Project 2: Project Report

Group Members

- Gloria Nazareth: 8221-8035
- Rishab Lokray: 9357-3447

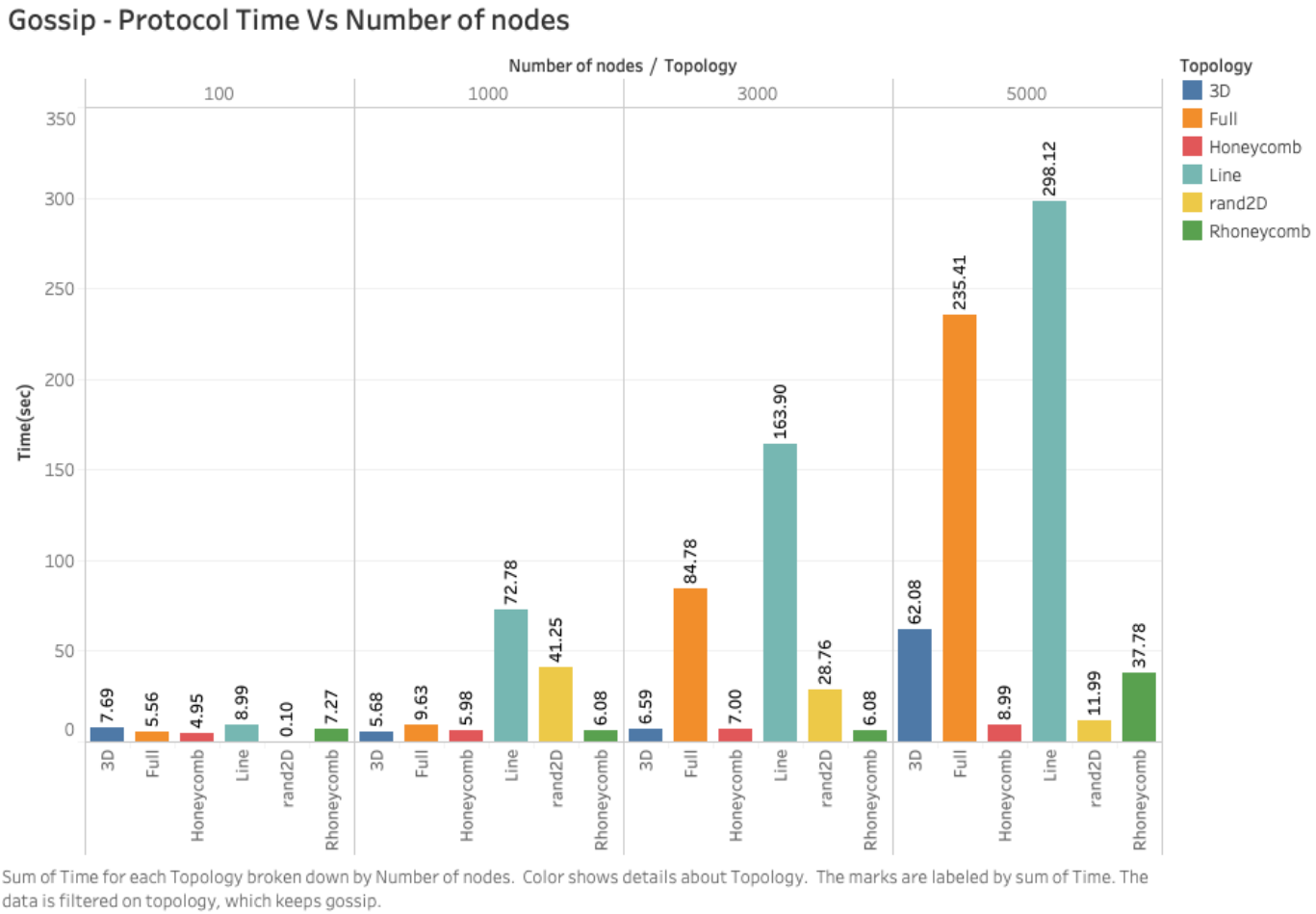
Topologies

- **Line**: All the actors are placed in a linear form forming a line, where an actor at position i has neighbours at positions i+1 and i-1. Maximum neighbours here are possible are 2 and minimum is 1.
- **Random 2D**: 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. This is an interesting case which have lot of degree of randomness here with unsure maximum and minimum neighbours. Will discuss more over this in our observations
- **3D torus**: Actors are placed in a 3 dimensional cube. With maximum neighbours 6.
- **Full**: in full topology an actor has all the others actors as its neighbour.
- **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.
- **Random Honeycomb**: 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:

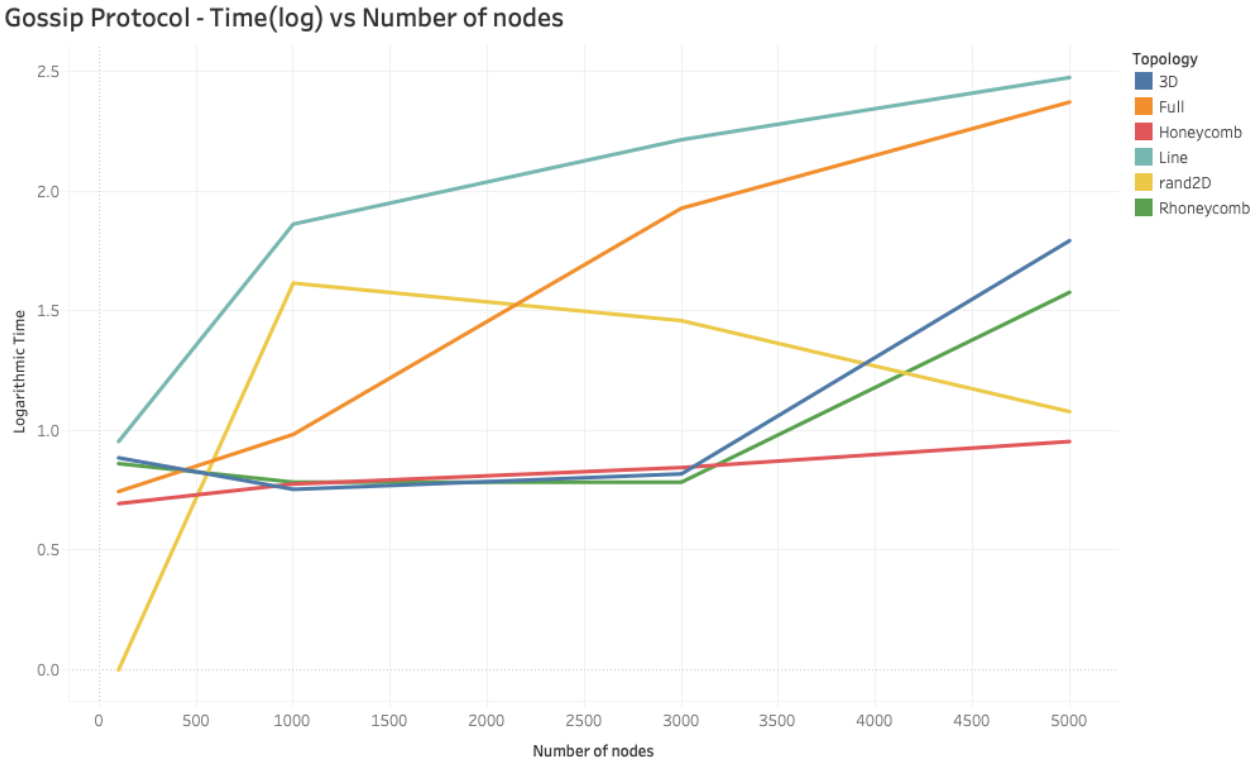
- Main module tells a random node a message.
- On receiving, each node sends all of its neighbours the message continuously after a periodic delay.
- Whenever a node receives a message , it tells the main module so that main module can keep track of the number of remaining nodes who have not received the message.
- Each node keeps track of the number of times it has received the message. If this count goes above 10, it stops sending the message to its neighbours.
- Once main module finds out that a preset percentage of nodes have converged it prints the total time it took and terminates the application.



Sum of Time for each Topology broken down by Number of nodes. Color shows details about Topology. The marks are labeled by sum of Time. The data is filtered on topology, which keeps gossip.

- Observation and reasoning: We can observe the times the different topologies take for different number of nodes.
- For fewer nodes the Line is the slowest algorithm and Honeycomb is the fastest
- As the number of nodes keep increasing the line topology performs the worst.
- The 2DRandom topology is a special case. There is no fixed output, as when the number of nodes are less <100 more nodes are disjoint and have no neighbours, hence it reaches convergence very quickly as barely any nodes get the message. For a larger set of nodes it finds enough neighbours so it takes lesser time with increasing number of nodes.
- This behaviour is the basic working principle of gossip where in Full network every node has all other nodes as neighbours, so it terminates quickly.
  But for larger numbers of actors full topology fails as the process overhead of asynchronous calling among actors takes more time and this time dominates the advantage of having all nodes as neighbours.
- Other topologies have limited number of neighbours so it performs better even with many nodes.

