Project 2: Project Report

Group Members

- Prafful Mehrotra UFID: 1099-5311
- Siddhant Mohan Mittal UFID: 6061-8545

Topologies

- Line: All the actors are placed in a linear order forming a line, where an actor at position i has neighbors at positions i+1 and i-1. Maximum neighbors possible are 2 and minimum is 1.
- Imperfect line: Actor placement is similar to line but in this case an actor(i) will also have a random actor as its neighbor apart from the forward(i+1) and backward(i-1) neighbors. Hence maximum neighbors possible are 3 and minimum is 2.
- 2D: Actors are placed in a 2D square grid which is formed if number of actors are perfect square (if it is not then we round it to nearest integer and then place the remaining actors on the edges of the 2D grid). Maximum number of neighbors are 4 and minimum is 2.
- Imperfect 2D: Same as 2D but each actor has one more random actor as its neighbor thus making maximum and minimum neighbors 4 and 3 respectively.
- 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 neighbors. Will discuss more over this in our observations
- 3D: Actors are placed in a 3 dimensional cube. With maximum neighbors 8 and minimum 3.
- Torus: it is two dimension with degree of 4, the nodes are imagined laid out in a two-dimensional rectangular lattice of n rows and n columns, with each node connected to its 4 nearest neighbors, and corresponding nodes on opposite edges connected. The connection of opposite edges can be visualized by rolling the rectangular array into a "tube" to connect two opposite edges and then bending the "tube" into a torus to connect the other two. communication can take place in 4 directions, +x, −x, +y, and −y.
- Full: in full topology an actor has all the others actors as its neighbor.

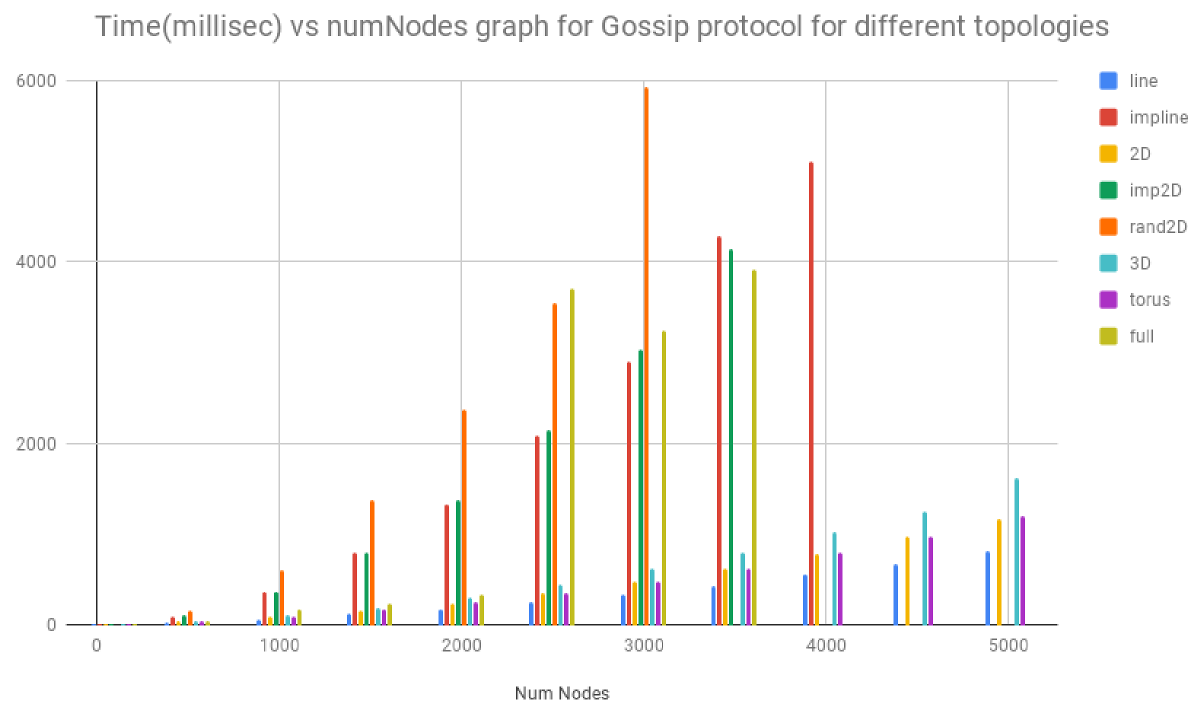
Gossip Protocol

Implementation:

- Driver module tells the middle node a message.
- On receiving, each node sends all of its neighbors the message.
- Whenever a node receives a message for the first time, it tells the driver module so that driver module can keep track of the number of remaining nodes who have not received the message and also update the list of nodes who have received the message
- Each node keeps track of the number of times it has received the message. If this count goes beyond 10, it stops propagating this message to its neighbors.
- Once driver module finds out that every node has received the message, it prints the total time it took and terminates the application.

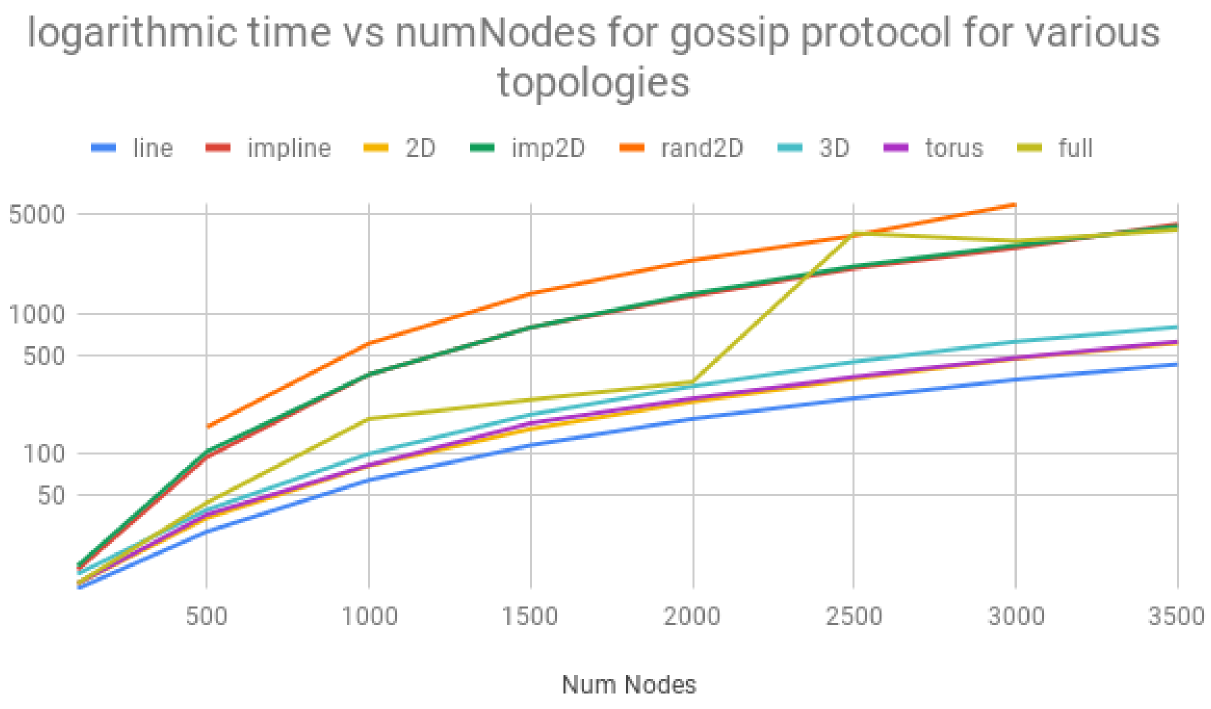
Observation and reasoning:

- Below graph shows the different time for various topologies used by us.



