Project Report

Implementation of Gossip Protocol

1.1 Team Members

1. Ashutosh Garg
UFID: 0489-9999
2. Sakshi Dubey
UFID: 4813-1141

1.2 Implementation Details

We have worked on Gossip Algorithm in this project with the assumption that the convergence of Gossip Algorithm will occur when all the nodes in the network have heard the rumors at least once. In our implementation, if a node hears the rumors initially for the first time, it will treat itself as converged. However, it will still keep on sending & receiving the rumors. The terminating condition of our project is the occurrence of 10th rumor. The program will print the convergence time and terminates when once the convergence is achieved. We have used the Genserver module of elixir for implementation of actors. Initially, first n nodes are created in general case & for 2D & 3D we generate m nodes where m is the nearest perfect square of n or perfect cube. Furthermore, we calculate the neighbor of every single node according to topology and send each node its neighbor list. We start the timer and send the first message to random node in the network. Also, a process monitor is implemented to make sure how many processes have converged. In this case, if the convergence is achieved the monitor will exit & then we can calculate the total time for convergence.

1.3 Gossip Protocol Implementations

As soon as a message is received by the node, it will increment its counter and furthermore, it will activate the periodic send function. Periodic send will carry on sending messages to its randomly selected neighbor. We have set termination criteria as the hearing of message for 10 times. Convergence Criteria has been set when at least 30% of the nodes have heard the message.

1.4 Push-Sum Protocol Implementation

If a message is received by the node, it will add the value of n and w received in the message to its own s and w values. Furthermore, it will send the half of this n and w value to any of the

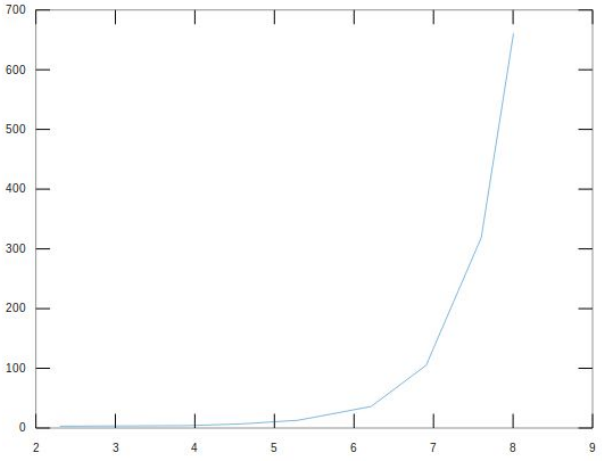
randomly selected neighbor and will keep half of the value for itself. This process will carry on till the difference between the old and updated ratio of n and w stabilizes for 30% of the total nodes. This is done because we are implementing it on a large network, thus if the program is terminated after the convergence of very first node then the average estimate of the other nodes in the network will differ from that of the converged node. Hence, we felt it's a wise choice to terminate the program after the convergence of 30% of nodes in the network. This way we can ensure the efficiency of the program.

1.5 Graphs

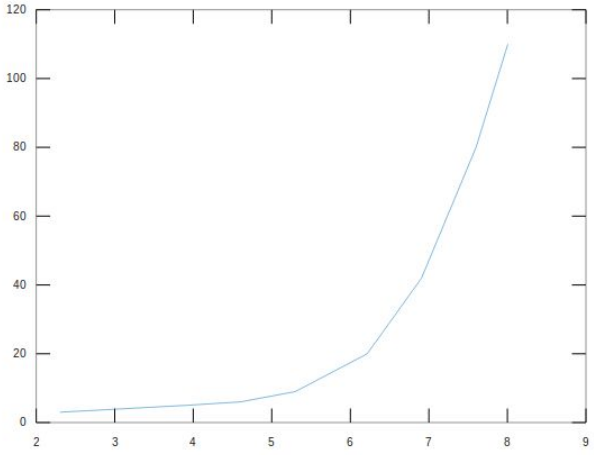
1.5(a) - Graph between convergence time of different topologies vs the size of the network for Gossip protocol

x-axis : Logarithm of number of nodes
y-axis : time in milliseconds

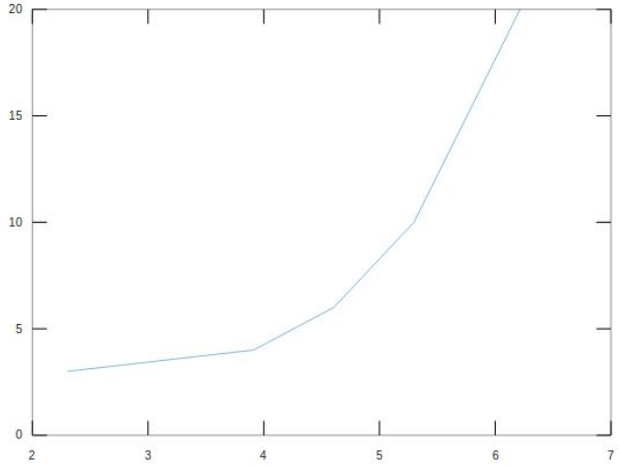
Full:



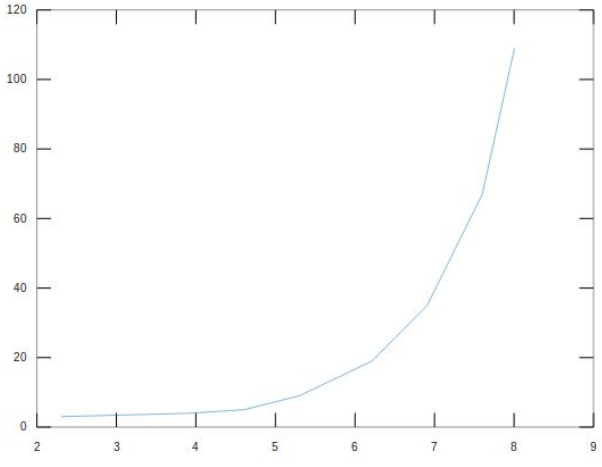
Rand2D:



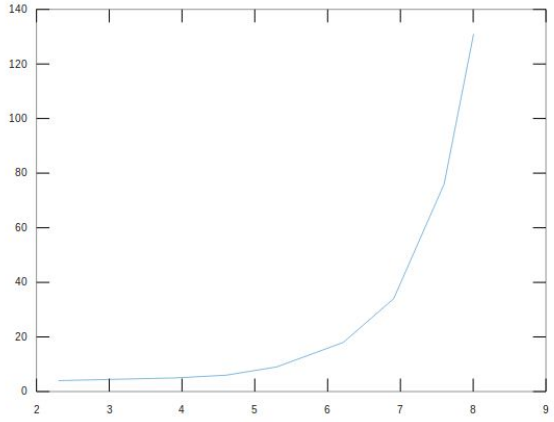
Line:



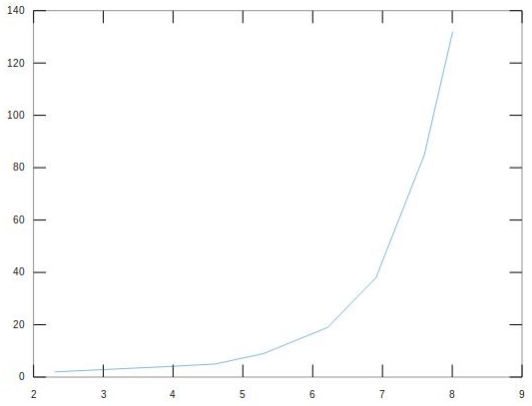
Torus:



3D:



Imp_line:

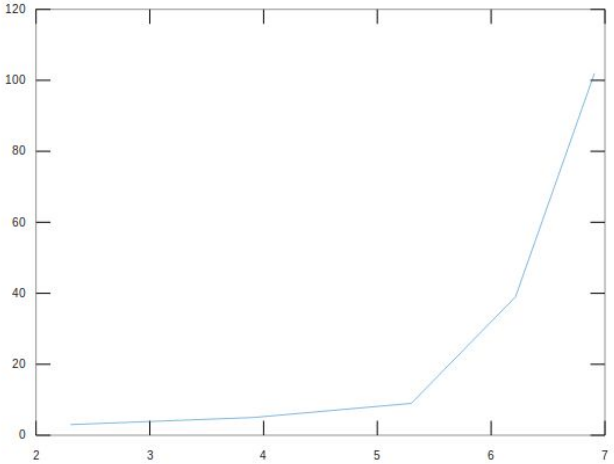


1.5(b) - Graph between the convergence time of different topologies vs the size of the network for Push-sum algorithm

