COP5615- Distributed Operating System Principles
Project 2: Gossip

Rishitha Reddy Kallu
UFID: 20015890

Abhijith Chandra Mergu
UFID: 33991217

Aim:

The main aim of this project is to build an actor - model based simulator in Erlang to determine the convergence of gossip and push-sum algorithms.

Pre-requisites:

- IntelliJ IDEA

Language used:

Erlang

Input:

The input to the program are the following arguments:
1. The integer value "numNodes”: number of actors involved
2. topology: one of the following values - "full", "line", "2D" or "improper3d"
3. algorithm: one of the following values - "gossip" or "push-sum"

How to run:

1. Unzip the Project2 folder.
2. Open the file using the IntelliJ editor
3. Open terminal and initialize the erlang shell using the eral command
4. Execute the command "project2: start(<numNodes>, <topology>, <algorithm>)" where the values in angular braces are dynamic.
5. The output on the terminal will display the convergence time for the entered algorithm and topology.

What is working:

Convergence time of Gossip algorithm for all the topologies - Full, Line, 2D, Improper 3D
Convergence time of Push-Sum algorithm for all the topologies - Full, Line, 2D, Improper 3D

Largest Network Managed:

Input	Algorithm	Topology	Maximum Number of nodes managed
1	Gossip	Full	5000
2		Line	10000
3		2D	10000
4		Imperfect 3D	5000
5	Push-sum	Full	10000
6		Line	1000
7		2D	10000
8		Imperfect 3D	15000

Implementation:

The program involves the following major components:

- pMonitor- This process spawns the workers based on the type of algorithm, inserts all the workers and calls dMonitor to start assignment. After the assignment is done, it will spawn and register the timer function. Then, it checks the type of algorithm to call the specific worker with the random neighbor and the message. After all the workers are dead, it ends the timer and shut downs the dMonitor.
- dMonitor- This process calls a function assign neighbors to get the neighbors based on the given topology. After the assignment is done, it will call pMonitor with a random neighbor.
- gossipWorker- This actor will spawn a gossip sender. If the sender is not active initially, we will make it active and send the gossip message. This will be recursively called until the message count is completed, then it kills the sender.
- pushSumWorker- This actor will spawn a push sum sender with new message every time. This will be recursively called until the success count is reached.
- Timer- The timer process calculates and prints the time for convergence.

The workflow of the program is as follows:

1. When start is called, the program spawns and registers both dMonitor and pMonitor.
2. pMonitor will spawn the actors and inserts workers with the PIDs in the list. Then, dMonitor is called to start assignment.
3. dMonitor gets the neighbors based on the topology from assign neighbors function and sends these neighbors list to every actor/worker. After all the workers have been assigned, it calls pMonitor and returns a random node from the neighbor list.
4. Then, pMonitor starts the timer and triggers the first worker as the random node sent with the message.
5. Every actor/worker spawns a sender as it gets the neighbor list. When the first message is received, the sender will start sending the message randomly to it's neighbor. Parallelly, the actor will be receiving the messages until the message count is reached, then it kills the sender. The actor sends a message dead to pMonitor before it dies.
6. The pMonitor will check if the worker count is done, then stops the timer and shut downs the dMonitor.

Gossip:

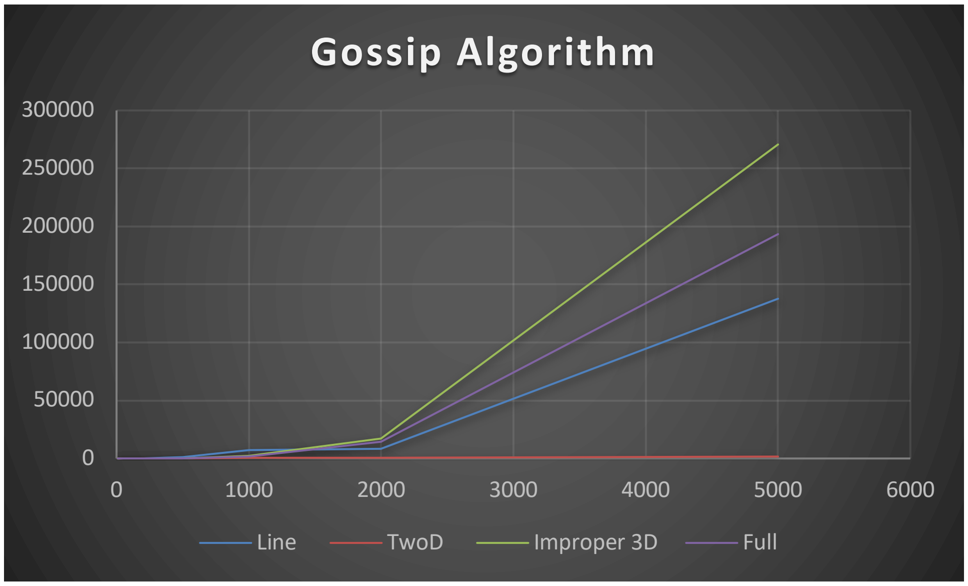
For the Gossip algorithm, we considered the termination point to be when an actor has heard the rumor 100 times. Initially, gossip sender is spawned as soon as the list of neighbors is received. When sender active is false, it starts sending message by calling sendGossip function with the gossip message received. The sendGossip function will randomly select a neighbor to send the message and gets called recursively. The message count keeps decrementing every time an actor hears the rumor and once it reaches 1, the actor stops sending the message and kills the sender.

