COP5612 – Fall 2020 Alin Dobra

Project 1 Submission

Akshay Kumar (UF: 4679-9946) | Rajat Rai (UF: 1417-2127)

Problem definition

Gossip type algorithms can be used both for group communication and for aggregate computation. The goal of this project is to determine the convergence of such algorithms through a simulator based on actors written in F#. Since actors in F# are fully asynchronous, the Gossip implemented is called Asynchronous Gossip

The program implements the following two group communication/aggregation algorithms

1. **Asynchronous Gossip:**

   Start: A participant(actor) it 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.

2. **Push Sum:**

   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 receiving, an actor should add the received pair to its corresponding value and sends the message to one of its neighbors chosen randomly.

   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, the sum estimate is $\frac{s}{w}$ where s and w are the current values of an actor.

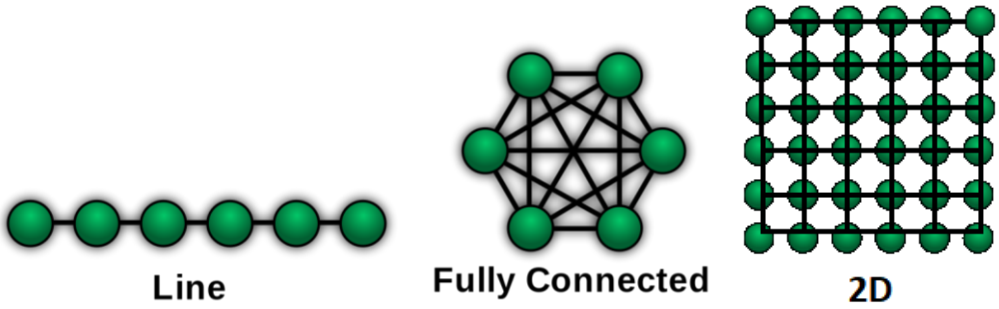
   Termination: If an actor's ratio s w did not change more than 10−10 in 3 consecutive rounds the actor terminates.

Topologies Implemented

for the topologies mentioned below:

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

- **Line:** Actors are arranged in a line. Each actor has only 2 neighbors (one left and one right unless you are the first or last actor).
- **2D Grid:** Actors form a 2D grid. The actors can only talk to their respective immediate neighbors.
- **Imperfect 2D Grid:** Grid arrangement but one random other neighbor is selected from the list of all actors (4+1 neighbors).



How to run the program?

To run the code, navigate to the project root directory using the below command. The root directory is the one where the `project2.fsx` file resides.

```
cd /path/to/proj2/
```

Now, you can run the program using the below command:

```
dotnet fsi --langversion:preview project2.fsx [N] [topology] [algorithm]
```

The program takes a total of three inputs in the order as mentioned.

1. **N:** The total number of nodes in the topology.
2. **Topology:** The topology arrangement of the given nodes.
3. **Algorithm:** The group communication algorithm that the user wants to simulate.

Convergence Issues Observed:

We faced issues with convergence in the case of the line topology. Often, the system terminated even before the complete distribution of the gossip to every node. Similar issues were also observed in the case of 2D, but rarely. In the case of the full network and the imperfect 2D grid, the program responded very well. Please refer to the results sections for a consolidated visualization of the results generated by both the algorithms on various topologies.

