COP5615- Distributed Operating System Principles

Fall 2023

Programming Assignment #3

Team 4

Ansh Khatri	Abhinav Aryal	Nandan Dave	Manvi Gupta
5754-5908	3327-1507	5364-2074	9050-7900
ansh.khatri@ufl.edu	abhinavaryal@ufl.edu	nandandave@ufl.edu	manvigupta@ufl.edu

1. Overview:

Gossip algorithms are a type of communication protocol used for disseminating information across a network of nodes. These algorithms are decentralized and rely on nodes gossiping or exchanging information with each other in an ad-hoc manner. The term "gossip" is used because the communication pattern resembles the way rumours or gossip spread in a social network.

2. Implementation:

The primary objective of the project is to develop a simulator in F# by utilising the actor model and the Akka framework. The goal of this simulator is to apply and assess the Push-Sum and Gossip algorithms for aggregate computing and communication over various network topologies. The goals are to create actors who mimic the behaviour of both algorithms, test several network configurations (such as a complete network, a 2D grid, a line, and an incomplete 3D grid), and determine whether both methods converge. The research aims to depict convergence time dependencies on network size for each method and topology combination, as well as analyse how different topologies affect communication speed and convergence. In the end, the study seeks to shed light on how Push-Sum and Gossip algorithms behave and function.

2.1. Akka.NET Actor System

Using actors in Akka.NET, the system is intended to replicate distributed algorithms (Push-Sum and Gossip).

It describes several algorithms (Push-Sum and Gossip) and topologies (full, line, 2D, and 3D) that may be selected according on user input.

2.2. Topologies

Various network topologies are implemented in the code:

Complete/Mesh: Every node is linked to every other node.

2.2.1. Line

A one-dimensional line in which every node is connected to all of its nearby neighbours.

2.2.2. 2D Grid

Each node in the grid is connected to its neighbours (top, bottom, left, and right) and is arranged in a square pattern.

2.2.3. Three-dimensional grid

nodes placed in the shape of a cube, with connections between each node and its six neighbours on the left, right, top, bottom, front, and rear.

2.3. Algorithms Applied

2.3.1. Gossip Algorithm

Actors trade messages (rumours) with a randomly selected neighbour in a looping fashion until a predetermined threshold (a certain number of rounds) is satisfied.

executed as the ActorGos feature.

2.3.2. Push-Sum Method

Actors engage in Push-Sum interactions with neighbours and execute computations in response to incoming messages until a convergence requirement is satisfied.

included as the ActorPS feature.

2.4. Various Actors

2.4.1. Main Actor

Coordinates the start of the selected algorithm and topology. handles termination and convergence checks.

2.4.2. Gossip Actor

Sends and receives messages (rumours) to and from random neighbours in order to mimic the behaviour of nodes in the Gossip algorithm.

2.4.3. Push-Sum Actor

By sharing computed values with neighbours and ensuring convergence, this model mimics the behaviour of nodes in the Push-Sum method.

3. Detailed Description:

3.1. Topology:

3.1.1. Full Network

Every actor in the context of a complete network is recognized as their neighbour. This indicates the capacity of any actor to communicate directly with any other actor because there are direct communication channels, the extensive interconnectedness that this configuration entails speeds up the spread of information.

3.1.2. 2D Grid

A methodical arrangement of actors is seen in the 2D grid setup. Actors are limited to interacting directly with other grid members, as communication is limited to grid neighbours. This topology’s structured structure can affect how quickly information spreads, depending on how well grid communication works.

3.1.3. Line

Actors are arranged in a line by placing them in succession in a linear order. Except for the beginning and terminal actors, every actor keeps exactly two neighbours: one to the left and one to the right. Because actors have a limited number of immediate neighbours, this linear structure may have an impact on the rate of diffusion.

3.1.4. Imperfect 3D Grid

The imperfect variant introduces a twist and looks like a 3D grid. A single neighbour is chosen at random from the whole list of actors, even if they are still arranged in a grid. As a result, every actor has eight neighbours on the grid plus one more neighbour that is selected at random, for a total of eight neighbours plus one. This unpredictability introduces a dynamic component that can influence the rate at which information spreads and may improve resistance to certain network changes.

