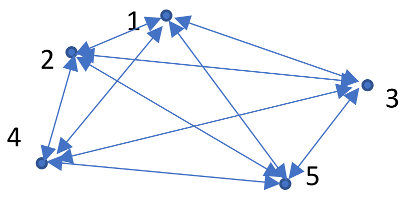
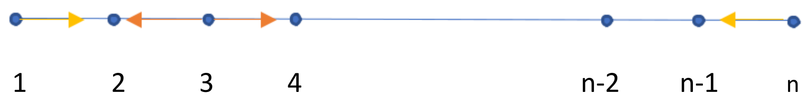
PROJECT-2 REPORT

Topologies

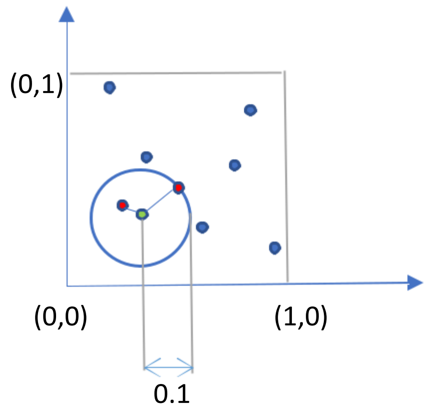
- **Full –** In full topology, one actor can send message to any other actor in the network by randomly choosing an actor. For example: Actor 1 can send message to any of the actors 2, 3, 4 or 5.



- **Line –** In line topology, an actor can send message to any of its two neighbors which are present on each side of the actor. The first and last one in the line can send message to only one actor as they have only one neighbor. For example, Actor 3 can send message to actor 2 or 4. Actor 1 can send message to only 2, and n can send message only to n-1.



- **2D-Rand –** In 2D-Rand, random (x, y) coordinates are generated within [0-1.0] x [0-1.0] range. An actor can send message to any actor which is within 0.1 distance. For example, the actor in green color can send message to actors in red color since they are within 0.1 distance from green actor.

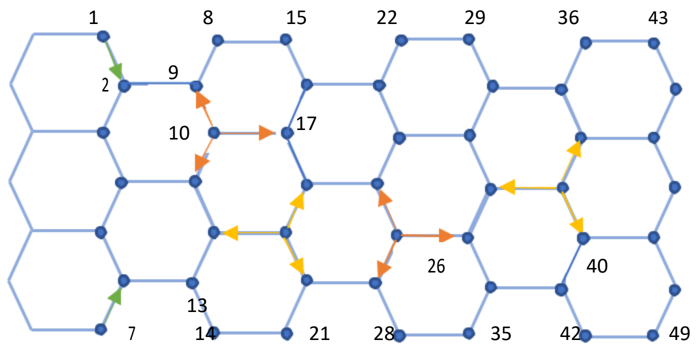


- **Honeycomb –** In honeycomb, the actors are arranged in the form of honeycomb, and they can send message to the neighbors as explained below. We have made a rectangular honeycomb as shown in the picture. The input is approximated to the nearest square of odd number.

  Following cases are possible, and the neighbors will be decided accordingly:

  For example: If the input nodes are 50, the number of nodes is approximated to 49 to create the rectangular honeycomb pattern. In this case, the length and width of the rectangle is 7 (sqrt 49).

