COP5615 Fall 2019

Project 2 Gossip Simulator

October 1, 2019

Group Members

Anirudh Mukundan Raghavan (UFID: 6416-8277)

Aditya Karlekar (UFID: 8888-9598)

Implementation for Gossip and Push-Sum

In Gossip algorithm, the information is sent by the main process to neighboring nodes in a random fashion. The selection of neighbors is dependent upon the underlying network architecture. To simulate the information spread, we made use of multiple GenServers and Supervisor. The Supervisor process, at the start, creates the number of workers depending on the number of nodes specified by the user. The nodes arrangement is based on the topology specified by the user at the time of input. The user also specifies the type of algorithm (Gossip or Push-Sum) at the start of the program.

The Supervisor initiates a random process and propagates the information, in case of Gossip and sum, weights and ratio values, in case of Push-Sum. The GenServer workers hold this information and stop transmitting once the stopping criteria is met. Once the stopping criteria is met, the worker stops transmitting and the sends message to the Supervisor to increment the count of nodes who have heard the message. The Supervisor keeps track of nodes (worker threads) who have heard the message.

The algorithm is assumed to be converged, in case of Gossip, once 90% or more nodes have heard the message. At this point, the Supervisor turns off and time taken for overall operation is calculated. For Push-Sum, the convergence is achieved when the ratio of sum to weight (s/w) ceases to change by not more than 10^-10 for three consecutive calling of same process. Once all the processes are over, the time taken to converged is print on the console.

We start measuring time when the Supervisor creates the first worker node and stop measuring when the algorithm converges. The System.monotonic time was used for measuring time.

The time of convergence in milliseconds is tabulated in Table 1 and Table 2 for Gossip and Push-Sum respectively.

Number of Nodes	Topologies					
	Full	Line	Random 2D	3D Torus	Honeycomb	Random Honeycomb
100	297	875	297	281	360	281
200	578	2828	485	343	345	306
300	1406	2516	1000	375	344	321
400	3641	3688	1750	406	453	406
500	9218	5313	5219	422	407	485
600	24219	8615	9438	469	469	452
1000		10052		672	656	623
1500				1093	953	955

Table 1: Table illustrating time in millisecond for nodes and various topologies for Gossip algorithm.

Number of Nodes	Topologies					
	Full	Line	Random 2D	3D Torus	Honeycomb	Random Honeycomb
100	281	891	328	312	352	345
200	2438	1437	375	328	359	353
300	6187	3968	359	352	365	325
400	3641	4688	438	396	401	396
500	13797	5625	491	425	425	422
600	37425	7157	504	453	468	465
1000		8750	656	527	625	610
1500			969	938	945	922

Table 2. Table illustrating time in milliseconds for nodes and various topologies for Push-Sum algorithm.

Observations:

We plotted the graph for time taken by both the algorithms for multiple topologies for varying number of nodes. Figure 1 shows the actual time in milliseconds taken by Gossip algorithm. Figure 2 plots the time on a logarithmic scale for Gossip algorithm. Figure 3 shows the actual time in milliseconds taken by Push-Sum algorithm. Figure 4 plots the same time on logarithmic scale for Push-Sum algorithm. The following observations were made from the graphs.

1. The **Full network** maximum time for convergence for both Gossip and Push-Sum. This is perhaps due to the overheads in maintaining the adjacency list for all nodes as every node is connected to every other node in the network.
2. The **Line** network has the second highest time for convergence for both Gossip and Push-Sum. This can be attributed to the underlying architecture if the line network. As each node is connected to just two other neighboring nodes, the probability of message and values reaching to every node on the network drops with increase in the size of the network.
3. The **Random 2D** algorithm works well for small number of nodes, though it fares quite bad as the network size increases. This can be attributed to the fact that the random 2d network has multiple nodes with no neighbors.
4. The **3D Torus** converges quite fast for both Gossip and Push-Sum algorithms and scales well with the increase in size of the network. This is perhaps due to the underlying architecture which ensures that each node has equal number of neighbors and arranged in a fashion that allows every node to be connected to every other node is some way.
5. One interesting observation is the time taken by **Honeycomb** and **Random Honeycomb** topologies. From the data it is quite evident that adding a random extra node to the honeycomb topology reduces the convergence time and does not add much overhead to the overall process.

