Project Report 2

Date: October 7th 2017

Project Team mates

1. **Shubham Agiwal- UFID 20562669**
2. **Karan Sharma- UFID 00174451**

Topologies Implemented

1. **Full Topology -** A nodes has all the other nodes in the network as its neighbor.
2. **Line Topology-** A node knows only its immediate front and back neighbor except in the case of first and the last node.
3. **2d Topology-** Actors form a 2D grid. The actors can only talk to the grid neighbors.
4. **Imperfect Topology-** Grid arrangement but one random other neighbor is selected from the list of all actors (4+1 neighbor).

Gossip Algorithm

We have implemented the gossip algorithm in two ways. There implementation details are given below with our observations

Gossip Algorithm implementation 1

A. In this gossip algorithm implementation, **we have assumed that convergence is reached when all the nodes have heard the rumor for the given gossip count.**

B. Nodes that have gossip count of 10 are terminated by killing the process and removing them from the alive node list kept on the server.

C. When we start our gossip algorithm we select a random neighbor out of the alive node list depending on the topology.

D. When a node dies, we randomly pick up a random node from the list of alive nodes and start the gossip protocol again

E. This process continues till all the nodes in the network are dead.

F. The largest network of nodes that we ran our program was **10,000**. We ran it on an 8-core i-7 processor.

Graphs for Convergence time vs number of nodes in the network for different topologies

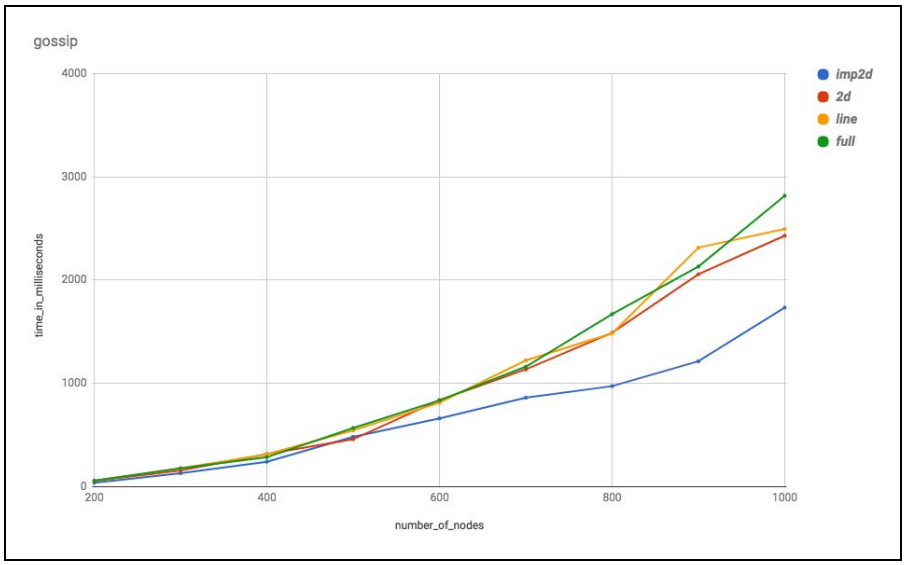

Fig 1. Convergence Graphs for different topologies on a linear scale for gossip algorithm

Analysis

From the Fig 1, we can clearly see that for the **assumed convergence criteria**, we find

imperfect 2D<2D<line<full

The above conclusion was obtained for networks unto 1000 nodes. We have ran the same code for a network of 10,000 nodes and similar results were obtained. But it was observed that for networks above 1000, there the time taken to converge would increase by a significant factor too for all the topologies.

num Nodes	imp2d(ms)	2d(ms)	line(ms)	full(ms)
100	38	56	58	57
200	131	156	178	177
300	241	316	313	285
400	482	459	545	568
500	661	838	813	835
600	861	1135	1222	1161
700	973	1490	1484	1670
800	1214	2057	2314	2131
900	1734	2430	2495	2816
1000	2149	3314	3360	3531
5000	102873	137171	129551	141831
10000	584791	722774	535185	797109

Interesting Points of Observation

1. Adding one extra random node in the 2D to make the topology to an imperfect 2D topology reduced the time to converge by a significant factor.

Gossip Algorithm implementation 2

A. In this gossip algorithm implementation, **we have assumed that convergence is reached when all the nodes have heard the rumor for the given gossip count or different nodes have sent continuous multiple dead messages for the number of nodes.**

B. When we start our gossip algorithm we select a random neighbor out of the alive node list depending on the topology.

C. Nodes that have gossip count are terminated by maintaining an is Alive state which is set to false once the gossip count reaches 10.