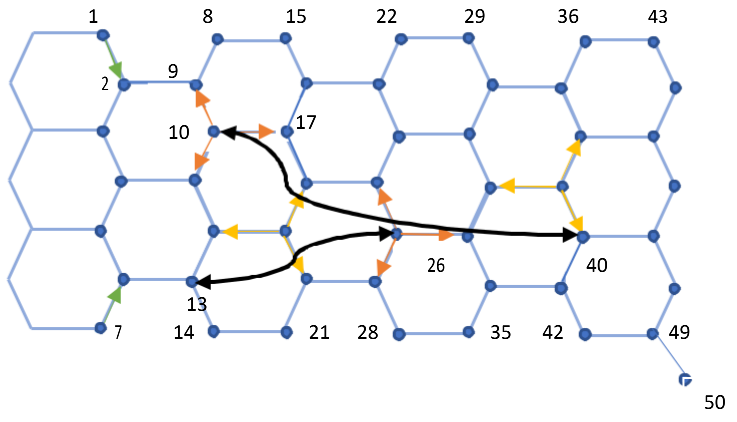
  Case 1: For inner nodes, if the actor number "n" is odd, the neighbors will be: n-1, n+1, n-7 (degree 3)
  Case 2: For inner nodes, if the actor number "n" is even, the neighbors will be: n-1, n+1, n+7 (degree 3)
  Case 3: For left outer nodes, if the actor number "n" is odd, the neighbors will be: n-1, n+1 (degree 2)
  Case 4: For left outer nodes, if the actor number "n" is even, the neighbors will be: n-1, n+1, n+7 (degree 3)
  Case 5: For right outer nodes, if the actor number "n" is odd, the neighbors will be: n-1, n+1, n-7 (degree 3)
  Case 6: For right outer nodes, if the actor number "n" is even, the neighbors will be: n-1, n+1 (degree 2)
  Case 7: For top outer nodes, if the actor number "n" is odd, the neighbors will be: n+1, n-7 (degree 2)
  Case 8: For top outer nodes, if the actor number "n" is even, the neighbors will be: n+1, n+7 (degree 2)
  Case 9: For bottom outer nodes, if the actor number "n" is odd, the neighbors will be: n-1, n-7 (degree 2)
  Case 10: For bottom outer nodes, if the actor number "n" is even, the neighbors will be: n-1, n+7 (degree 2)
  Case 11: For the corner nodes (1, 7, 43, 49), the neighbors are added accordingly.



- **Rand-Honeycomb –** In Rand-honeycomb, the actors will have similar neighbors as in honeycomb, with an additional random neighbor as displayed below. The random neighbor coupling is unique. The input is approximated to the nearest square of odd number + 1 (Note: For coupling two nodes, the total number of nodes should be even).

  For example: If the input nodes are 52, the number of nodes is approximated to 49+1 = 50 to create the rectangular honeycomb pattern. In this case, the length and width of the rectangle is 7 (sqrt 49).

  In case of actor 10, the neighbors in honeycomb topology are 9, 11 and 17, along with a random neighbor 40, and 10 gets added as a neighbor for actor 40 (40's random neighbor is 10). This pairing (10, 40) is unique.

