COP5615: Distributed Operating Systems

Project -2

Submitted by:

Varsha Ranka UFID: 5651 – 7308

Harika Bhavya Mulumudi UFID: 3814 – 3105

Implementation details:

Gossip Protocol:

In our implementation of the gossip protocol algorithm, a node will stop transmitting once it has heard the rumor 10 times. Firstly, to start the Gossip protocol, a node is chosen at random from total nodes. Then the node transmits the message to a random node selected. Each time a node receives the gossip, count is incremented, which determines number of times each node received the message. Once the count reaches 10, the 11^{th} time the node converges. Convergence occurs if all the nodes heard the rumor at least once. We maintain a table to note the number of nodes converged.

Push Sum Algorithm:

In our implementation of Push sum algorithm, an actor exits if the s/w ratio did not change more than 10^{-10} in 3 consecutive rounds. Each actor maintains two quantities: s and w. Initially, $s = x_i = i$ and $w = 1$. Firstly, to start the Push-Sum algorithm, actor is chosen at random from total actors. Then the node transmits the message to a random node selected from the adjacency list of each actor that is maintained based on the chosen topology. Messages sent and received are pairs of

the form (s, w). When sending a message to another actor, half of s and w is kept by the sending actor and half is placed in the message. Convergence occurs when all the actors are terminated.

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 the degree of each actor is 6.

Honeycomb: Actors are arranged in the 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.

Interesting findings:

For gossip:

From below convergence tables, for Random Honeycomb, the convergence time is much less because, apart from its neighbors, it also spreads the rumor to one extra randomly picked node. This way the rumor is spread to all the nodes quickly. The convergence condition, i.e. all nodes should hear the rumor at least once, will be achieved faster than other topologies.

The next best convergence time is for 3DTorus network, because, the spread is huge, and it has 6 neighbors across the network. Technically it should be highest for Full network, because the adjacency list has all the nodes except the node that it is transmitting. Since we are picking the node randomly, there is a possibility that the nodes are picked in a line fashion which aren't spread across the network. Hence the convergence time varies.

The convergence times are in the following fashion:

RandomHoneycomb< 3DTorus < HoneyComb<Random 2D< Full < Line

The convergence time for line is the highest because each node has 2 neighbors and it takes time to spread the rumor through those 2 nodes for the entire network.

Also, we noticed that if the stopping condition (i.e. the node stops transmitting once it has heard the rumor 10 times) value is increased, the convergence time also increases because the node which already received the gossip message might receive it again and again thereby reducing the chances of other nodes which haven't heard the rumor at least once

The converge times for **Gossip** are as follows:

Number of Nodes	Full
10	3
100	10
500	177
1000	631
1500	1684
2000	2902
5000	30428
6000	72689
8000	142551

Number of Nodes	Line
10	4
100	493
500	8515
600	15255
700	45412
800	28792
900	38783

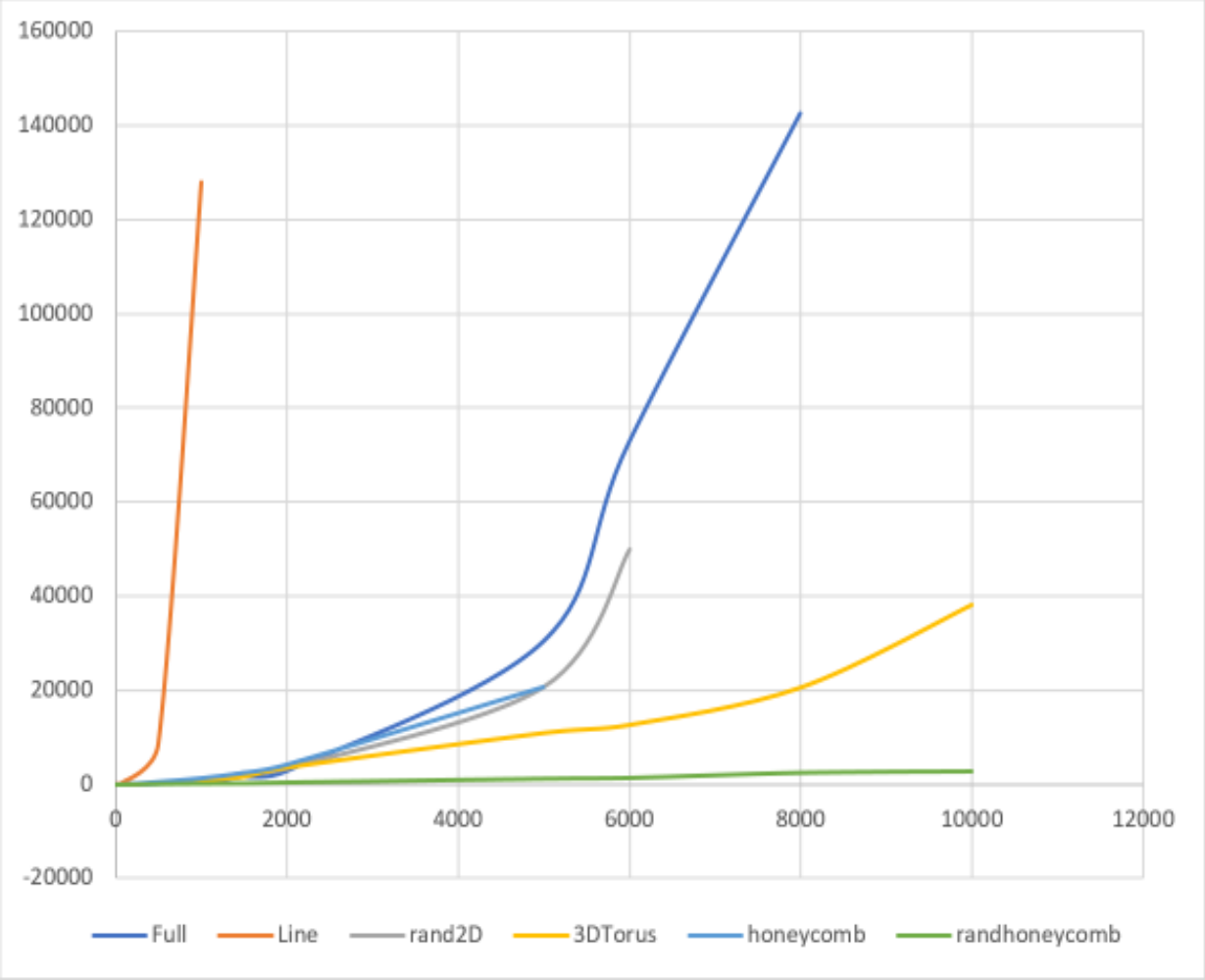
Number of Nodes	Random 2D
10	8
100	70
500	344
1000	884
1500	2528
2000	3499
5000	20795

