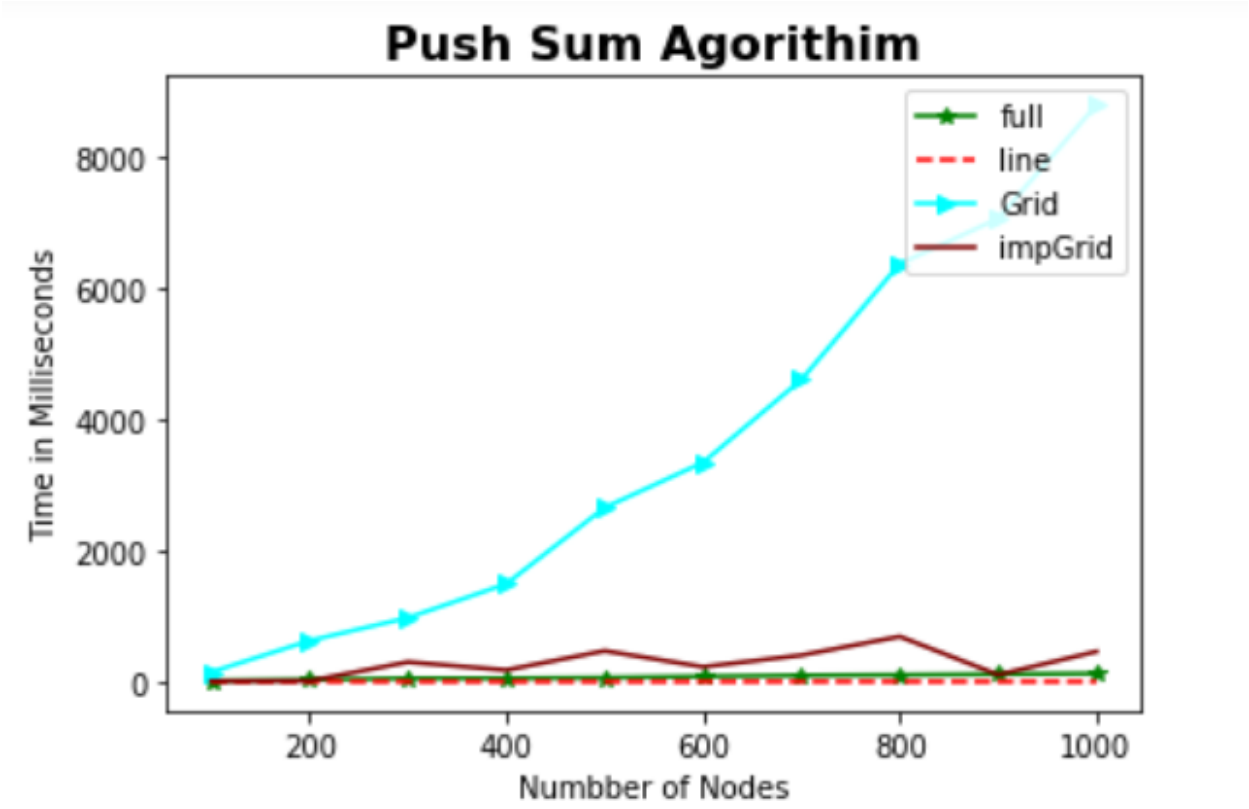


- From our observation, we have seen that Line topology is the best topology because in our implementation if a node is receiving a message 10 times then that node is not selected and another random node that has received a message less than 10 times is invoked. That's how every node is getting the message for sure and finally, the network is converging.

- An imperfect 2D grid takes the most time to converge because of the random node that is selected as its neighbor besides the four adjacent nodes.

- Grid and full network work in average for our implementation.

Graph showing push-sum protocol variation with respect to the number of nodes and time.



- In Push-Sum protocol also Line topology is the best as the number of nodes that are involved in the process is less as only two neighbors are participating that's why in average computation fewer nodes are participating, and it is converging earlier.
- Another thing to note here is that imperfect 2D is taking less time than the 2D topology because of that extra node involved.