Project 2 Report

Group Members
Balaji Balasubramani - 98763981
Divyareddy Surapa Reddy - 13727408

Problem
To calculate the convergence of gossip algorithms: gossip and push sum, when used for either group communication (gossip algorithm) or while computing the aggregation (push sum algorithm) using simulation models.
Given is one amongst the different network topologies (Full, 2D, Imp2D and Line) with a number of nodes as size of each network.
Our goal is to experiment the given network of n nodes using akka actors and find the convergence time.

Interesting Points / Findings:
Gossip:
1. First we implemented the model where we assume that the convergence occurs or program terminates when all the actors were terminated.
   - When all the actors were continuously sending messages till they terminate, we have noticed that, in a few cases when all actors except one or two were terminated, the leftover actors just keep sending messages and never terminate because they don't get messages from any other actor.
   - To handle this, we have implemented a method where the actor keeps track of the number of its neighbours that are not terminated, and once all of its neighbours were terminated, this actor also terminates itself. The reason behind this approach is if none of its neighbours are active, the actor is not going to spread the rumour further and so logically terminates itself.
2. Second model is implemented in a way that it terminates when all of the actors have reached the rumour at least once.
   - To avoid the issue in the above model we have built this approach so that the system terminates once all of them have received the rumour.

PushSum:
For pushsum we assume that convergence occurs when the nodes' ratio has not changed more than 10^-10 in three consecutive rounds and the program terminates when all actors are terminated.

Note: When we implemented this algorithm as mentioned in the given document(which is actor sends messages upon receive), it became sequential and the program halted after one of the nodes converged and it didn't proceed further.

To address the above issue we handled as below but we also have observed that all the actors were not converging when all the actors were sending messages to neighbours and when all of them were terminated except few. In such cases, the system runs into a loop with few actors continuously sending messages without terminating or the actors stopped transmitting and other actors are not receiving any messages.

We handled the above problem similar to the gossip model by tracking the active neighbours. We tried two approaches here with one model sending randomly to neighbours and the other sending messages only to the active neighbours. In the first case, we have noticed a few times that systems did not terminate in full and line topology because of the random functionality and in such situations the actor sends the messages to only the active neighbours to help them converge.

In the above two approaches, the second model showed better performance and also the ratios as this model only sends messages to active neighbours as sending them to already terminated nodes is not going to achieve anything as these messages are simply ignored by the terminated node.

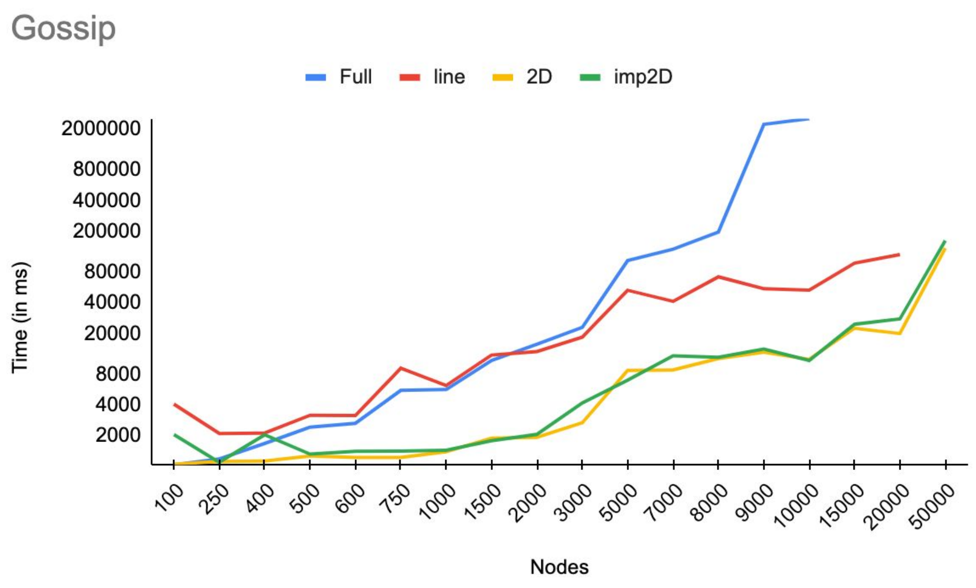
Note: To continuously send the messages, the actor sends a signal to itself after each second everytime using the Akka Dispatchers. This dispatcher is set to send a self message to the actor after one second to keep it continuously sending the messages. Also, in the dispatch wait time one sec other actors were picked up and run thus concurrently running all the actors.