D. When a node is terminated, we randomly pick up a random node from the list of alive nodes and start the gossip protocol again

E. **This process continues till all the nodes in the network are dead or different nodes have sent continuous multiple dead messages for the number of nodes.**

F. The largest network of nodes that we ran our program was **10,000**. We ran it on an 8-core i-7 processor.

Graphs for Convergence time vs number of nodes in the network for different topologies

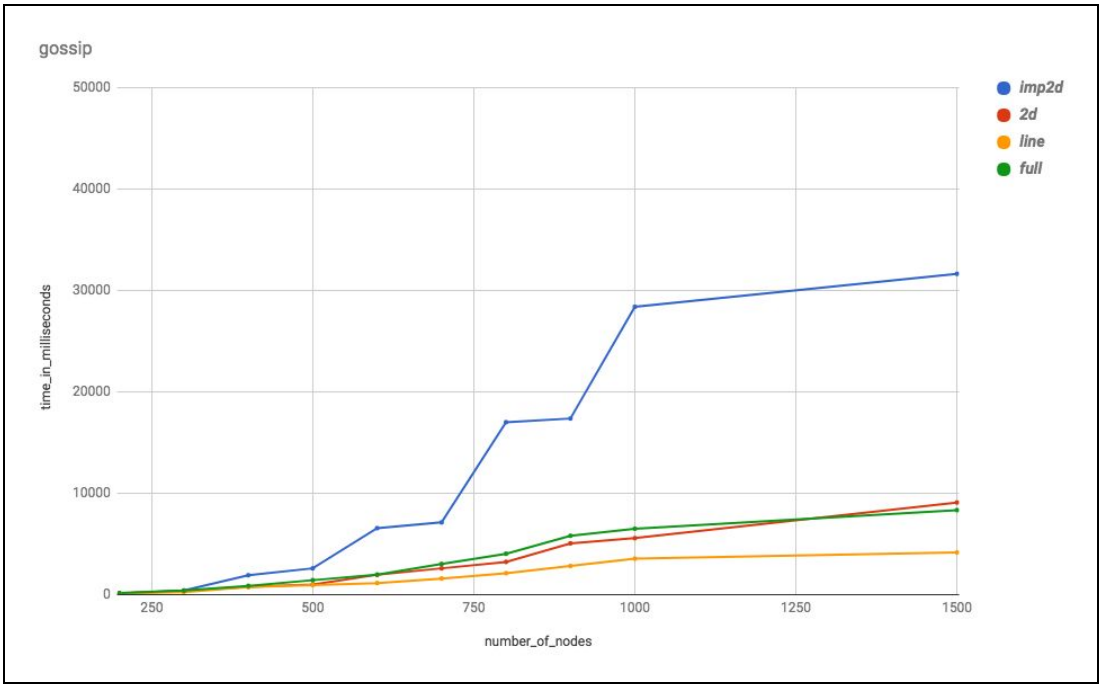

Fig 2. Convergence Graphs for different topologies on a linear scale for Gossip algorithm

Analysis

From the Fig 2, we can clearly see that for the **assumed convergence criteria**, we find

line<full<2D<imp2D

The above conclusion was obtained for networks unto 1500 nodes. We have ran the same code

num Nodes	imp2d(ms)	2d(ms)	line(ms)	full(ms)
100	61	76	56	103
200	354	310	207	350
300	1862	727	700	818
400	2541	945	901	1375
500	6507	1903	1092	1927
600	7070	2542	1539	2975
700	16955	3178	2062	3983
800	17319	5005	2775	5752
900	28337	5520	3495	6449
1000	31587	9034	4112	8285
1500	43555	19886	10062	21681

for a network of 10,000 nodes and similar results were obtained. But it was observed that for networks above 1000, there the time taken to converge would increase by a significant factor too for all the topologies.

Interesting Points of Observation

1. Adding one extra random node in the 2D to make the topology to an imperfect 2D topology and adding is alive state we could observe that it increased the time by 4 times compared to a 2D architecture.

Conclusion for the above implementations for gossip algorithm

From the above implementations, we concluded that gossip is not a reliable algorithm to converge in a given network. If the convergence criteria are changed we can obtain different results. Based on the convergence criteria, different topology work in different fashion to achieve the convergence criteria. We also implemented the algorithm where the convergence criteria was dependent on the ratio between the dead_nodes/ total number of nodes. From that we concluded that, gossip never converges for line and 2d topologies. So we decided to submit the second implementation as we found out that when the node sends dead messages most of the nodes in the network have converged.