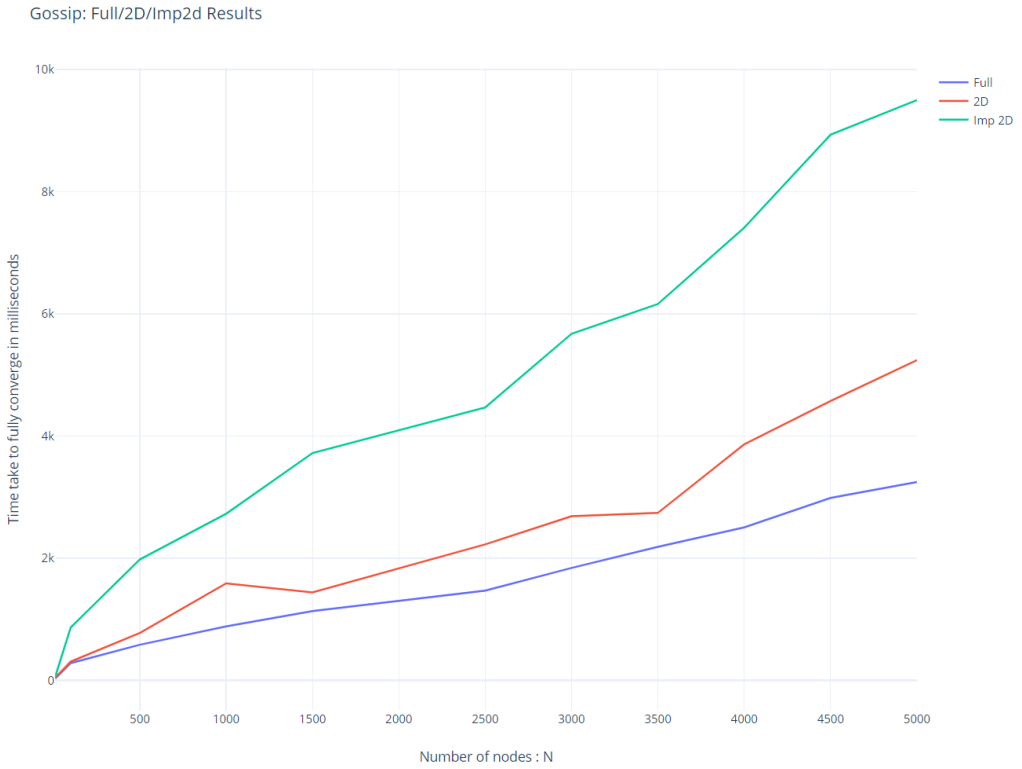
Workaround Implemented:

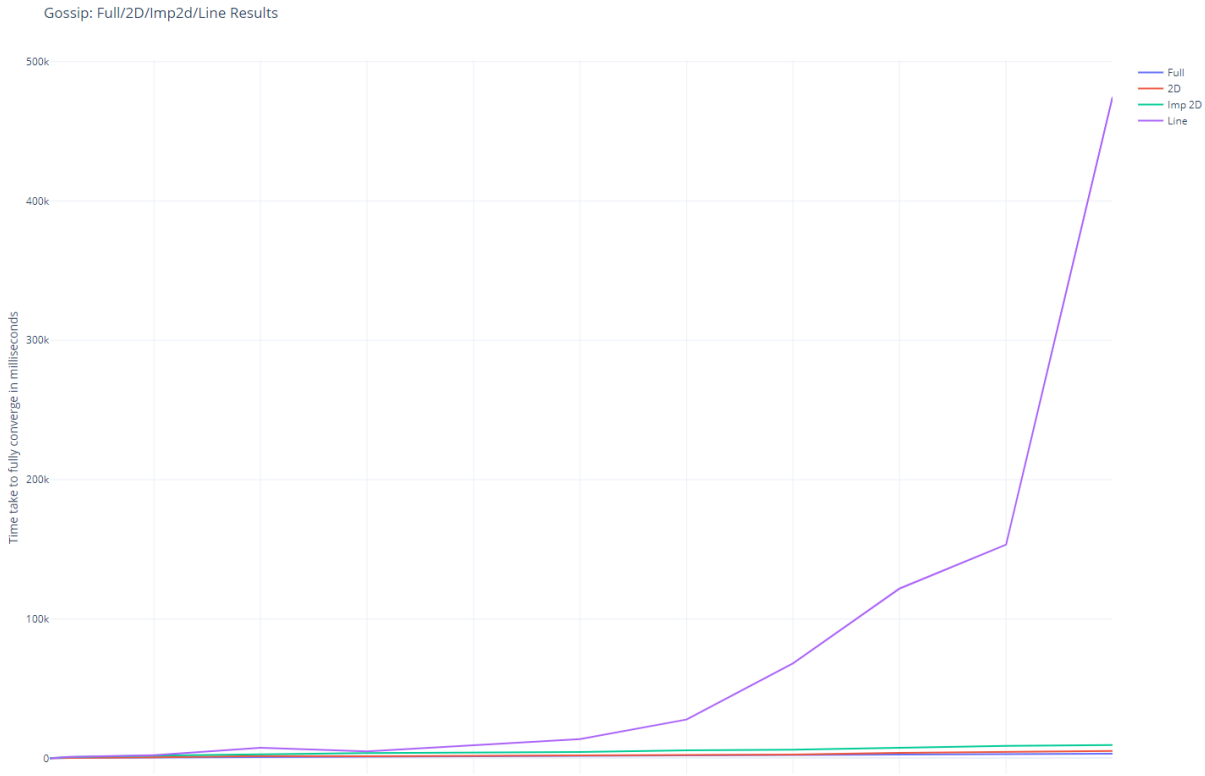
To overcome the issues with convergence, we made a small but considerable change in the way the termination state is reached.