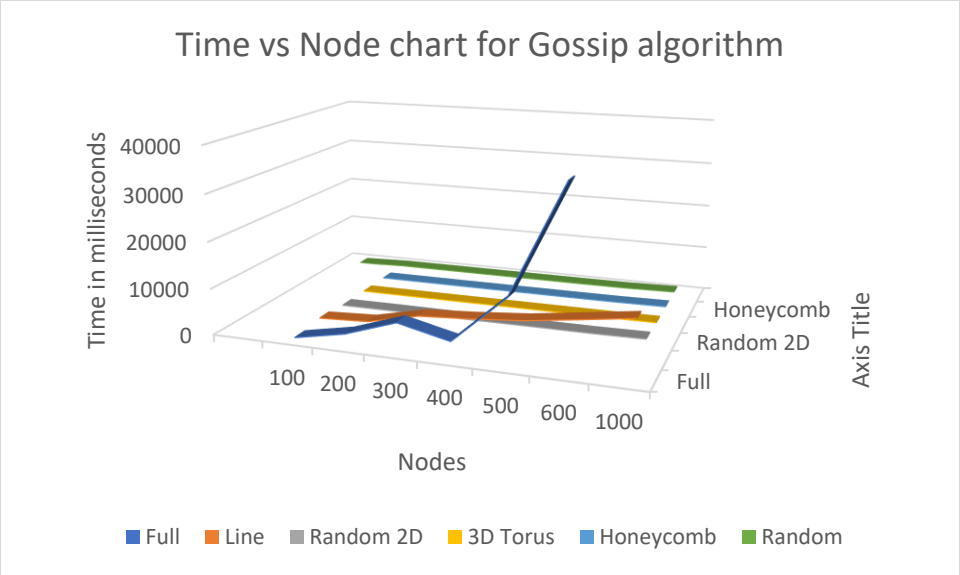

Figure 1. Time vs Nodes chart Gossip Algorithm

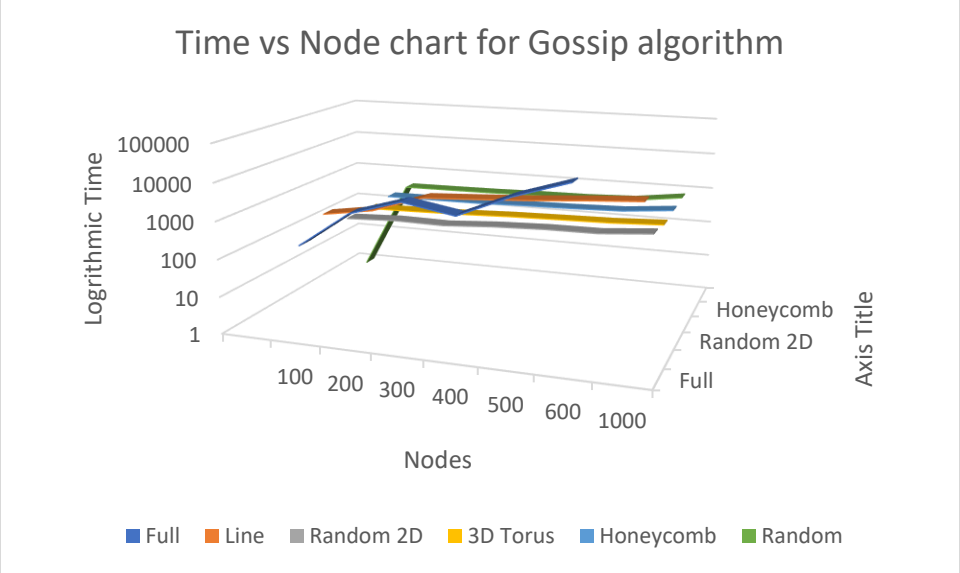

Figure 2. Logarithmic Time vs Nodes graph for Gossip algorithms