3.2. Algorithms

3.2.1. Gossip Algorithm:

Within the project, the Gossip algorithm functions as a basic protocol for communication among dispersed actors. It makes it easier for rumours or information to spread over a network architecture with many players. Every actor, standing in for a member of the network, spreads the rumour to neighbours they have chosen at random. The number of times an actor hears and spreads the rumour is represented by the count in this algorithm.

The Gossip algorithm's termination criteria heavily depend on the count. Once an actor has heard the rumour a predetermined number of times, often set at 10 for this project but can be adjusted, it stops transmitting the rumour further.
This count-based termination method aids in determining the algorithm's convergence, which indicates that the data has adequately dispersed over the network.

Comprehending and evaluating this tally inside the project's framework is imperative for evaluating the efficacy of the Gossip algorithm in conveying data throughout diverse network architectures. Assessing the impact of this count on convergence time and information propagation is crucial in determining the algorithm's scalability and performance under various conditions.

3.2.2. Push sum:

The Push-Sum algorithm is a distributed technique designed for computation and information distribution in a network of connected nodes. Distributed systems and wireless sensor networks are two common uses for it. When nodes in the network need to efficiently exchange information or calculate aggregate values collectively, this technique comes in handy.

The fundamental goal of the Push-Sum algorithm is to distribute the computation of sums and averages across the network. Nodes share information with surrounding nodes during each cycle of operation, which is carried out in iterations. The efficient exchange and calculation of values throughout the network is ensured by this iterative approach.

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 neighbour nodes or the nodes that are allowed to communicate with each other. The random loop picks or selects a random node from which the rumour 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 neighbour 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

4. How to run:

Use the command “dotnet run numberOfNodes topologyName AlgorithmName” after opening the terminal in the directory of the file.

```
dotnet run 5 line pushsum
```

Here, 5 is the number of nodes line is the name of the Topology and pushsum is the Name of the algorithm.

For running

Topology Name:

line, full, 2D, imp3D

Algorithm name:

Gossip, pushsum

5. What is working

5.1. Actor Implementation

To mimic a distributed system, the project effectively uses F# actors inside the Akka framework. Actors exchange messages to engage asynchronously as participants in the network.

5.2. Gossip Algorithm

When the gossip algorithm is used, actors disseminate rumours by choosing at random which of their neighbours to contact with information. The actors count how many times they have heard the rumour, cutting the feed after a set number of times (10 by default).

5.3. Push-Sum Algorithm

The use of the Push-Sum method, in which each actor keeps track of two values (s and w), interacts with neighbours, and computes the sum using ratio convergence criteria, is another functional feature.

5.4. Topology Creation

Different network topologies (Line, Imperfect 3D Grid, Full Network, and 2D Grid) are created, and their arrangement determines how the actors interact.

5.5. Main Process & Termination

When all players converge or the rumour spreads sufficiently, the main process ends the system. It also initiates the start of algorithms and records convergence.

5.6. Command-Line Input

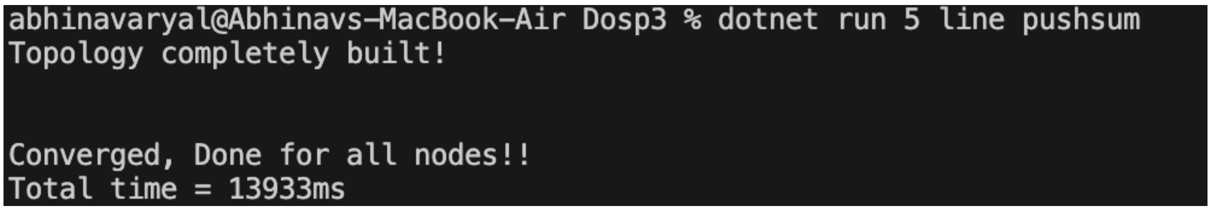
To define the number of actors, the topology to be used, and the method (e.g., dotnet run 20-line gossip), the project allows command-line input.