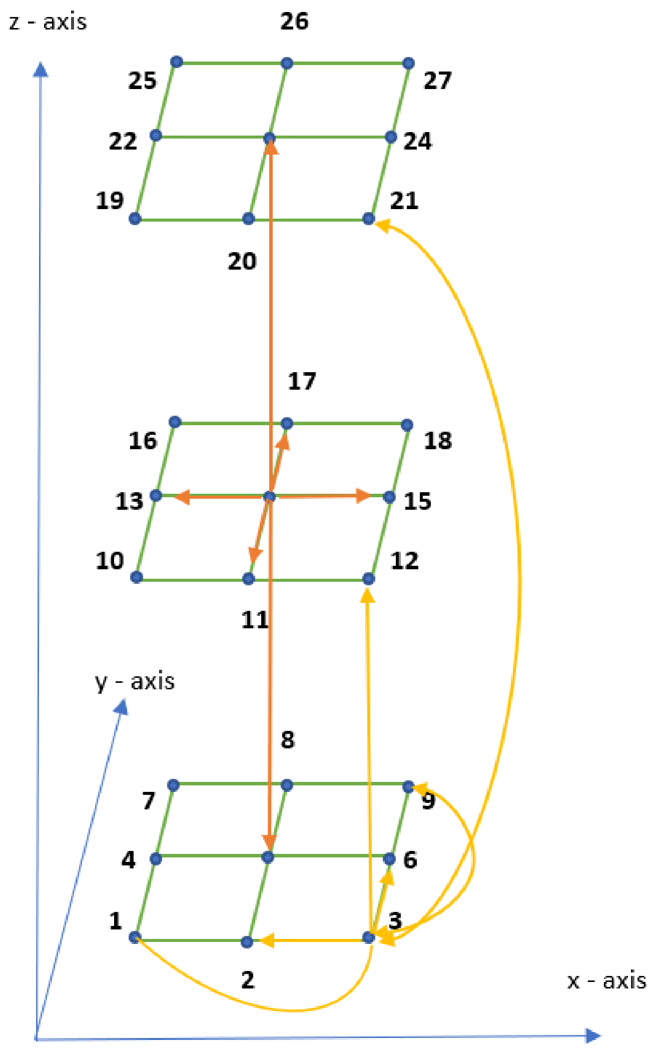


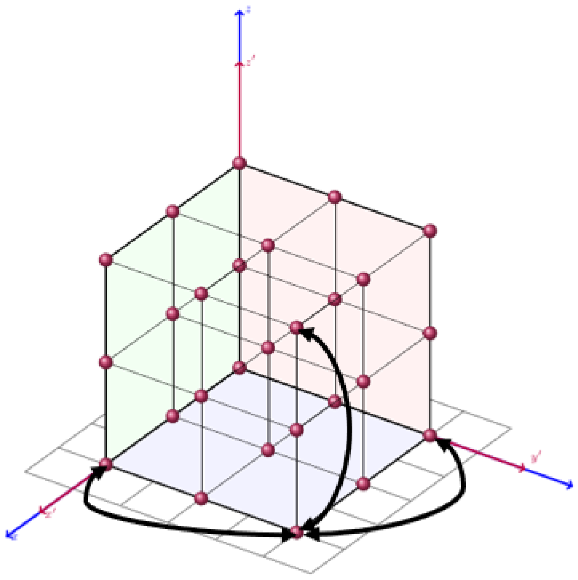
- **3D-Torus –** In 3D-Torus, the actors form a 3D-grid as shown below. The actors will be connected to the neighboring actors, as shown in the picture.

  The number of nodes will be approximated to the nearest cube. For example: If number of nodes entered is 60, the algorithm will take 64 as the total nodes (cube of 4).

  Case 1: The inner nodes, the neighbors will be the nodes on both sides on x - axis, y - axis, and z - axis. For example:
  for node 14, the neighbors will be: 13 and 15 on x - axis, 11 and 17 on y - axis, 5 and 25 on z - axis.

  Case 2: For the outer nodes on edges of the cube, the nodes will have one neighbor on each axis, and 3 other neighbors as shown in the picture. For example: for node 3, the neighbors will be: 2 on x - axis, 6 on y - axis, 12 on z -
  axis, and three other neighbors to form a torus: 1 on x - axis, 9 on y - axis, 21 on z - axis.