- Good topologies for gossip protocol are torus, 2D, line and 3D for large numbers of actors. For smaller values of actors full network also gives good convergence values.

- The reason behind this is based on the underlying principle of gossip protocol of spreading rumors. In Full network all actors are neighbors of the actor to which the rumor is spread first thus the program terminates quickly as every other actor will have the rumor from the first node.
- But for larger numbers of actors this topology fails as the process overhead of asynchronous calling among actors dominates the advantage of having all nodes as neighbors.
- Where torus, 2D and 3D perform well for large number of n due the balance between the number of neighbors and actor-neighbor asynchronous interaction
- 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.
- Note the time in each graph for a particular value of numNode is the average of three runs.



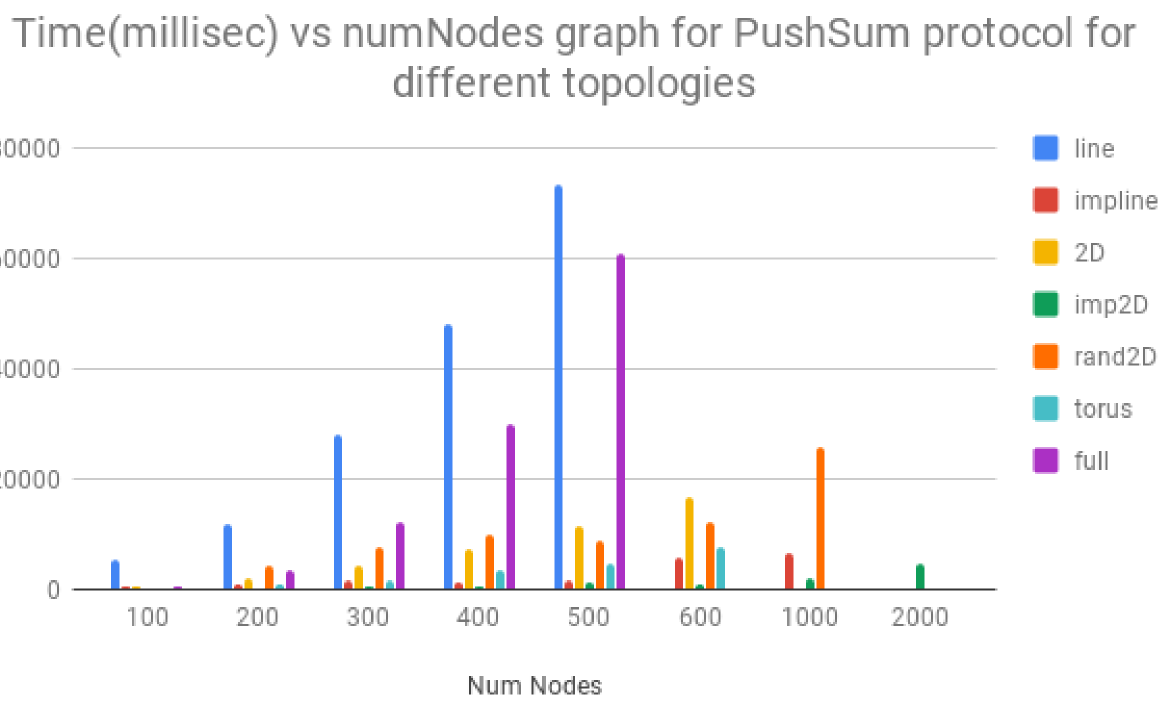
<u>PushSum Protocol</u>

Implementation:

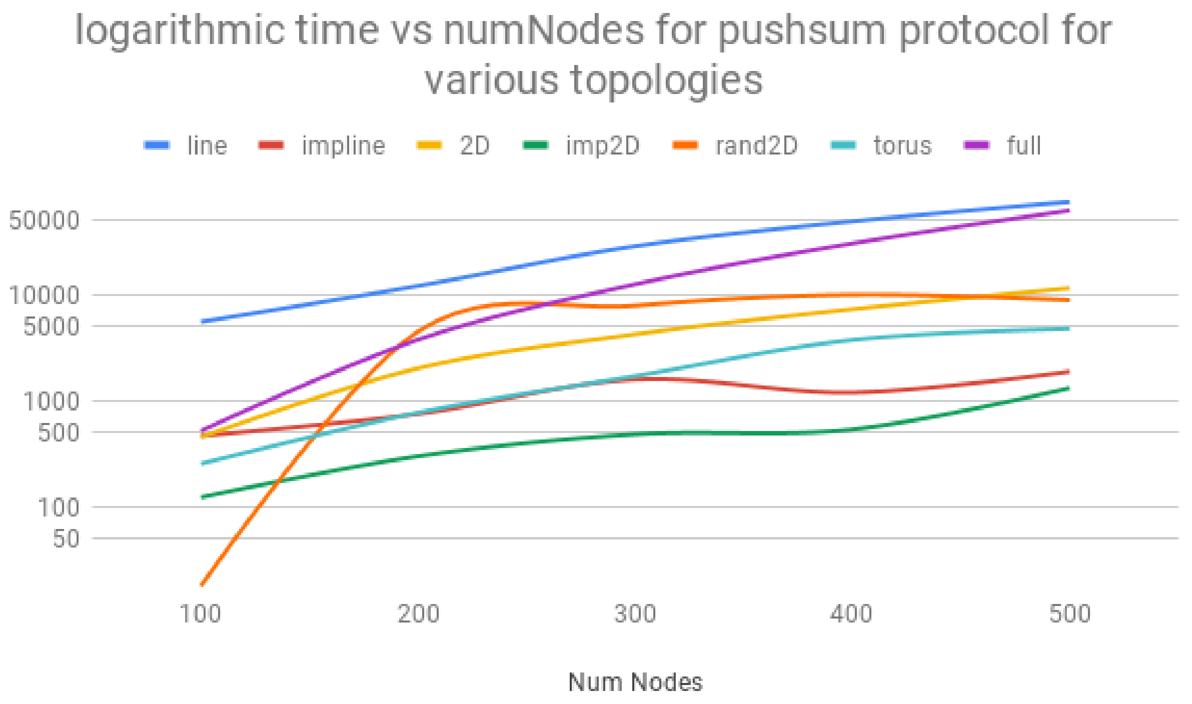
- Each node has four values in a tuple as its state. These values are s, w, s/w and change_count. ‘s’ and ‘w’ are same as defined in the problem statement and change_count is the number of times the ratio ‘s/w’ has not changed by more than pow (10,-10).Driver module initiates by selecting the middle node and tells it a message {0,0}.
- On receiving a message of the form{rec_s,rec_w}, a node adds this tuple to its state values s,w and sends half of the new state values to a randomly selected neighbour.
- Before sending, a node updates its ratio and change_count.
- Once the change_count has reached the value 3, it tells the driver function that it’s ratio (sum estimate) has converged. This node shall not transmit any longer.
- Once every node has converged, the driver function prints the total time it took to converge and terminates.

Observation and reasoning:

- Below graph shows the time taken by various topologies for different numNodes while propagating using PushSum Protocol.
- We can observer that line and full networks are worst performers which is completely opposite of what was happening in gossip protocol. Which is well justified with the implementation of both protocol. Since pushsum is more of a linear propagate (choosing one random neighbor at a time) the spreading of message takes more time in linear topologies. Also full network performs better than line because of random selection of neighbors hence it linearly propagates in different directions more of a zig zag propagation thus converging faster than line.
- A good thing to note here is also that imperfect line performs better than line because of that extra random neighbor it has and randomly neighbor picking makes it converge faster than line.
- Torus and 2D topology performs really well because of the fact that each node has degree 4 thus random neighbor picking makes the convergence fast
- Best topology for Pushsum protocol is Imperfect 2D because it has best of both the worlds that is chance of four-way directional propagation spread at each node and a random to increase the converging rate