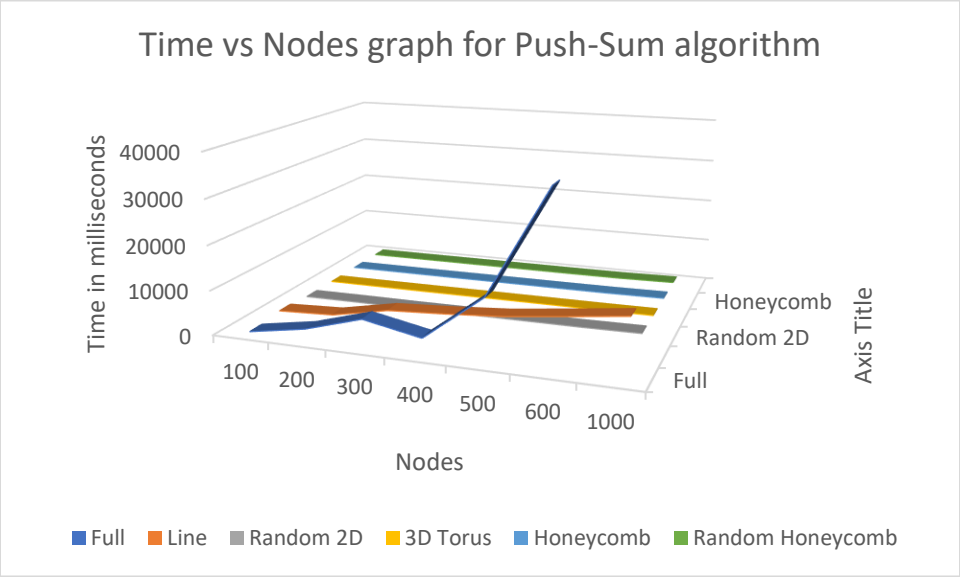

Figure 3. Time vs Nodes graph for Push-Sum algorithm

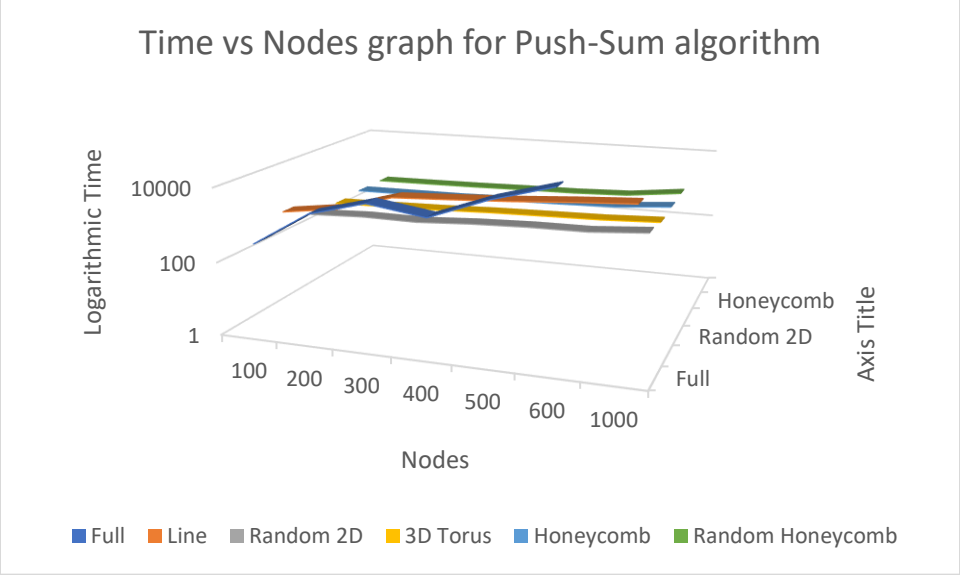

Figure 4. Logarithmic Time vs Nodes graph for Push-Sum algorithm.