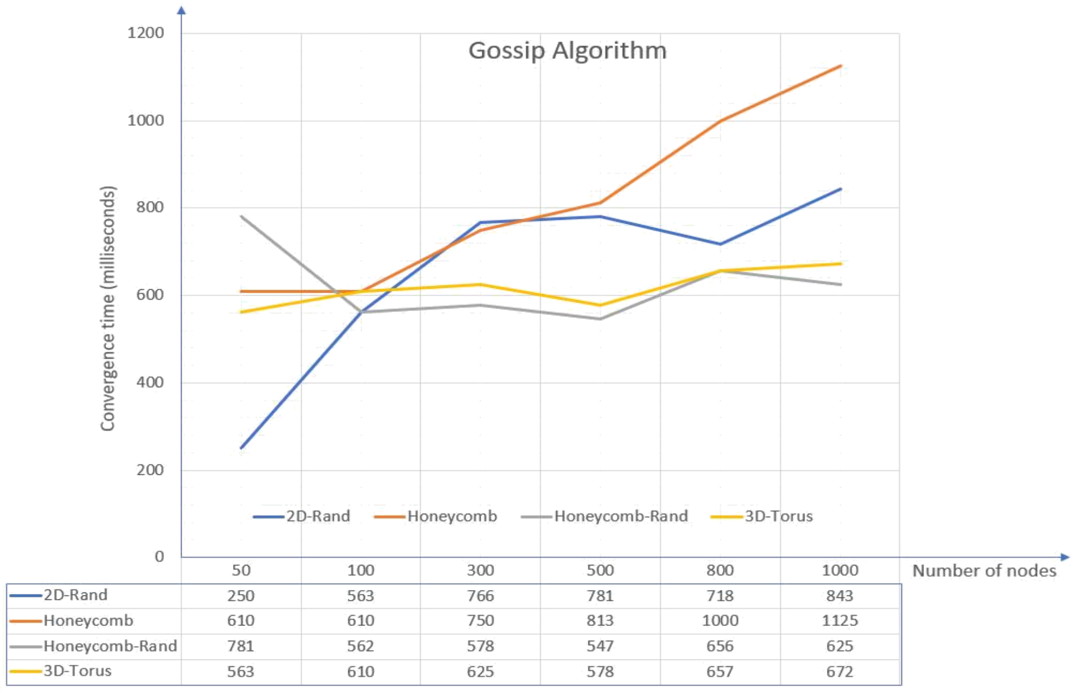


Gossip Algorithm observations

1. All nodes converged for all the topologies.
2. 2DRand is showing 100% convergence for more than 300 nodes but failing to obtain 100% convergence for nodes below 300.
3. The above scenario is happening because while creating nodes, they were randomly created on a 2D grid and if they did not lie within 0.1 distance, they were not connected to one another. Since there is poor connection between nodes for values below 300, there is less scope for complete convergence.
4. In our implementation, every actor sends messages periodically to its neighbors from the time it received its first message to the time it terminated. Convergence is achieved due to this model. It is also observed that the periodic time interval chosen to send messages also affects total convergence time.
5. In case of randhoneycomb, when the count value was increased, the nodes were exhibiting consistent convergence, and this is happening because when count is increased, every node's probability to receive message increases.
6. Additionally, the increase in count value is increasing the time taken for convergence.

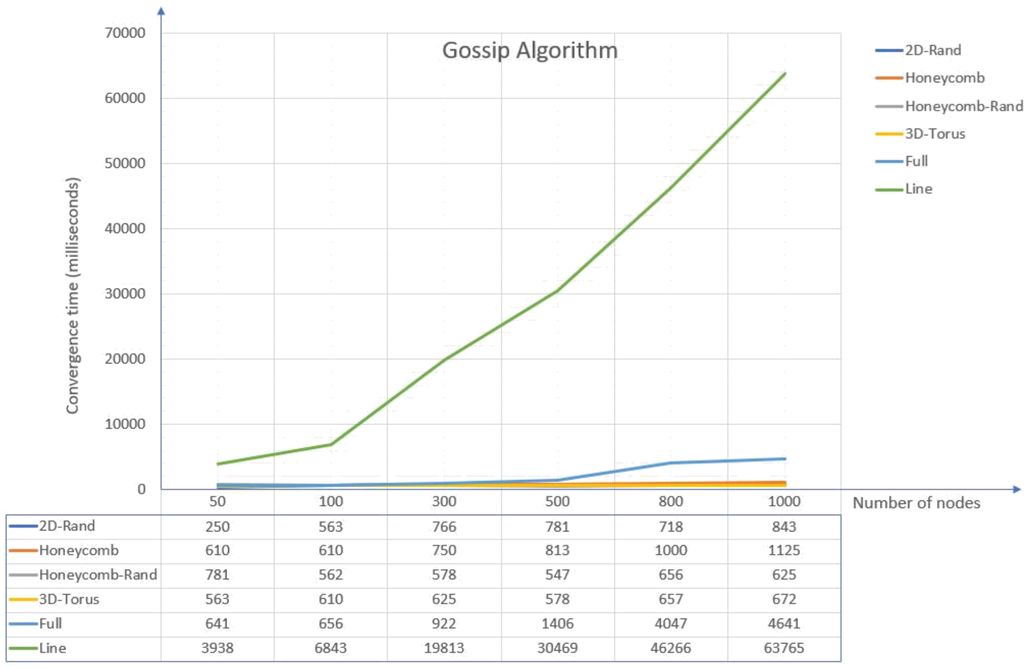
Topology	Total Nodes	Time (milliseconds)	Nodes Converged	Count
Randhoneycomb	1090	2594	1090	50
	1090	1890	1090	40
	1090	1406	1090	30
	1090	1063	1090	20