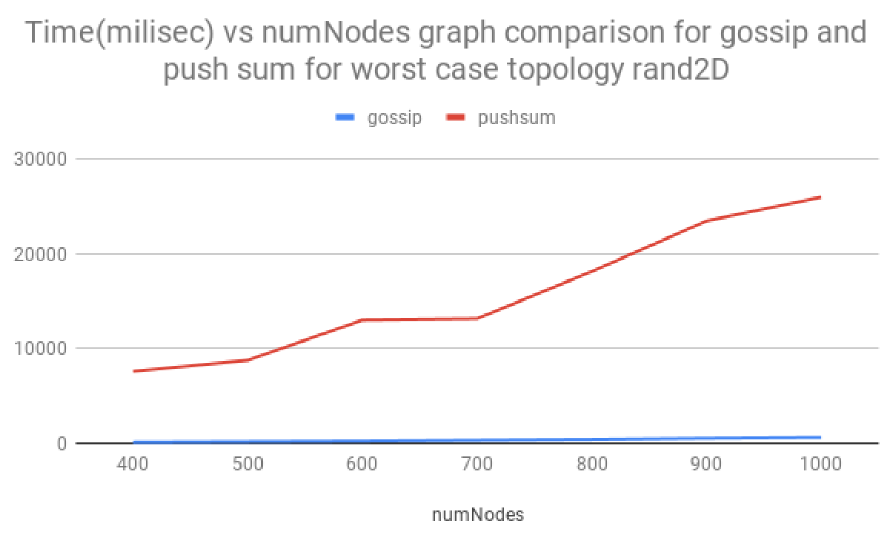


- This comparison of time and numNodes can also be well understood by the log time vs nodes graph below which gives asymptotic convergence of time for different topologies

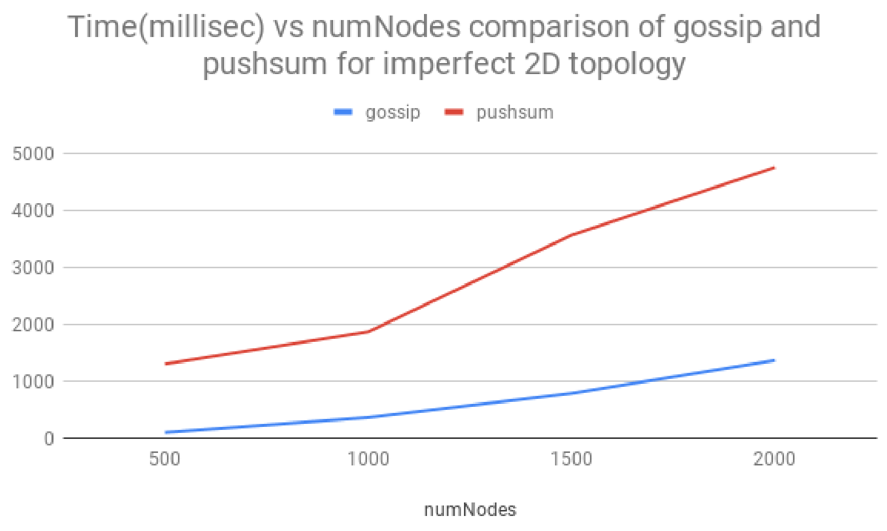


Some more interesting observation

- We compared the worst performing topology of gossip and pushsum protocol with each other to see which algorithm perform better
- The theory is well supported by our experiment as the gossip propagation takes less time for even its worst topology as compared to pushsum liner propagation
- Below is the graph for Random 2D topology for both the protocol.



- Another experiment compared the best topology for pushsum performance in both the protocol.
- We found that even though imperfect 2D performs very good for pushsum it is not able to beat the vast spread propagation of gossip protocol and imperfect 2D which is a mediocre topology for gossip protocol out perform its performance for pushsum protocol.


Time(millisec) vs numNodes comparison of gossip and pushsum for imperfect 2D topology
gossip
pushsum
5000
4000
3000
2000
1000
0
500
1000
1500
2000
numNodes