Graphs:

- For each of the algorithms the consolidated graph containing results from 4 network topologies with different numbers of nodes is constructed.
- The graphs are plotted with the number of nodes in X axis vs time taken in Y axis.

Gossip:

Nodes	Full	line	2D	imp2D
100	1046	4111	1060	2068
250	1193	2118	1126	1110
400	1692	2141	1140	2058
500	2448	3191	1276	1334
600	2664	3184	1236	1420
750	5598	9237	1235	1423
1000	5701	6249	1407	1454
1500	10949	12400	1911	1799
2000	15842	13411	1943	2079
3000	23204	18574	2694	4220
5000	104254	53323	8791	7035
7000	134578	41587	8866	12213
8000	197883	72262	11474	11779
9000	2234768	55235	13247	14217
10000	2544066	53486	11261	10972
15000		98441	22680	24852
20000		119383	20164	28013
50000		140342	137848	163428

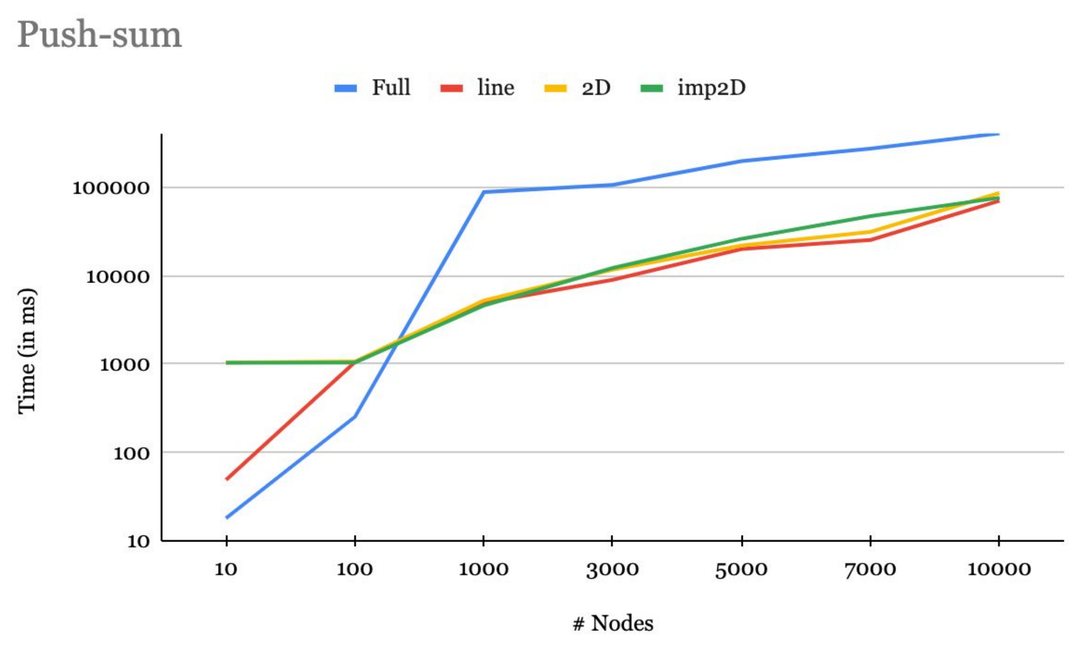


Observations:

The results of the gossip model 1 shows that for the lesser number of nodes, full showed better performance. But as the number of nodes increased, it took a lot of time as each is a neighbour to all other nodes and randomly picking a neighbour from the bigger pool takes time to repeat the neighbours. Few times, when the random function behaves oddly we notice the line program did not terminate properly. 2D and imp2D showed almost the same performance overall.

Push-sum:

Nodes	Full	line	2D	imp2D
10	18	49	1038	1034
100	254	1055	1072	1043
1000	88602	4925	5259	4618
3000	106434	8991	11722	12242
5000	198570	20056	21987	26233
7000	274512	25423	31435	47233
10000	408346	70547	86435	76325



Observations:
The results were again similar to the gossip model but the order has changed slightly. Full topology has again shown better performance when the nodes are lesser. But as the number of nodes increased, it took to the top. Then the rest three were almost similar in terms of their performance when the total number of nodes are more with the overall order Full>imp2D>2D>line.