# Project 2 Report

**Contributors**:

- Rahul Singh Bhatia, 5993-0121
- Foram Bipinchandra Nirmal,6219-9417

**Topologies**:

- **Full Network**: Every actor is a neighbor of all other actors. That is, every actor can talk directly to any other actor.
- **Line**: Actors are arranged in a line. Each actor has only 2 neighbors (one left and one right, unless you are the first or last actor).
- **Random 2D Grid**: Actors are randomly position at x, y coordinates on a [0- 1.0] x [0-1.0] square. Two actors are connected if they are within .1 distance to other actors.
- **3D torus Grid**: Actors form a 3D grid. The actors can only talk to the grid neighbors. And, the actors on outer surface are connected to other actors on opposite side, such that degree of each actor is 6
- **Honeycomb**: Actors are arranged in form of hexagons. Two actors are connected if they are connected to each other. Each actor has maximum degree 3.
- **Honeycomb with a random neighbor**: Actors are arranged in form of hexagons (Similar to Honeycomb). The only difference is that every node has one extra connection to a random node in the entire network.

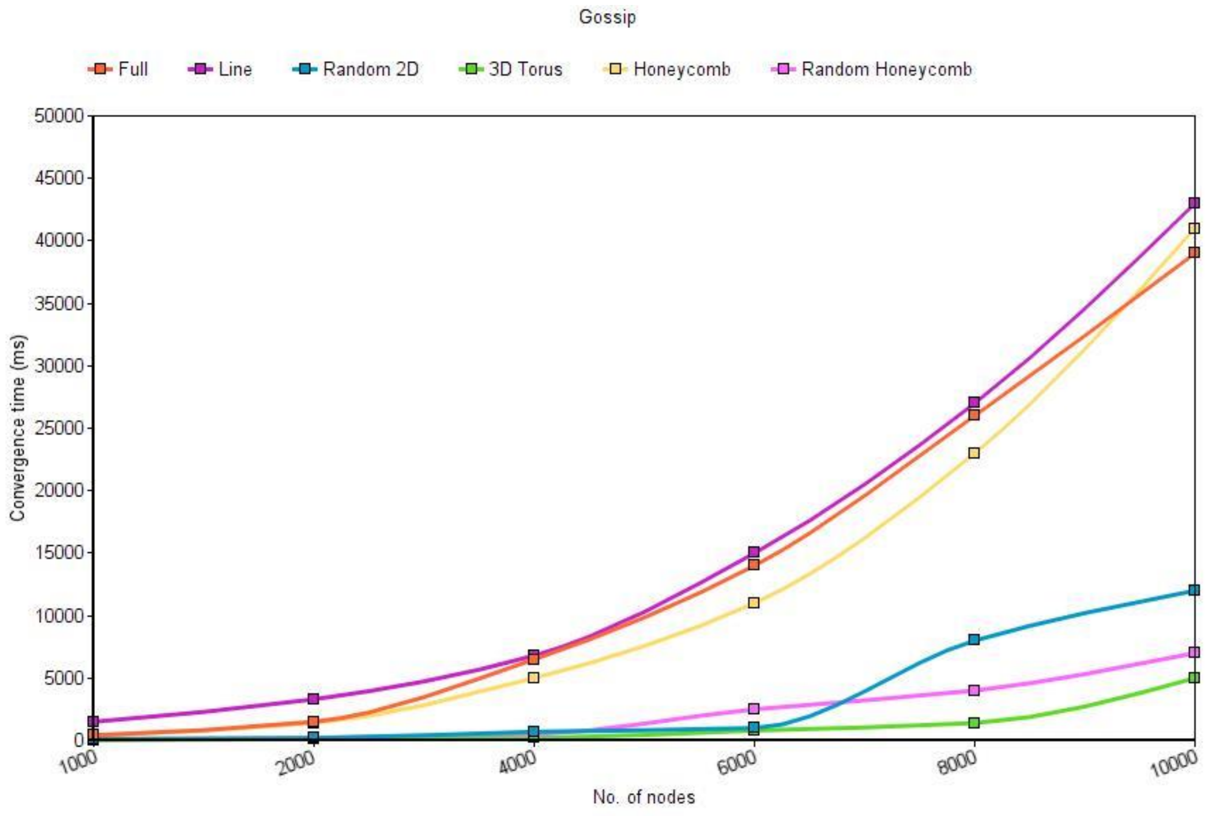
## Gossip Protocol

## Implementation:

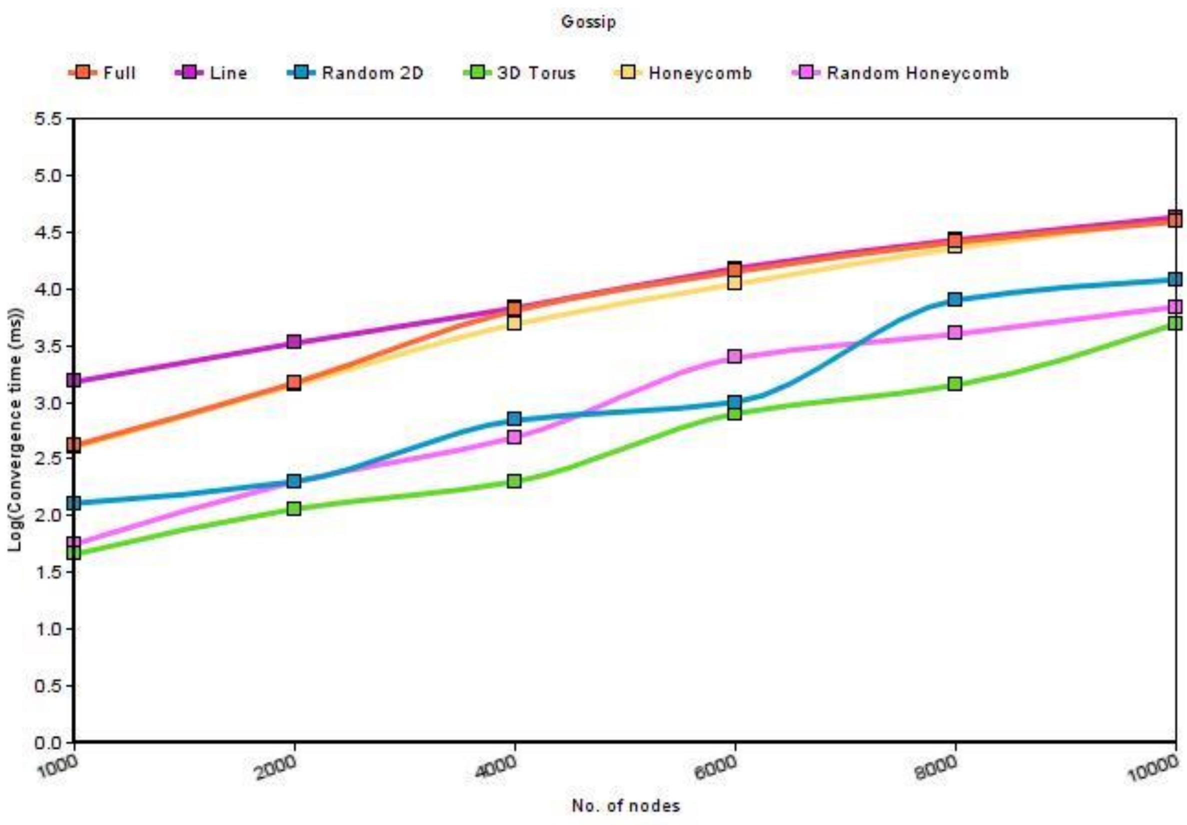
- A node is randomly chosen as the start point to transmit the message. This node sends the message to one of its neighbors
- It waits for 10 ms to send another message
- On receiving a message, each node sends the same message to its neighbors, following a timer of 10 ms
- A counter is maintained to check the number of times a message is received by the node. If this counter crosses 10, the node stops transmitting
- Once all the nodes have received the message 10 times, the programs terminates and prints the convergence time

