COP5612 – Fall 2013
Project 2 – Gossip Simulator

Team Members:-

Name: Akhil Jain
UFID: 29806485

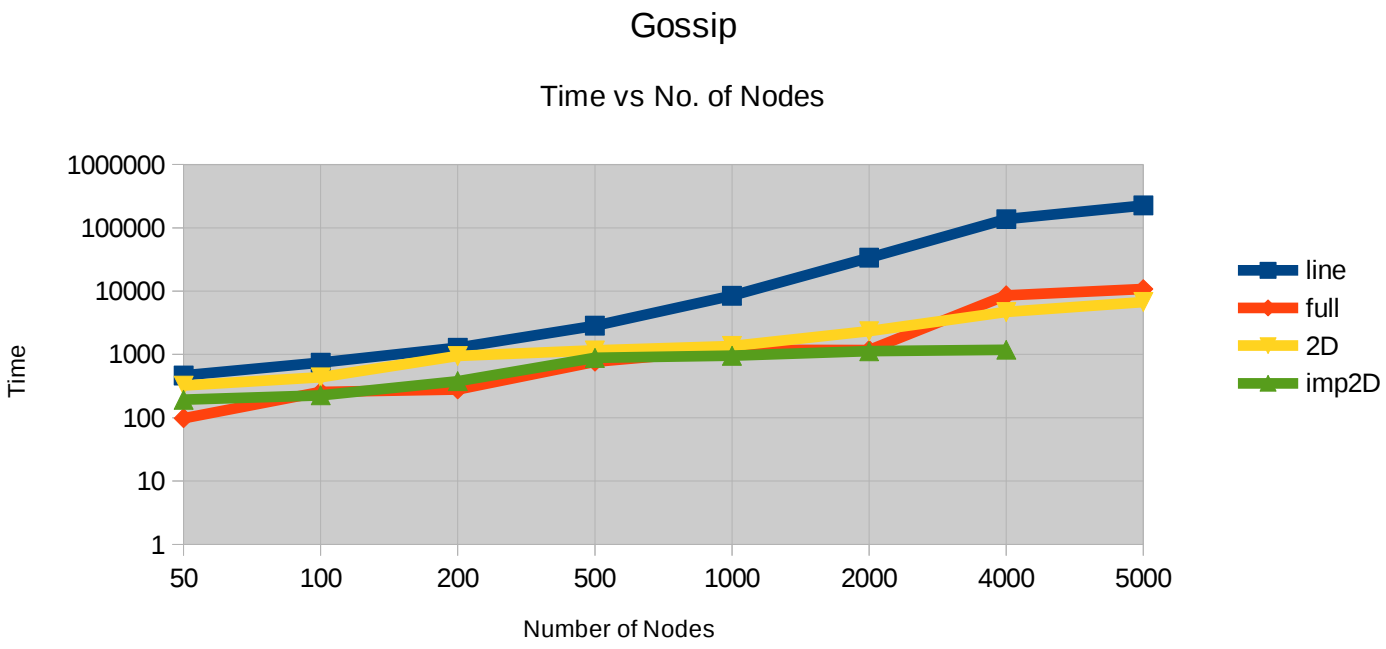
Name: Vinay Kini
UFID: 38161319

The aim of this project was to design a simulator for checking the time it took for messages to propagate in the four given topologies using the corresponding algorithms. The two algorithms used were:
1) Gossip Algorithm for Information Dissemination
2) Push-sum Algorithm
Gossip type algorithms can be used both for group communication and for aggregate computation. Thus the push sum algorithm happens to be built on gossip algorithm. The gossip algorithm involves sending a message to a random neighbor and terminating the system when all the nodes have crossed a certain threshold.

The four given topologies against which we checked against were:
1) Line : All Nodes are placed in a linear fashion such that a node can have at most 2 neighbors.
2) Full: All Nodes are connected to every other Node in the system, essentially forming a mesh.
3) 2D: The 2D topology looks like a matrix in which each node has at most 4 neighbors.
4) Imperfect2D: The imperfect2D is similar to the 2D topology with the difference that every node has one additional random node to select from in its list of neighbors to send. The 'random' factor actually has the advantage of speeding up the algorithm and converging to the right value as we shall see later.

We start the gossip algorithm by selecting the first node and sending it the 'fact' from the master. The first node goes on to propagate the rumor to one of its random neighbors and also sends the rumor to itself. A counter is kept to keep track of the number of times the Node has heard the fact. The threshold is kept at 10, after which the Node shuts down. In order to make sure the rest of the system continues working, if the terminated Node receives a rumor again, it tells the sender to send the rumor to another random neighbor thereby making sure everyone hears the fact. The system shuts down when all the nodes have heard the rumor at least 10 times. This algorithm is run on all the topologies listed above and the results are as follows. The program was run with values of 50,100,200,500,1000,2000,4000 and 5000 as scala project2 500 line gossip.

A graph of the time taken by all the topologies gives a better indication of the performance of the various topologies. To get a better picture, the logarithmic scale of time has been used on the Y axis.

