PROJECT REPORT

Group Information

1. Niraja Ganpule : UFID 17451951
2. Harshal Patil : UFID 55528581

Project Description

The main goal of this project is to implement Gossip and Push-sum algorithms which can be used for group communication as well as aggregate computation. In this project we have implemented these algorithms to run on the following topologies

1. Fully connected network
   In a fully connected network all the nodes are connected to each other, every node knows every other node in the network.

2. Line topology
   In a line topology, all nodes except the nodes at the two ends of the line will have exactly 2 neighbors, that is one on each side. The nodes at the end-points of the line will have only 1 neighbor.

3. Imperfect line
   In this topology, each node has the same neighbors as in the line topology. Additionally, each node will have one more neighbor that will be randomly selected from the total number of nodes.

4. Random 2D grid
   This topology consists of randomly distributed nodes on a grid of size [0-1.0] X [0-1.0] The neighbors of each node will be all points that are at a distance of 0.1 or lesser from the node.

5. 3D Grid
   In this topology, the nodes form a 3D grid. The neighbors of each node are the grid neighbors of that node.

6. Torus
   In this topology, the nodes form a torus. The neighbors of each node are the torus neighbors.

The following algorithms have been implemented in this project

1. Gossip Algorithm for information propagation

   The Gossip algorithm involves the following:

   - Starting: 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

2. Push-sum Algorithm
   The Push-sum algorithm is specified as:

   - State: Each actor A_i maintains two quantities: s and w. Initially, s = i (that is actor number i has value i,) 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 receive, an actor should add received pair to its own corresponding values. Upon receive, each actor selects a random neighbor and sends it a message.
   - 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 being sent to the selected actor, the actor also sends the same message to itself (as stated in the paper)
   - Sum estimate: At any given moment of time, the sum estimate is s/w where s and w are the current values of an actor.
   - Termination: If s/w ratio of any actor did not change more than 10^{-10} in 3 consecutive rounds the actor terminates.

Running the code

Build
The code can be built by opening a terminal in the project directory and typing : mix escipt.build

Run
To run, enter the command escript proj2 <numberofnodes> <topology> <algorithm>

1. numberofnodes can be any natural number
2. topology is one of : full , line, impline ,rand2D,3D,torus
3. algorithm is one of : gossip, push-sum

Implementation

1. Main.ex
   This file is the entry point to the code , containg the main method . It parses the commandline input and based on input it makes a function call to the corresponding function for that algorithm and topology combination

2. Logic.ex
   This file contains the math for every protocol and topology combination.
   Based on the input, the topology is built and the appropriate algorithm is run on this topology.
   Besides the code for detecting how many of the actors reached their converging criteria and for displaying the time required for this convergence in milliseconds is housed in this file.

3. Gserver.ex
   This file contains the business logic for each actor for the push-sum protocol.
   It contains code regarding how the actor is spawned, how it is initialized, what it does when it receives a message,how it checks if it has achieved the convergence criteria and what it does once this happens.

4. Pserver.ex
   This file contains the business logic for each actor for the Gossip protocol.
   It contains code regarding how the actor is spawned, how it is initialized, what it does when it receives a message,how it checks if it has achieved the convergence criteria and what it does once this happens.