7. Order of time taken for convergence is as below
   T(3DTorus) ~ T(randhoneycomb) <T(2DRand) < T(honeycomb) < T(full) < T(line)



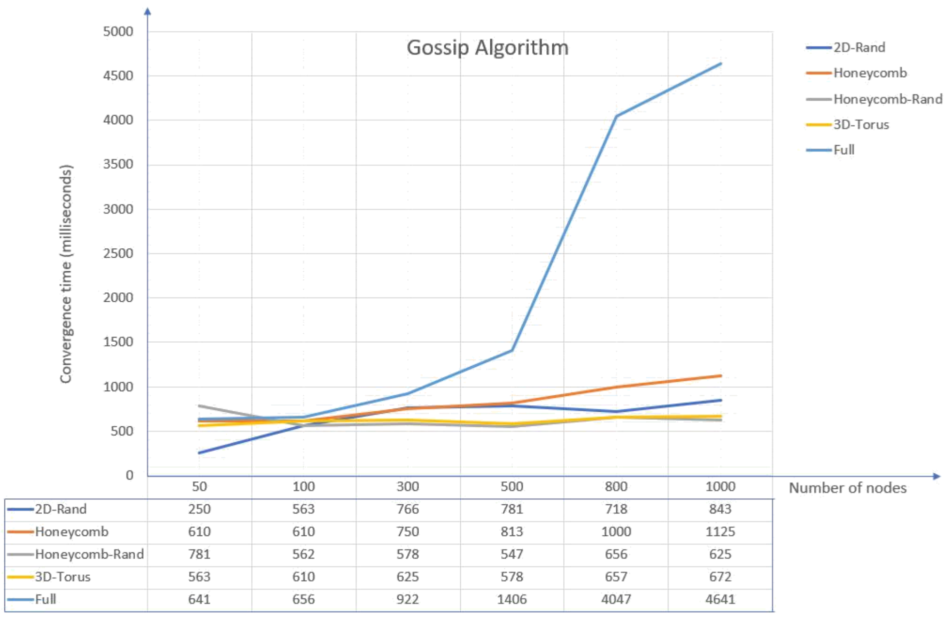
	50	100	300	500	800	1000
2D-Rand	250	563	766	781	718	843
Honeycomb	610	610	750	813	1000	1125
Honeycomb-Rand	781	562	578	547	656	625
3D-Torus	563	610	625	578	657	672

Figure:1. Above figure displays topologies with similar convergence rate



	50	100	300	500	800	1000
2D-Rand	250	563	766	781	718	843
Honeycomb	610	610	750	813	1000	1125
Honeycomb-Rand	781	562	578	547	656	625
3D-Torus	563	610	625	578	657	672
Full	641	656	922	1406	4047	4641
Line	3938	6843	19813	30469	46266	63765

Figure:2. In the below figure, Line topology displays inconsistent performance, and worst convergence rate



	50	100	300	500	800	1000
2D-Rand	250	563	766	781	718	843
Honeycomb	610	610	750	813	1000	1125
Honeycomb-Rand	781	562	578	547	656	625
3D-Torus	563	610	625	578	657	672
Full	641	656	922	1406	4047	4641

Figure:3. Full topology displays consistent 100% convergence rate but takes more time than other topologies.