Push-sum:

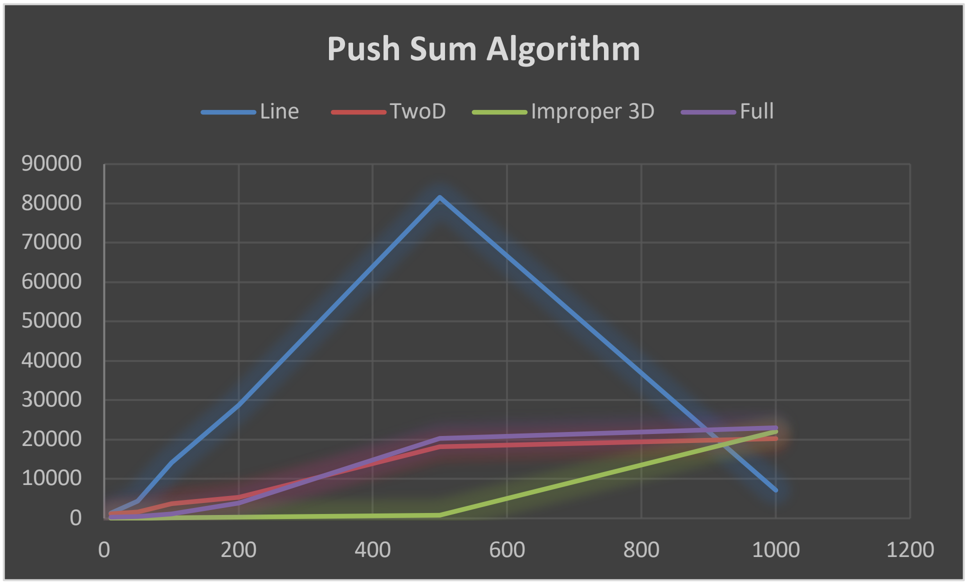
The Push- sum algorithm maintains in two quantities s and w from the main procedure which are the pairs of message sent and received. The actor spawns a sender with pushSumSender function with new message every time. The function will randomly select a neighbor to send the message and gets called recursively until the success count is reached. The ratio s to w is calculated. If the ratio is less than 10^{-10} and success count is 10, then the actor terminates.

Results:

Gossip:

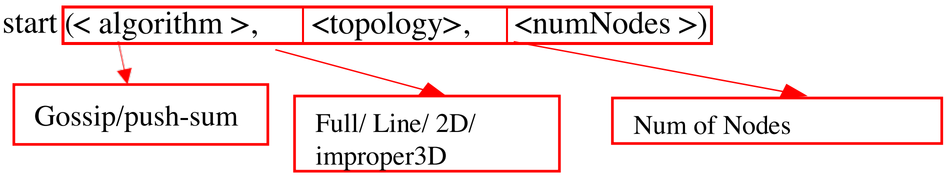


Push Sum:



Output:

Command used – project2: start (< algorithm >, <topology>, <numNodes >)



All times in ms

PushSum

Nodes	Line	TwoD	Improper 3D	Full
10	1331	1156	27	294
50	4419	1567	60	503
100	14155	3768	128	1154
200	28704	5334	231	3821
500	81569	18252	784	20322
1000	7060	20254	22057	23012

Gossip

Nodes	Line	TwoD	Improper 3D	Full
10	5	3	3	2
50	23	15	7	7
100	84	29	35	14
200	239	71	45	46
500	1351	260	259	209
1000	7452	370	2344	1709
2000	8213	436	17404	14392
5000	137728	1660	270715	193411