Results

Gossip full

# of nodes	Time (ms)
10	710
100	1015
200	1019
500	1117
1000	1123
2000	1332
5000	1340
10000	4326

Gossip line

# of nodes	Time (ms)
10	1012
100	9117
200	17782
500	43565
1000	63853
2000	192071
5000	364347
10000	629252

Gossip random 2D

# of nodes	Time (ms)
10	1817232
100	2312312
200	3132554
500	7623
1000	8032
2000	8237
5000	8890
10000	8900

Gossip 3d

# of nodes	Time (ms)
10	1315
100	911
200	1319
500	1925
1000	2024
2000	2231
5000	2678
10000	2873

Gossip imperfect line

# of nodes	Time (ms)
10	1010
100	1718
200	1819
500	2023
1000	2328
2000	2532
5000	2567
10000	2951

Gossip Torus

# of nodes	Time (ms)
10	609
100	1114
200	1216
500	2025
1000	2533
2000	2887
5000	4267
10000	5387

Push-Sum full

# of nodes	Time (ms)
10	610
100	1050
200	1102
500	1255
1000	1627
2000	1722
5000	3130
10000	3645

Push-sum line

# of nodes	Time (ms)
10	715
100	1275
200	1619
500	3267
1000	5365
2000	83728
5000	1736217
10000	1877899

Push sum Torus

# of nodes	Time (ms)
10	1720
100	1728
200	1932
500	1950
1000	2824
2000	4069
5000	8465
10000	275000

Push-sum 3D

# of nodes	Time (ms)
10	510
100	1426
200	3569
500	4273
1000	7605
2000	31990
5000	35254
10000	34460

Push Sum random 2D

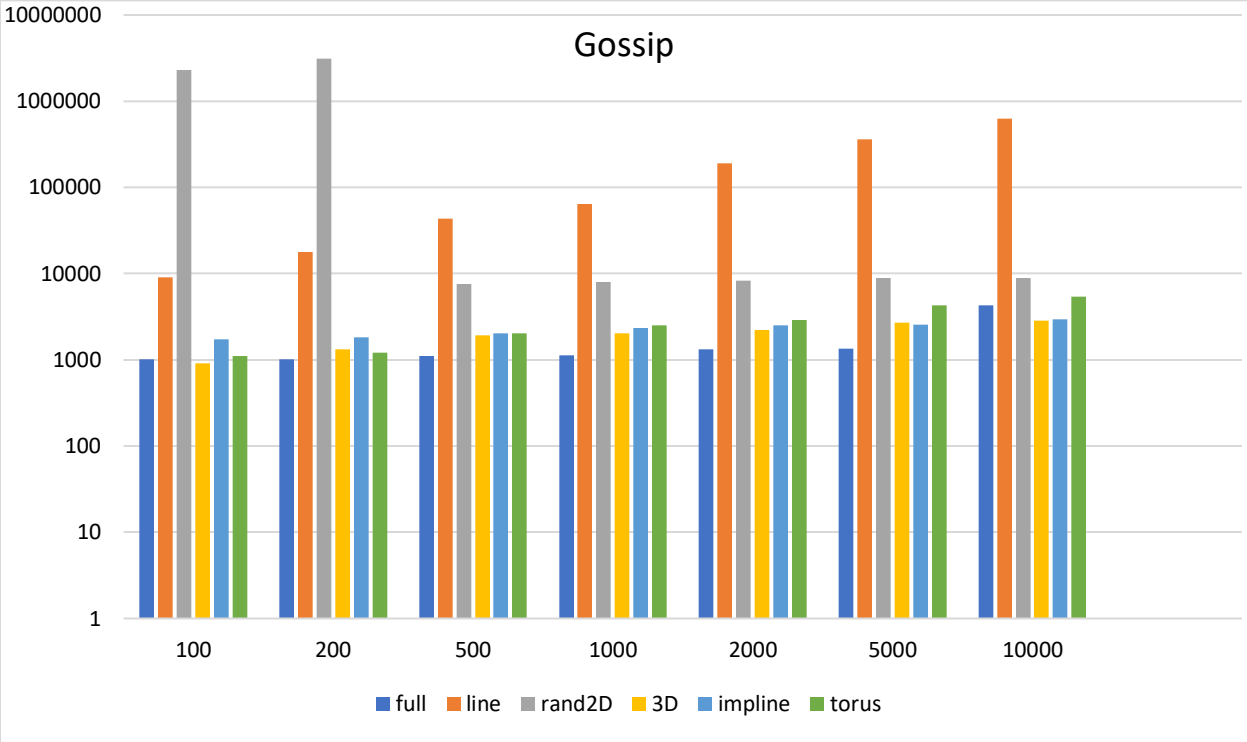
# of nodes	Time (ms)
10	1228722
100	1326733
200	20454
500	32742
1000	88466
2000	117394
5000	158723
10000	167412

Push-sum Imperfect line

# of nodes	Time (ms)
10	914
100	1718
200	3363
500	3707
1000	6138
2000	7418
5000	7845
10000	12000

Graphs and Interesting Observations

Following is a comparison of performance of gossip algorithm over various topologies for a range of nodes