Comparison of time and number of nodes can also be understood by the log time vs nodes graph below which gives asymptotic convergence of time for different topologies.



PushSum Protocol

- Each node has four values in a tuple as its state. Its PID, input_neighbours input_state, input_weight. Where state and weight are same as defined in the problem statement.

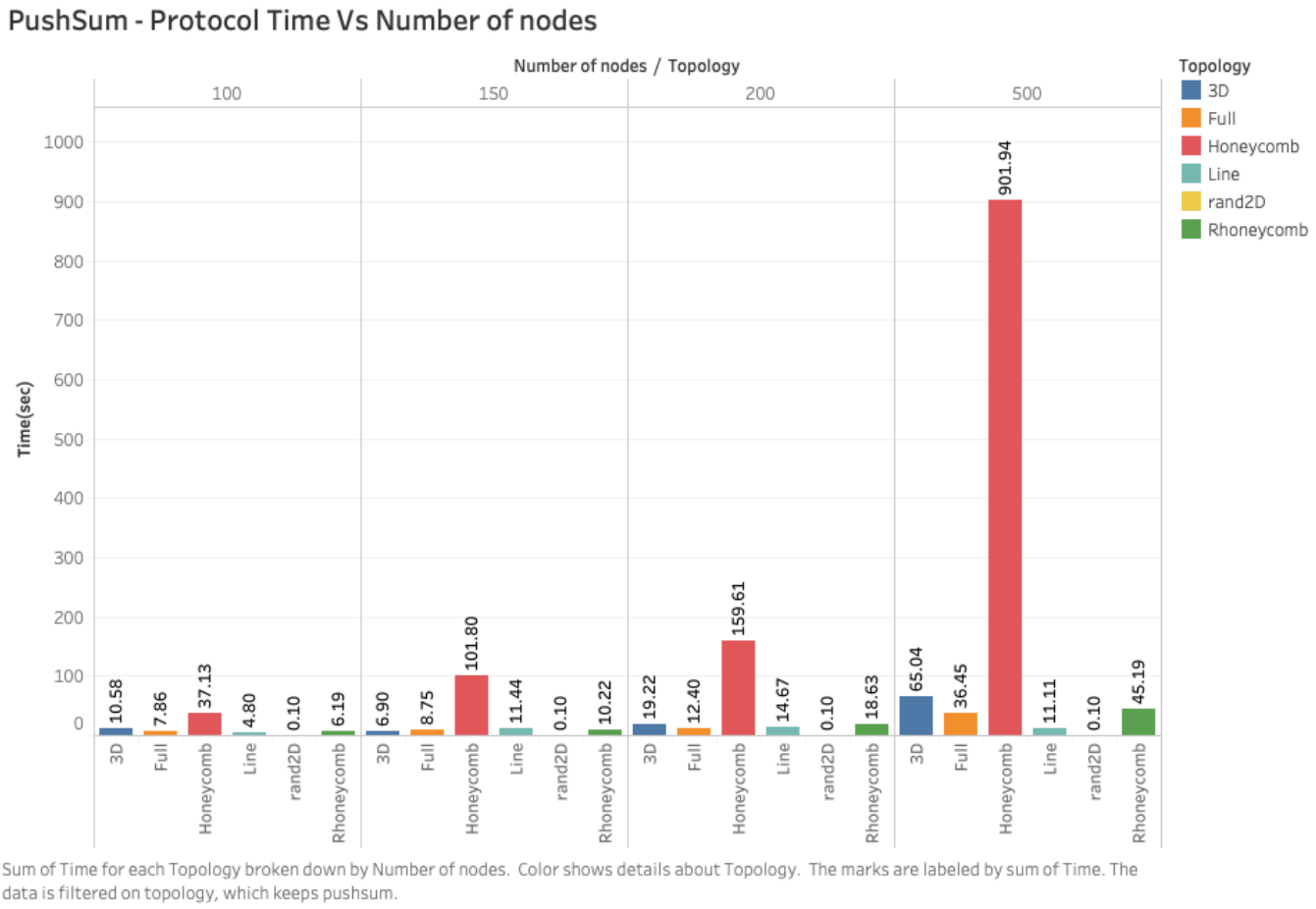
- When we start the protocol, the receiving node first adds its value of S & W with the received S and W and then send the half of it to a random neighbour and update its S and W with the newly halved value.

- Before sending the message every node checks if the difference between its original S and W and the newly formed S and W has increased more than 10^-10 at least. If not it terminated the node and converges.