Full topology - Gossip

Number of Nodes	Convergence Time (milliseconds)
50	641
100	656
300	922
500	1406
800	4047
1000	4641

Line topology - Gossip

Number of Nodes	Convergence Time (milliseconds)
50	3938
100	6843
300	19813
500	30469
800	46266
1000	63765

2D-Rand topology - Gossip

Number of Nodes	Convergence Time (milliseconds)
50	250
100	563
300	766
500	781
800	718
1000	843

Honeycomb topology - Gossip

Number of Nodes*	Convergence Time (milliseconds)
49	610
121	610
289	750
529	813
841	1000
1089	1125

Honeycomb-Rand topology - Gossip

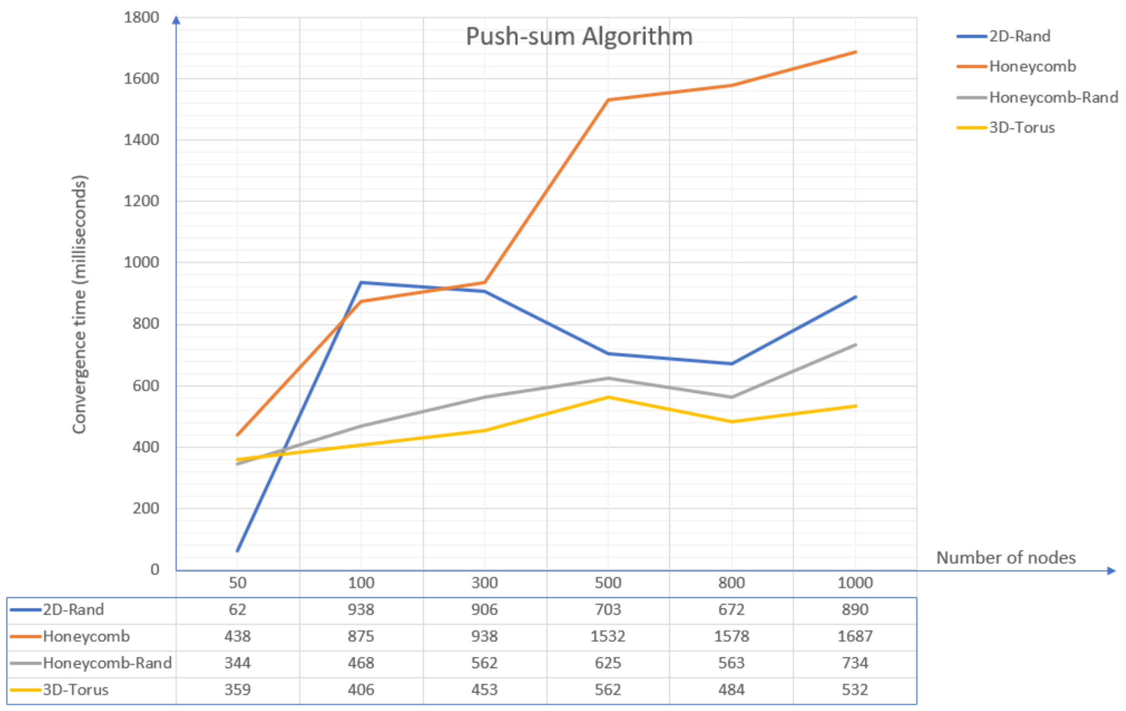
Number of Nodes*	Convergence Time (milliseconds)
50	781
122	562
290	578
530	547
842	656
1090	625

3D-Torus topology - Gossip

Number of Nodes*	Convergence Time (milliseconds)
64	563
125	610
343	625
512	578
729	657
1000	672