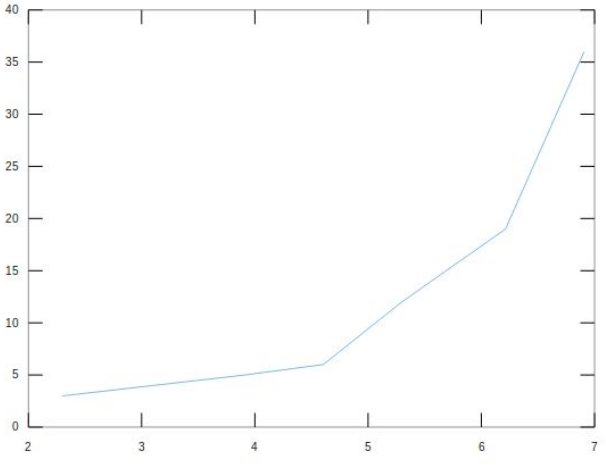
x-axis : Logarithm of number of nodes

y-axis : time in milliseconds

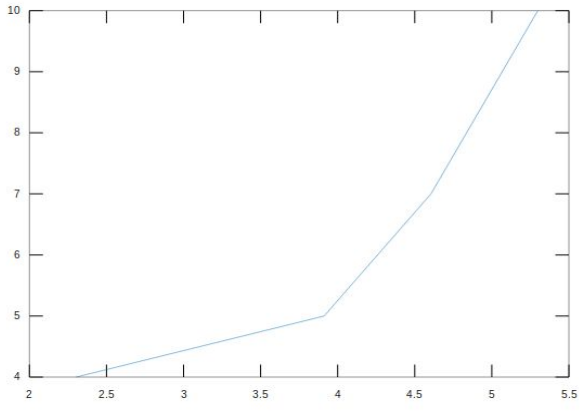
Full:



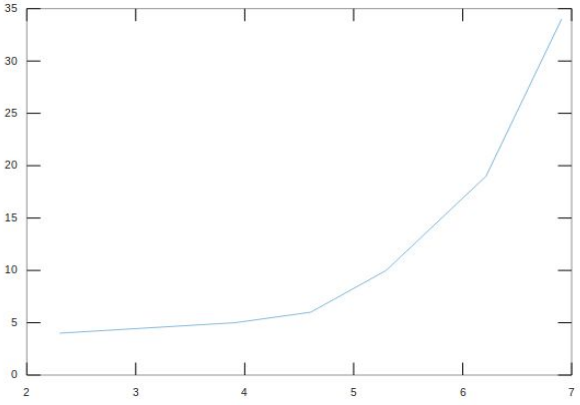
Rand2D



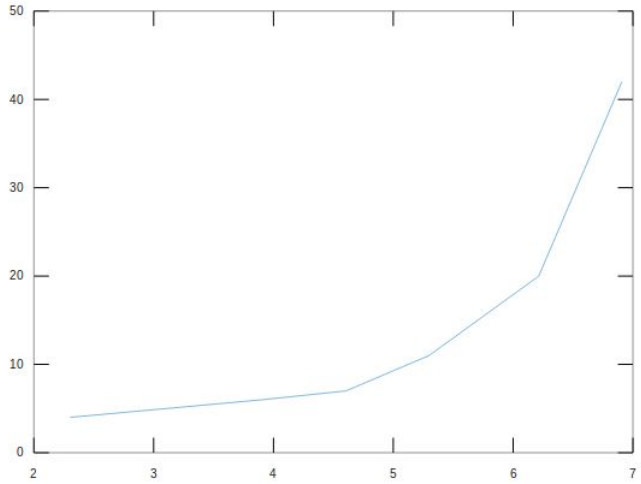
Line:



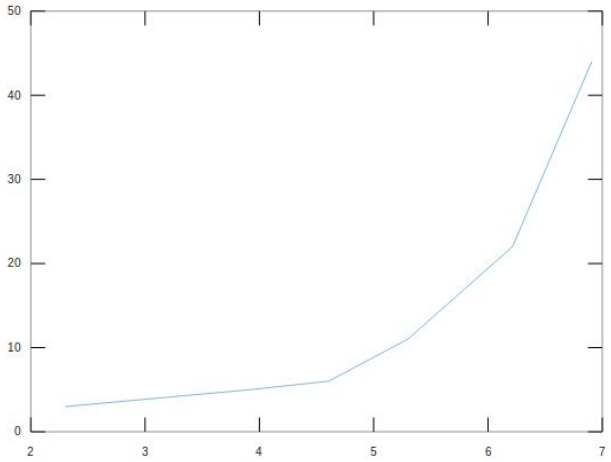
Torus:



3D:



Imp_line:



1.6 Interesting Findings

1. The line topology converged on less number of nodes compared to other topologies. Reason behind this could be forward and backward bouncing of messages between two neighbors which causes shutting down of the line path and this the convergence condition was unfulfilled in higher number of nodes.

2. Full topology has less convergence time for smaller networks of 100 nodes but for larger networks its convergence time increases exponentially due to large memory occupied by the adjacent node lists for every node. Surprisingly, Random 2D gives better performance for large number of nodes because size of adjacent list remains small for every node and we are even considering randomness factor in this.

3. Imperfect line works for larger number of nodes compared to plain line topology as the bouncing between just two neighbor nodes is reduced and the message is able to travel to more number of nodes and converge.