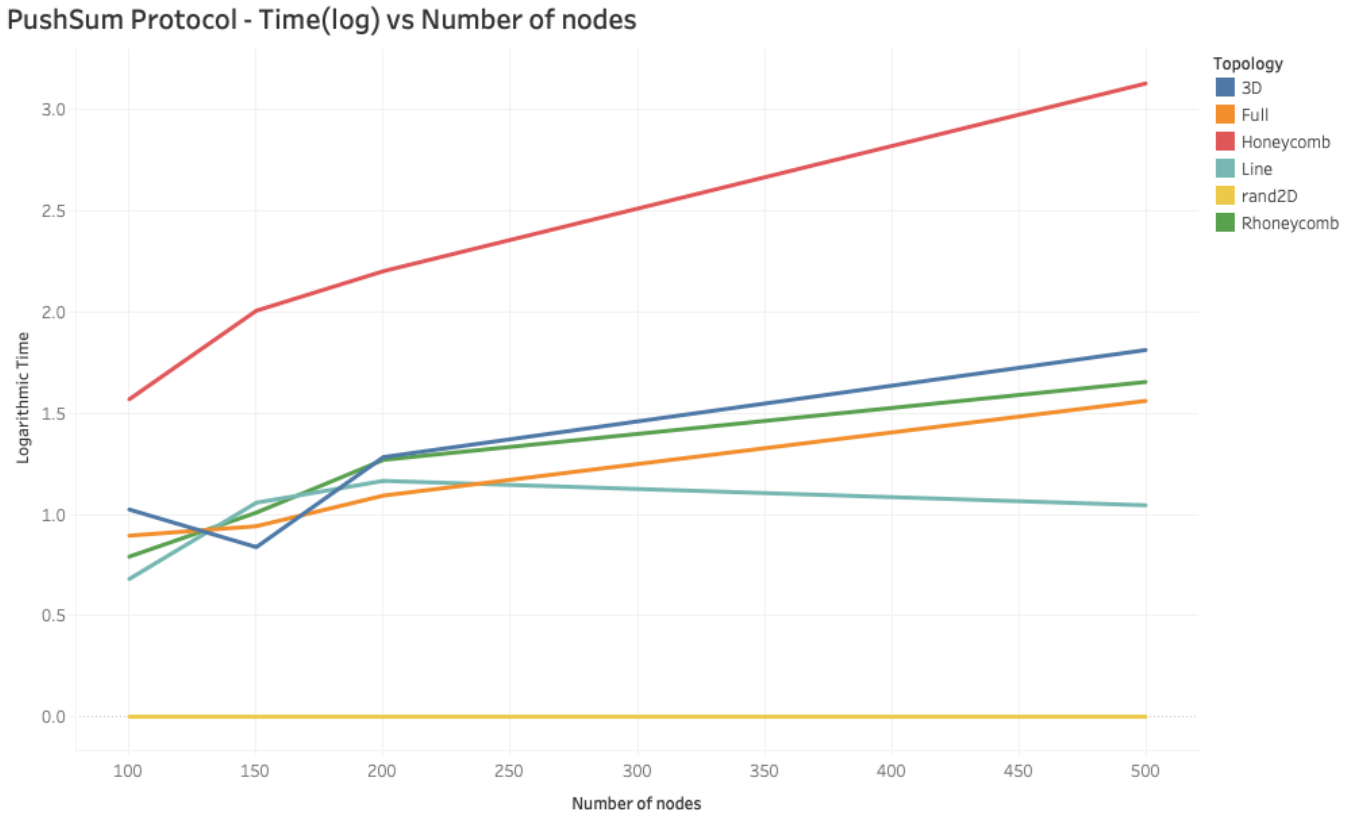
- When the node terminates then delete it from the lists of all its neighbours.

- If the list of neighbours of any node becomes empty after all the neighbours die we terminate the process.

Observations:



- The graphs show the times taken by various topologies at various set of nodes.
- The push sum is a linear propagation protocol so in our case honeycomb takes the most amount of time as there is no dual communication.
- Rand2D again has a very unpredictable performance because we do not know when a node has empty set of neighbours. So the program terminates immediately.
- We have simulated this system for a maximum of 500 nodes as propagation takes a lot of time in line/2d topologies and they would never converge.
- This comparison of time and numNodes can also be well understood by the log time vs nodes graph below which gives asymptotic convergence of time for different topologies



The trend of sum of Logarithmic Time for Number of nodes. Color shows details about Topology. The data is filtered on topology, which keeps pushsum.