Distributed Operating System Principles

(Fall 2020)

Project 2

Venkata Vikramaditya Varma Kunaparaju 1537-8659

Prajan Tikayyolla 6690-9943

Project Description:

We have implemented two algorithms, Gossip Protocol and Push Sum Algorithm, for information propagation in this project. We have implemented these algorithms in 4 different topologies.

How to Run the Project:

Open the FirstIonideProject and in the terminal run the following command.

dotnet fsi --langversion:preview project2.fsx 100 2D gossip

The first Input is the number of Nodes, the second one the topology, and the third is the Algorithm we want to run.

Topologies:

Full: In this topology, all the actors are neighbors of everyone.

Line: In this topology, each actor has two neighbors except for the starting and ending actors, which have only one neighbor.

2D Grid: In 2D Grid topology, the nodes are arranged in an n*n matrix. The nodes present on the edges of the matrix have two neighbors each, while the rest of the nodes has four neighbors.

Imperfect 2D: This is similar to 2D Grid topology with an additional random neighbor for each node.

Code Implementation:

Gossip Protocol:

- The main program randomly selects an actor and sends the message; here, the message is a tuple which consists of the current actor number, total actors, and topology. This actor will start randomly selecting neighbors based on topology and sending messages to them.
- Actors stop sending Gossip once the actors receive the message 10 times.
- During Gossiping, the network may reach a deadlock state meaning all the actors will not converge.
- In 2D Grid topology, a simple case would be when all diagonal elements converge. There is no actor on one side of the diagonal that has received a message at least once.

S	S	C
X	C	X
C	X	X

- The actors marked with X will not receive a message as all their neighbors are in a completed state.
- Deadlock state seen while executing the Gossip protocol for 100 Nodes in 2D topology is as below.

C S C C C C C C C S
C C C C C C C C S C
C C C C C C C S C S
C C C C C C S C S C
C C C C C C C C C S
C C C C C C C C C C
C C S C S C S C C C
C C C C C C C S C C
C C C C C C C C C C
C C C C C C C C C S

- Similarly, even inline topology when an actor completes and all actors on one side have not heard the message at least once will lead to a deadlock state.

S C S C X X

- The actors marked with X will not receive a gossip message as its neighbors are in a completed state, and some are in not started state.
- Due to these deadlock states, the network will not converge at all times. To make the network converge, we used an Anti entropy scheme where the Gossip itself pushes the message and another method that pulls the message from its neighbors.
- The fraction of servers that remain ignorant of the message is given by the formula $s = e^{-\left(\frac{1}{p_{stop}}+1\right)(1-s)}$ where p_{stop} is the probability of actor stopping sending of messages, which in our case is 1. Leading s to be 0.203188.
- Random neighbors are selected on the flow of execution instead of building topology before starting the protocol to save memory by not storing a neighboring list for each actor.

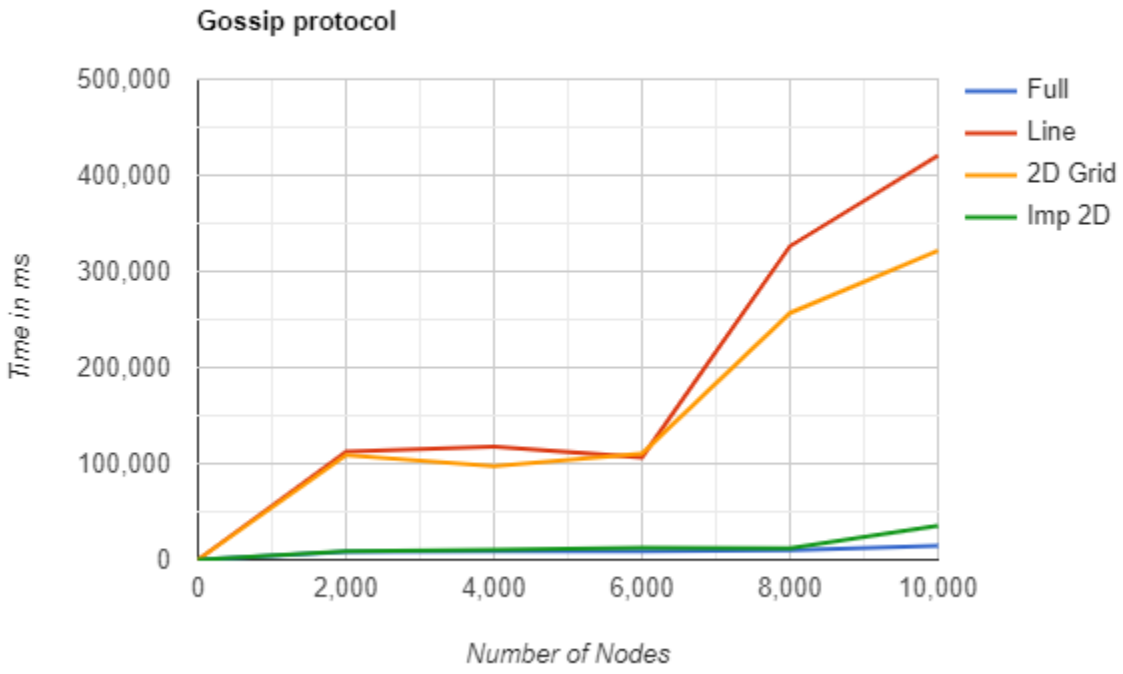
Push Sum Algorithm:

- In Push Sum Algorithm, each node is set with an initial sum and a weight. We start the Algorithm by selecting a random Actor to start the Algorithm.
- In this Algorithm, the receiver adds the sum and weight values it received from the gossip message to its corresponding values. Then it spreads the Gossip by sending half of the result and keeping the rest of the half.

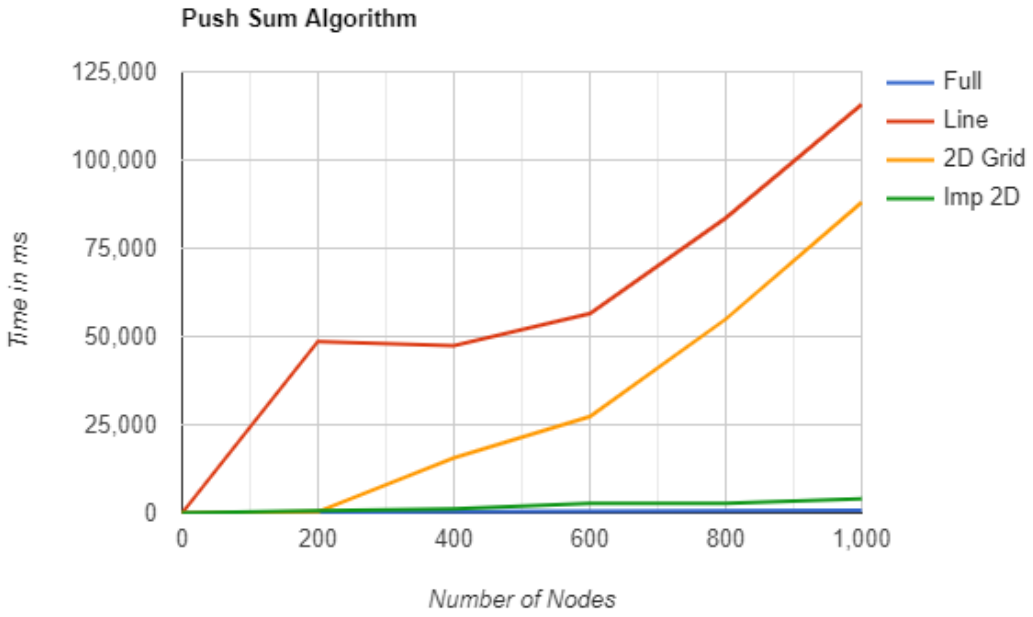
- Here each actor is terminated when the difference in s/w is less than 10^{-10} for three consecutive rounds involving that actor.
- Now for the implementation of Push Sum, the function first creates the Actors with the corresponding topology required and starts the Gossip with a random node.
- The Algorithm reaches convergence when the difference of s/w ratio is less than 10^{-10} for three consecutive rounds of a Node.
- Random neighbors are selected on the flow of execution instead of building topology before starting the protocol to save memory by not storing a neighboring list for each actor.

Results:

Gossip protocol :



Push Sum Algorithm:



Observations:

As we can see from both the Gossip and Push Sum graphs above, Line topology takes a longer time to converge than the other three topologies. The order of convergence for the topologies is as below.

T (Full) < T (Imp 2D) < T(2D) < T(Line)

- Full topology is the fastest, the reason being, all the actors are neighbors to each other. So it is easier to spread the Gossip in this topology.
- In Imperfect 2D topology, in addition to its adjacent neighbors, each node can have a random neighbor selected from the Actors. This makes the topology to spread the Gossip faster than a 2D Grid.
- 2D Grid takes much more time to converge than Full and Imperfect 2D. This is because when all the Actors in any of the diagonal in the Grid are converged, and the actors below the diagonal still haven't started Gossiping, then the Actors below the converged diagonal will be in a deadlock state.
- Line topology takes more time compared to all the others. This is because each node has only two neighbors, and it is tough for the Actors to converge in this condition.