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

For all the topologies using Gossip Algorithm, we were able to implement up to 10000 nodes,
But while implementing the Line topology took significantly more time when compared to the other topologies.

For the PushSum we were also able to implement up to 10000 nodes but we were only able to implement 5000 nodes for the PushSum algorithm.

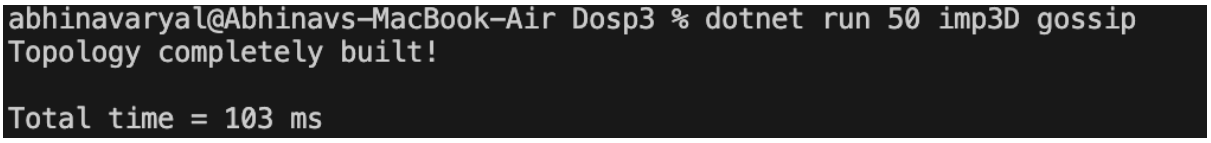
7. Output:



```
abhinavaryal@Abhinavs-MacBook-Air Dosp3 % dotnet run 5 line pushsum
Topology completely built!


Converged, Done for all nodes!!
Total time = 13933ms
```

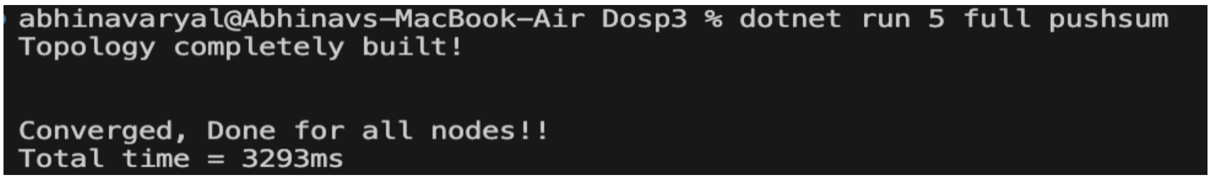
Figure: running output for 5 nodes with line topology and push sum algorithm



```
abhinavaryal@Abhinavs-MacBook-Air Dosp3 % dotnet run 50 imp3D gossip
Topology completely built!

Total time = 103 ms
```

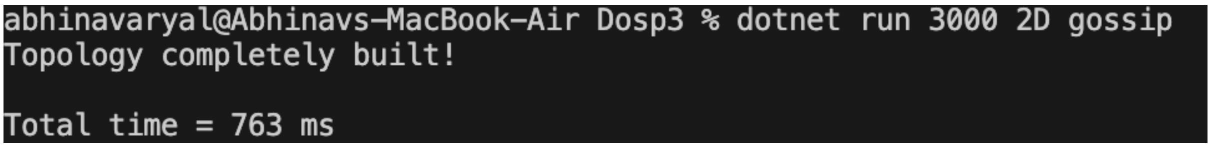
Figure: running output for 50 nodes with improper3D Grid topology and Gossip algorithm



```
abhinavaryal@Abhinavs-MacBook-Air Dosp3 % dotnet run 5 full pushsum
Topology completely built!


Converged, Done for all nodes!!
Total time = 3293ms
```

Figure: running output for 5 nodes with full topology and push sum algorithm



```
abhinavaryal@Abhinavs-MacBook-Air Dosp3 % dotnet run 3000 2D gossip
Topology completely built!

Total time = 763 ms
```

Figure: running output for 3000 nodes with 2D Grid topology and Gossip algorithm

As we went on running we started to collect multiple data values to finally draw the dependency of convergence time as a function of the size of the network.