**Observations:**

- Graph below depicts the convergence time for various topologies used



- It is observed that Random 2D, 3D torus and Random Honeycomb performs better than the rest for large number of nodes. Full and Honeycomb also prove to be useful for small no. of nodes
- Full network works fine for a small no. of nodes since all the actors will spread the rumor as quickly as possible due to good connectivity, and the program terminates quickly. But for larger number of nodes, it fails due to the overhead of asynchronous calling among huge number of nodes



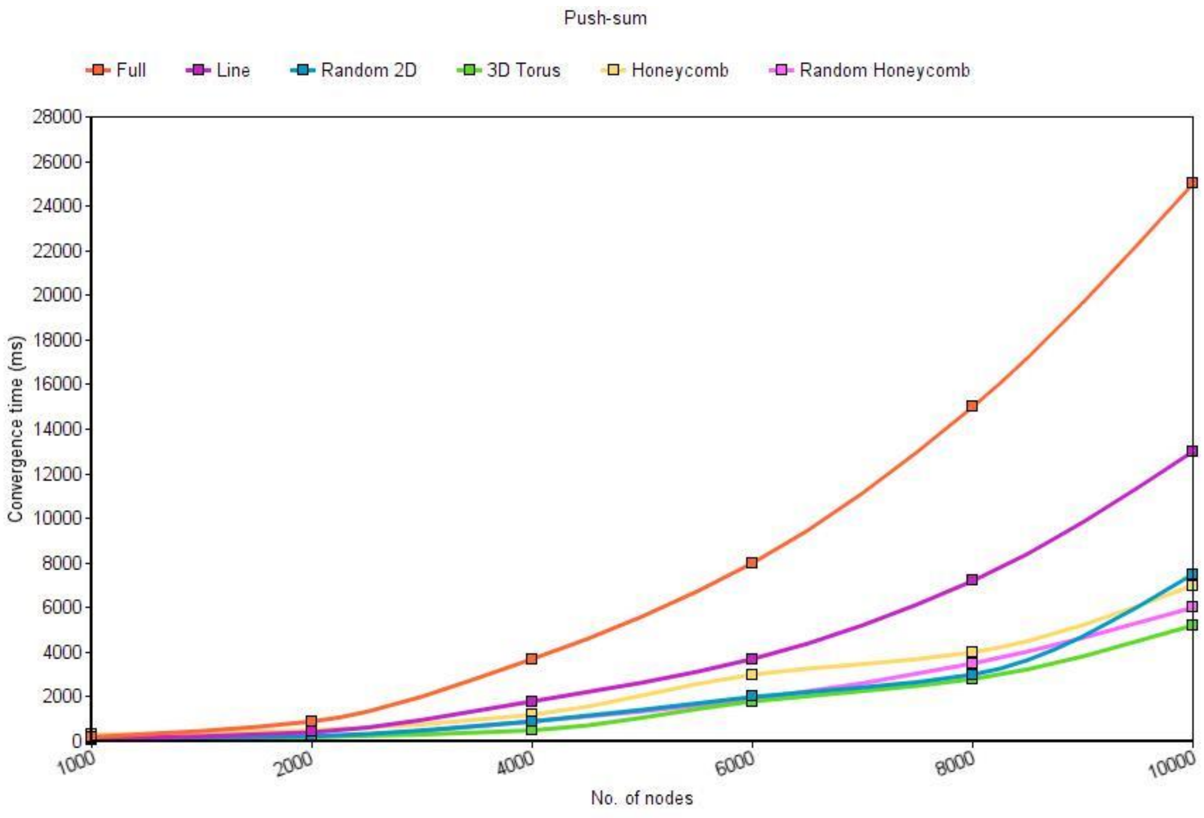
**Push-sum**

**Implementation:**

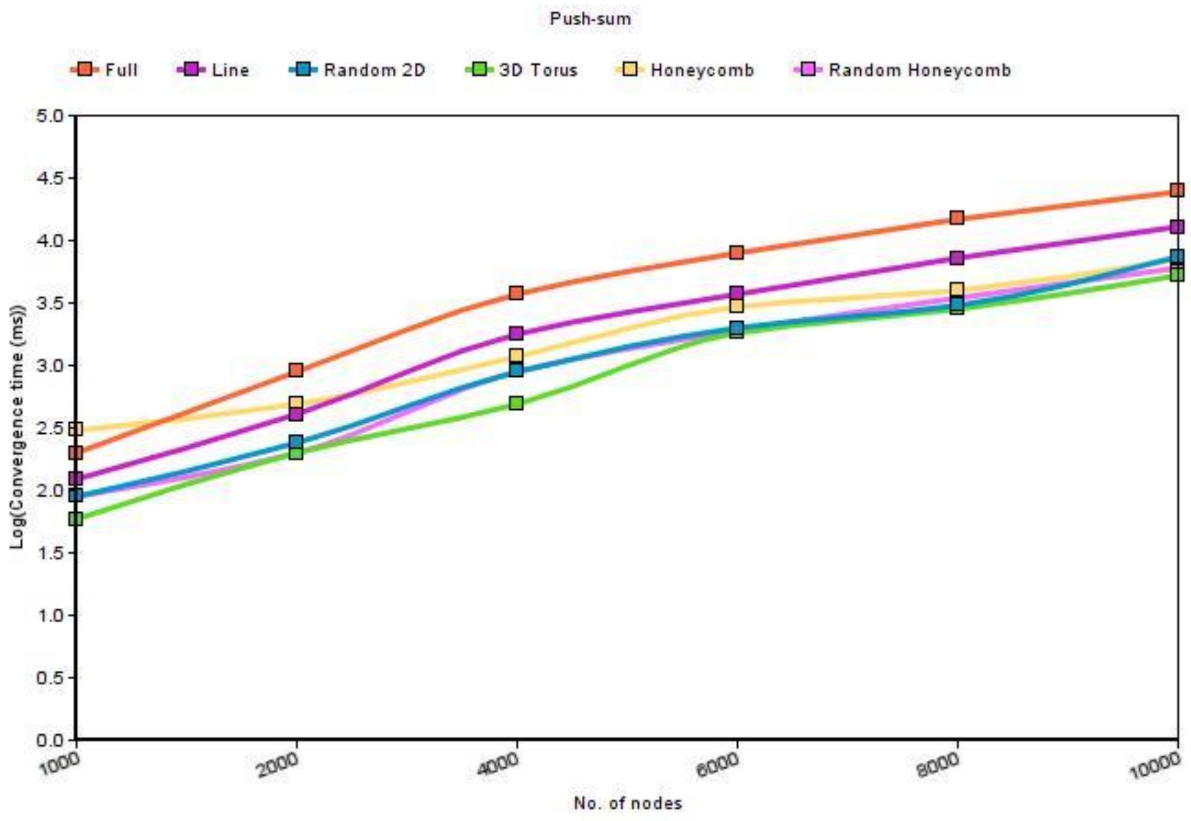
- Each node stores two values, i.e. s and w
- We pick a node randomly and start sending message to its neighbor. A 10 ms timer is maintained, after which the node again sends the message
- On receiving a message of the form {s,w}, node adds this to its s,w and sends half of it to its neighbor
- Once it finds out that the ratio ‘s/w’ has not changed by more than pow(10,-10), the node no longer terminates
- Once every node has converge, convergence time is printed out

**Observations:**

- Please refer the below graph which maps convergence time of various topologies



- Full and line are worse as compared to the rest of the topologies
- 3D torus performs the best in this protocol as well
- Though, in push-sum, Random 2D, Honeycomb, Random Honeycomb and 3D torus performs almost equivalently


Push-sum
Full
Line
Random 2D
3D Torus
Honeycomb
Random Honeycomb
Log(Convergence time (ms))
No. of nodes
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
1000
2000
4000
6000
8000
10000