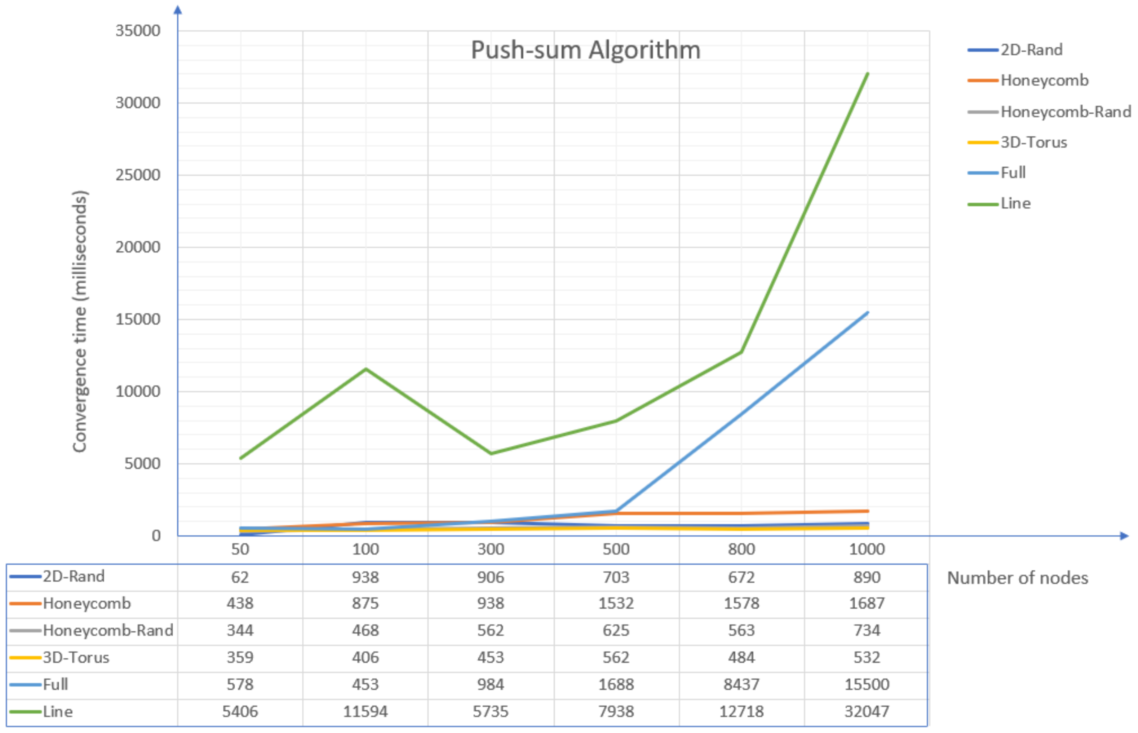
Push-sum Algorithm observations

1. All nodes converged for all the topologies.
2. 2DRand is showing 100% convergence for node count more than 300 but failing to obtain 100% convergence for nodes below 300.
3. The above scenario is happening because while creating nodes, they were randomly created on a 2D grid and if they did not lie within 0.1 distance, they were not connected with one another. Since there is poor connection between nodes for values below 300, there is less scope for convergence.
4. In our implementation, every actor sends messages periodically to its neighbors from the time it received its first message to the time it is terminated. Convergence is achieved due to this model. It is also observed that the periodic time interval chosen to send messages also affects total convergence time.
5. Ist was difficult to achieve convergence time for line topology as it was failing many times. However, when count value was increased to 10, line topology exhibited consistent convergence.
6. Order of time taken for convergence is as below:
   T(3DTorus) < T(randhoneycomb) < T(2Drand) < T(honeycomb) < T(full) < T(line)