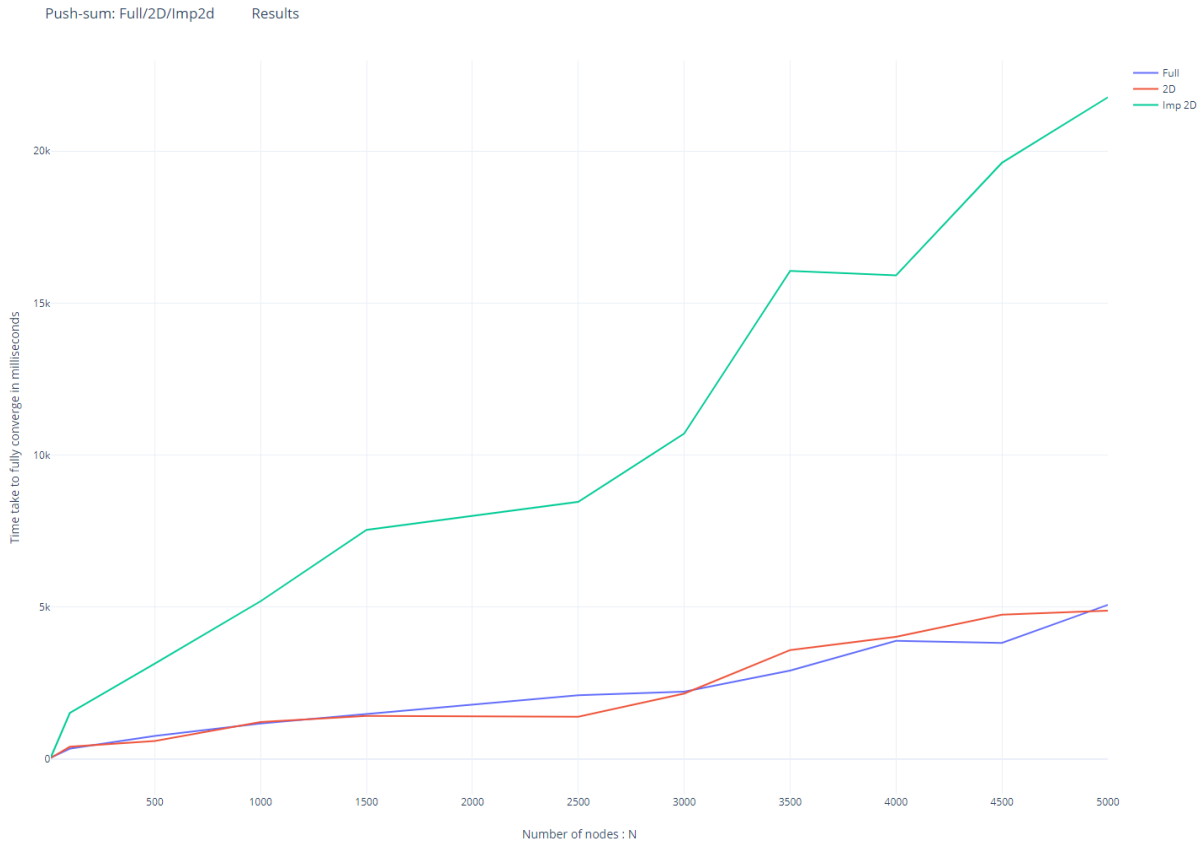
Gossip modified termination condition: Stop when every node has a heard count of at least 10. Guarantees complete distribution of gossip.

Push-sum modified termination condition: Stop when all the nodes record a change in the ratio less than 10^{-10} for 3 times, consecutively.

Results:


Gossip: Full/2D/Imp2d Results
Time take to fully converge in milliseconds
Number of nodes : N
Full
2D
Imp 2D
0
2k
4k
6k
8k
10k
500
1000
1500
2000
2500
3000
3500
4000
4500
5000


Gossip: Full/2D/Imp2d/Line Results
Time take to fully converge in milliseconds
Full
2D
Imp 2D
Line
0
100k
200k
300k
400k
500k



Observations:

After carefully observing and analyzing the results produced by the two algorithms we arrived at the following inferences.

1. The algorithms perform the worst in the case of the line topology. Either convergence is not met or if it is then the algorithm progresses very slowly.
2. Full network and imperfect 2D grid enabled the algorithms to perform their best. The main reason behind this can be the fact that the more connections a node gets, the more possibility there is to spread the gossip to further distances in the network.
3. For large input sizes, we did not observe any unexceptional run-times. However, in the case of 10^5 line and 2D topology, we encountered large run-times for the push-sum algorithm.