Push-sum Algorithm

We have implemented the push sum algorithm in a single way. The implementation details are given below with our observations

Push sum Algorithm implementation

A. In the push sum algorithm implementation, **we have assumed that convergence is reached when all the nodes have reached an s/w ratio of less than 10^-10 for three consecutive counts**

B. Nodes that reach the above mentioned convergence criteria are killed(or terminated)and removed from the alive node list kept on the server.

C. When a node dies, we randomly pick up a node from the list of alive nodes and start the push sum protocol again.

D. This process continues till all the nodes in the network are dead.

E. We terminate the algorithm after all the nodes in the network achieve convergence. We do this because on a large network, if we terminate the algorithm after the first node converges, the average estimate of the other nodes in the network will differ from the average estimate of the converged node. Moreover, the average estimate of the converged node varied from the actual average by a significant factor for large network, if we terminate the algorithm after the first node converges.

F. The largest network of nodes that we ran our program was **10,000**. We ran it on an 8-core i-7 processor.

Graphs for Convergence time vs number of nodes in the network for different topologies

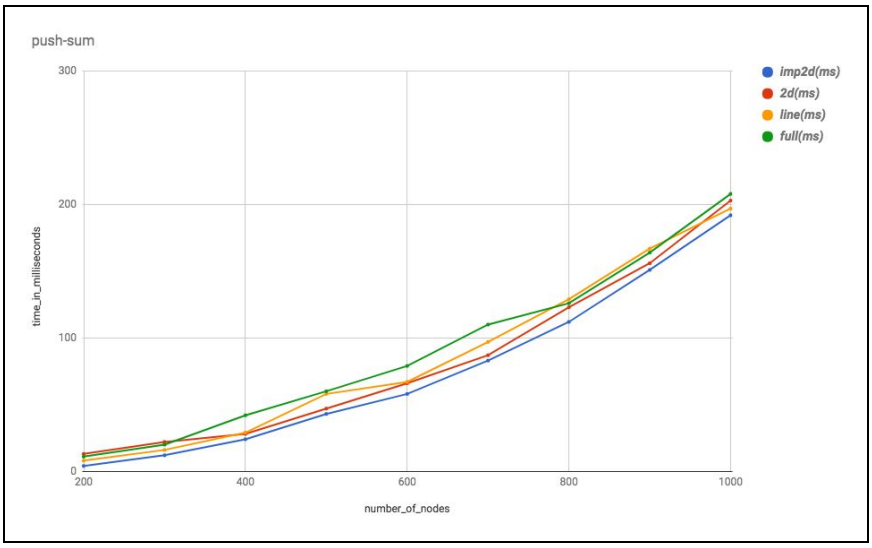

Fig 3. Convergence Graphs for different topologies on a linear scale for push sum algorithm

Analysis

From the Fig 1, we can clearly see that for the **assumed convergence criteria**, we find

imperfect 2D<2D<line<full

The above conclusion was obtained for networks unto 1000 nodes. We ran the same code for a network of 10,000 nodes and similar results were obtained. But it was observed that for networks above 1000, there the time taken to converge would increase by a significant factor too for all the topologies. But it is hard to find which is faster or slower as they vary by +300/-300 ms for most of the topologies.

num Nodes	imp2d(ms)	2d(ms)	line(ms)	full(ms)
100	4	13	8	11
200	12	22	16	20
300	24	28	29	42
400	43	47	58	60
500	58	66	67	79
600	83	87	97	110
700	112	123	129	126
800	151	156	167	164
900	192	203	197	208
1000	254	264	246	259
5000	5699	5630	5614	5703
10000	22684	22825	22552	22588

Interesting Points of Observation

1. Adding one extra random node in the 2D to make the topology to an imperfect 2D topology did not significantly affect a large network of 10,000 nodes by a major factor.

Conclusion for the above implementations for push sum algorithm

From the above implementations, we concluded that push sum is a reliable algorithm to converge in a given network. If the convergence criteria are changed we can obtain different results. Based on the convergence criteria, different topology work in different fashion to achieve the convergence criteria. We also implemented the algorithm where the convergence criteria was dependent on the ratio between the dead_nodes/ total number of nodes. From that we found out that, push sum converges for line and 2d topologies as well compared to gossip protocol