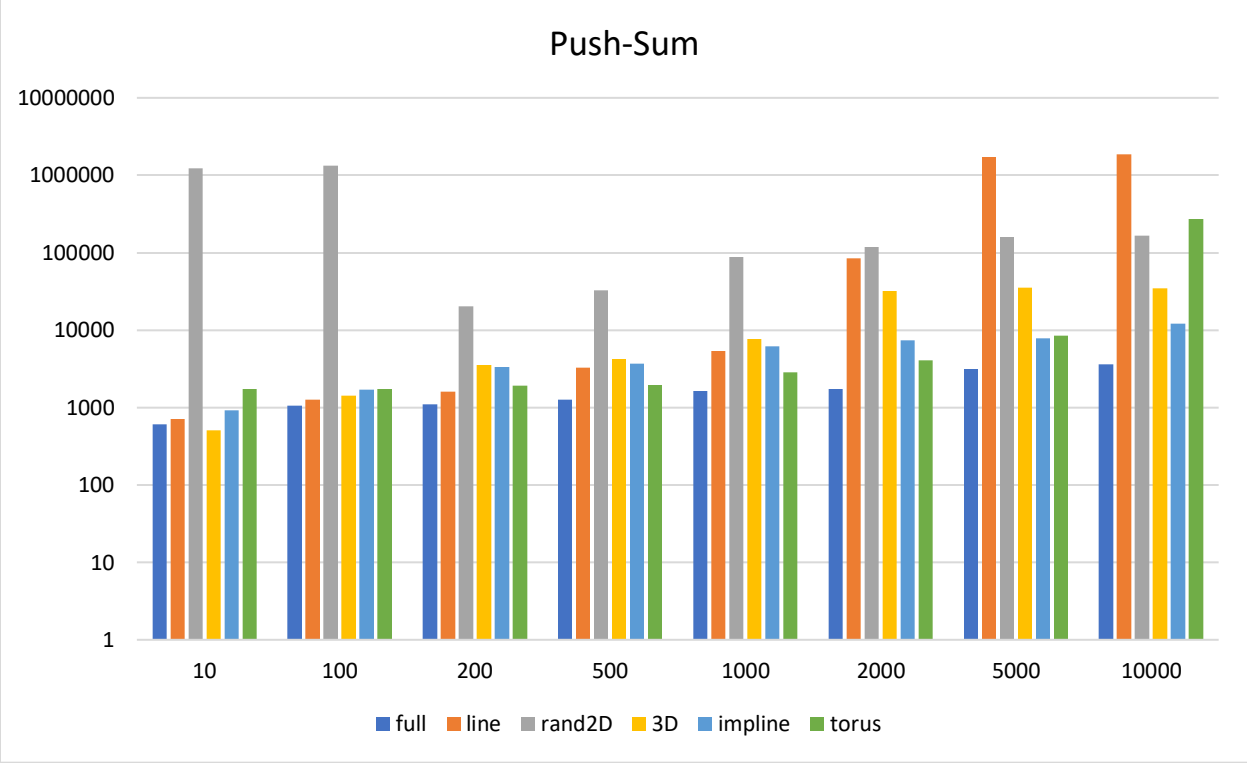


a. Fully-connected network topology will converge the fastest as compared to all others since each node will be the neighbor of all others.
b. The next fastest topology is 3D since each node has 6 neighbours maximum and atleast 3 neighbours , which means probability of selection is better than other topologies that have lesser neighbours per node
c. Performance of imperfect line almost mirrors that of 3D except that it has lesser maximum number of neighbors per node,that is all nodes will have a fixed number of nodes 3. The selection of the third neighbor is random while building topology as well as while perpetuating gossip because of which this topology sometimes takes more time to converge
d. For lesser nodes, Torus topology performs as well as or better than imperfect line topology except at higher number of nodes where imperfect line outperforms the Torus. Torus also has a fixed number of neighbors that is 4 per node because of which probability of selection is good.
e. For a higher number of nodes, the random 2D topology performs better than the line topology since probability of having an elaborate neighbor list is high with

more number of nodes distributed over a small [0-1.0] X [0-1.0] grid, while the nodes in line topology will have a maximum 2 neighbors

f. For a small number of nodes, some nodes in the random 2D topology may not have any neighbors since there are too few nodes distributed in the grid and there may not be any neighbor within a distance of 0.1; however the line topology will always have minimum 1 neighbor (and maximum 2) because of which line topology outperforms random 2D grid.
g. In general, for lesser nodes the random 2D grid may not always converge since there is a high probability that a node has no neighbors

Following is a comparison of performance of Push-Sum algorithm over various topologies for a range of nodes



a. Fully-connected network topology will converge the fastest as compared to all others since each node will be the neighbor of all others.
b. For a small number of nodes, performance of Torus, Imperfect line, Line and 3D grid is comparable. Each node in all these topologies will definitely have atleast 2 neighbors (and more in general) because of which probability of selection is good.
c. For a large number of nodes, Torus generally outperforms Line, Imperfect line and 3D grid as each node will definitely have 4 neighbours while in the others

each node in line will have maximum 2 neighbors, each node in imperfect line will have maximum 3 neighbors and each node in 3D grid may have maximum 6 neighbors but some nodes may have just 3 neighbors, very close to each other because of which fast propogation of message is inhibited concentrating the propogation in a smaller area initially.

d. For a small number of nodes, some nodes in the random 2D topology may not have any neighbors since there are too few nodes distributed in the grid and there may not be any neighbor within a distance of 0.1; because of which the performance of this topology is not up to the mark.
e. For larger number of nodes, in a random 2D topology each node will definitely have a neighbor and hence it may outperform other topologies like the line topology
f. For a large numberof nodes, line topology performs the worst since there are only 2 neighbours ; one the neighbors of a node shutdown, this node is starved and convergence may not be achieved.

HOW TIME WAS MEASURED

The time is measured by putting a time stamp after all the nodes are spawned and neighbour list is calculated for all the nodes.; this is the start time.
Then, the gossip or push-sum algorithm is initiated and we await the convergence of all nodes
After all the nodes have converged, another timestamp is recorded, this is the end time.
The difference in start time and end time is the time taken for convergence.