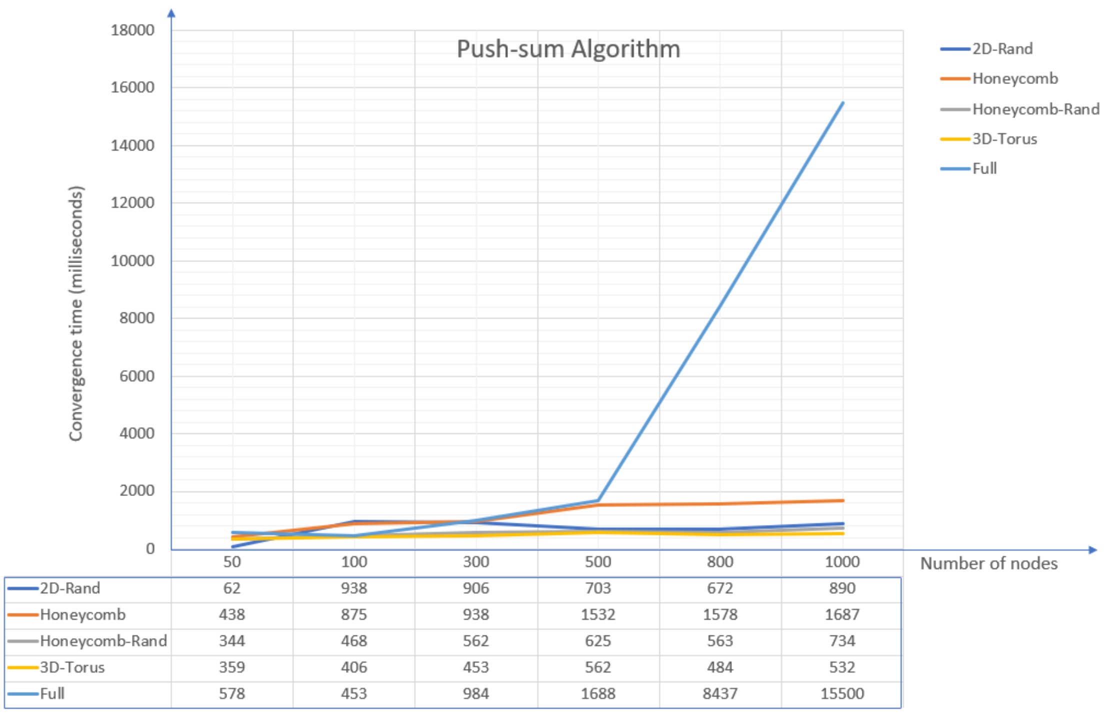
	50	100	300	500	800	1000
2D-Rand	62	938	906	703	672	890
Honeycomb	438	875	938	1532	1578	1687
Honeycomb-Rand	344	468	562	625	563	734
3D-Torus	359	406	453	562	484	532

Figure:4. Above topologies display similar convergence rate.



	50	100	300	500	800	1000
2D-Rand	62	938	906	703	672	890
Honeycomb	438	875	938	1532	1578	1687
Honeycomb-Rand	344	468	562	625	563	734
3D-Torus	359	406	453	562	484	532
Full	578	453	984	1688	8437	15500
Line	5406	11594	5735	7938	12718	32047

Figure:5. Line topology displays inconsistent convergence rate, and has the worst performance



	50	100	300	500	800	1000
2D-Rand	62	938	906	703	672	890
Honeycomb	438	875	938	1532	1578	1687
Honeycomb-Rand	344	468	562	625	563	734
3D-Torus	359	406	453	562	484	532
Full	578	453	984	1688	8437	15500

Figure:6. Full topology displays consistent 100% convergence rate, but takes more time than other topologies.

Full topology – Push Sum

Number of Nodes	Convergence Time (milliseconds)
50	578
100	453
300	984
500	1688
800	8437
1000	15500

Line topology – Push Sum

Number of Nodes	Convergence Time (milliseconds)
50	5406
100	11594
300	5735
500	7938
800	12718
1000	32047

2D-Rand topology – Push Sum

Number of Nodes	Convergence Time (milliseconds)
50	62
100	938
300	906
500	703
800	672
1000	890

Honeycomb topology – Push Sum

Number of Nodes*	Convergence Time (milliseconds)
49	438
121	875
289	938
529	1532
841	1578
1089	1687

Honeycomb-Rand topology – Push Sum

Number of Nodes*	Convergence Time (milliseconds)
50	344
122	468
290	562
530	625
842	563
1090	734

3D-Torus topology – Push Sum

Number of Nodes*	Convergence Time (milliseconds)
64	359
125	406
343	453
512	562
729	484
1000	532