7.1. Gossip Algorithm

Here the measurement is in ms and all 4 topologies are covered

No. of Nodes	Line topology	2D topology	Full topology	Imp3D
10	182	122	23	102
100	1083	303	102	233
200	2242	422	163	222
500	5662	563	183	223
1000	17012	683	190	224

Table: Gossip Algo values

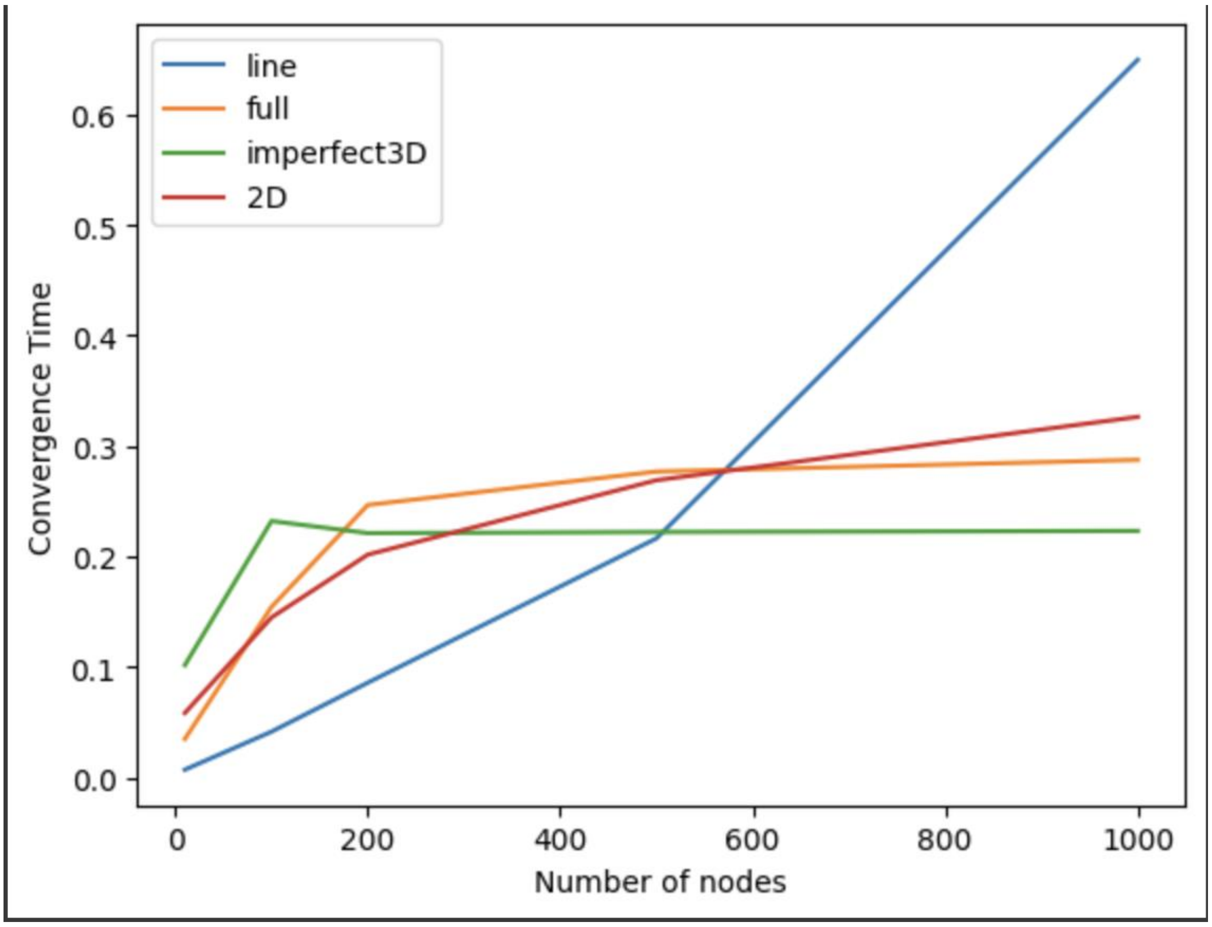


Figure: Gossip Graph

7.2. PushSum Algorithm

Here the measurement is in ms and all 4 topologies are covered

No. of Nodes	Line topology	2D topology	Full topology	Imp3D
5	9120	7573	2772	4032
10	39872	14892	4032	7473
15	103753	14182	4573	6972
20	181382	24672	4752	7932
25	167113	23652	4812	7092
30	279162	34822	4752	11052