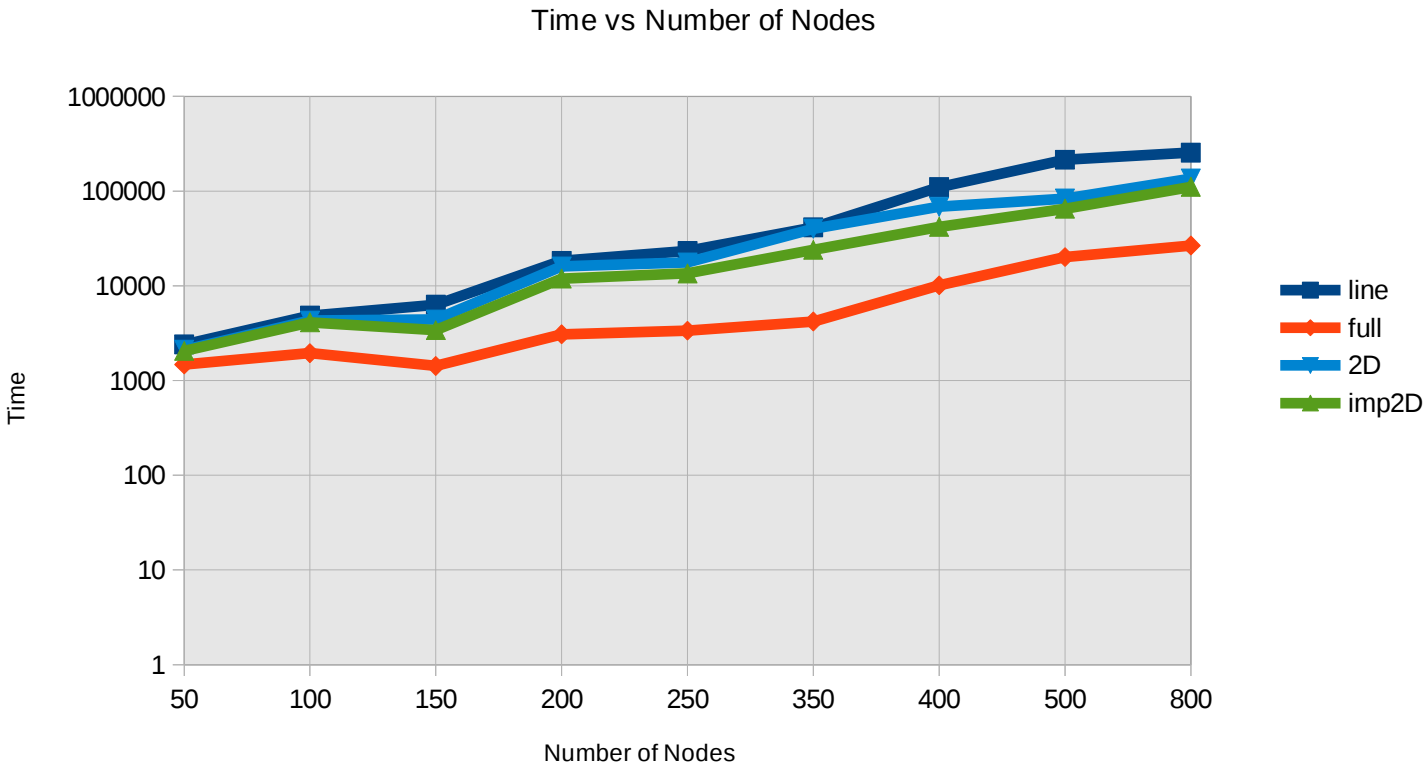


Observations : As evident from the graph above, full topology gives the best performance for less number of nodes but as the number of nodes increase beyond 2000, the time taken by full topology to disseminate the fact is worse than that of 2D topology. The 'random' node (4+1) factor present in Imperfect2D gives it a better performance than a normal 2D topology. Using Line topology gives the worst performance as compared to the other three and keeps getting worse as the number of nodes increases.

Following is the observations from the push sum algorithm along with the corresponding graph.
The program was run with the parameters such as scala project2 200 full push-sum

Observations: From the graph we notice that full topology has the best performance over the other topologies. Also because of the random node factor imperfect 2d has better performance than 2D. Line has the worst performance. The graph has been formed using logarithmic scales on the X axis. Line topology has the problem of not finding convergence at all times. This problem can be alleviated by the use of seeding 25% of the nodes simultaneously. Another problem with line topology is that it takes time for the last few nodes to converge while the rest of the nodes have converged and terminated. This can be solved by using 95% saturation as limit for system to terminate. Thus if 95% of the nodes reach convergence, the system will shutdown conveying that rest 5% haven't converged to the right value. However the above results were recorded for 100% saturations i.e. the system shuts down when all the nodes have converged. Our implementation does incur Garbage collection overhead if it runs out of heap space because of excessive nodes. For higher number of nodes we ran the program with the JVM environment variables set to -Xmx4096m.

Push-Sum


Time vs Number of Nodes
1000000
100000
10000
1000
100
10
1
Time
50
100
150
200
250
350
400
500
800
Number of Nodes
line
full
2D
imp2D