Table: PushSum Algo values

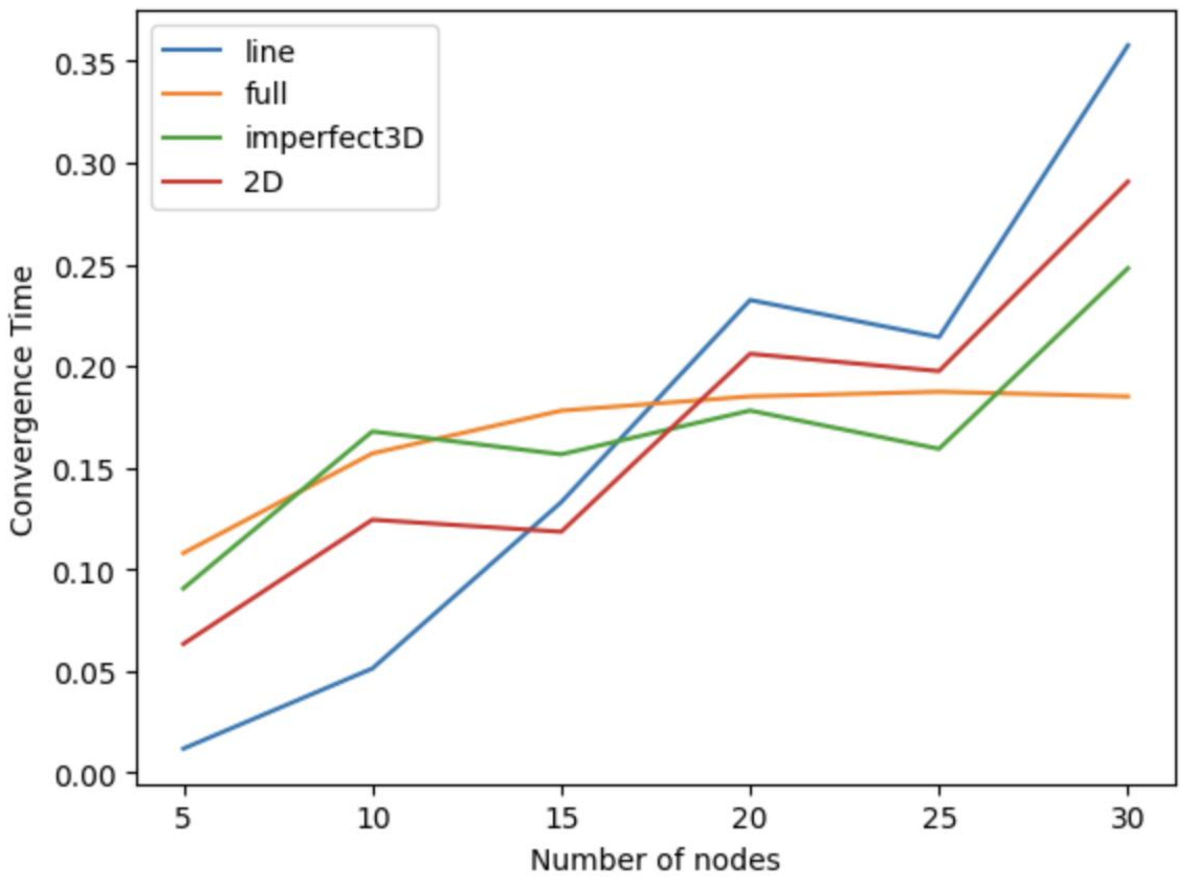


Figure: PushSum Graph

8. Team Contribution

The whole assignment was done by the group together, helping each other with their specific tasks, but here is a basic distribution of work:

Abhinav Aryal was assigned to implement the Gossip Algorithm.
Ansh Khatri was assigned to report-making and working with 2 topologies.
Nandan Dave was assigned to implement the PushSum Algorithm.
Manvi Gupta was assigned to report-making and working with 2 topologies.
The whole group assembled and worked together to avoid any confusion during the work of the project.