Number of Nodes	3D Torus
10	3
100	39
500	401
1000	1053
1500	1656
2000	3617
5000	10933
6000	12642
8000	20558
10000	38082

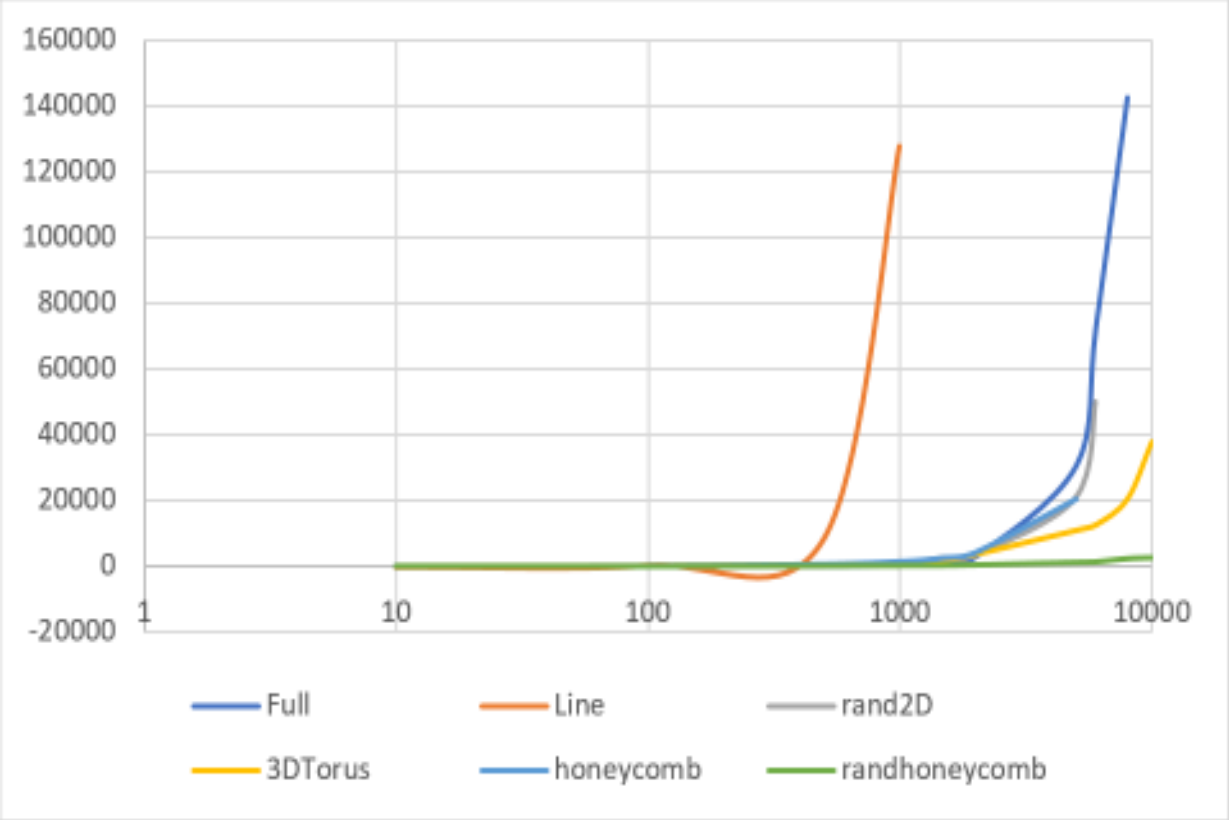
Number of Nodes	Honeycomb
10	8
100	40
500	576
1000	1356
1500	2490
2000	4217
5000	20674

Number of Nodes	Random Honeycomb
10	2
100	26
500	130
1000	205
1500	257
2000	443
5000	1198
6000	1367
8000	2400
10000	2688

Convergence Time of Gossip Protocol:



Convergence Time of Gossip Protocol with log of number of nodes:



For Push-Sum:

Similar to gossip, the convergence time for push sum is less for Random Honeycomb and more for line for the same similar reasons. In push sum, the maximum number of nodes for which the algorithm runs reduces when compared to gossip because, the convergence condition is if the s/w ratio did not change more than 10^{-10} in 3 consecutive rounds. This involves a lot more computations than spreading the rumor. Hence, the total number of nodes for which the push sum works is much less than the total number of nodes for which gossip works.

The convergence times are in the following fashion:

RandomHoneycomb< 3DTorus < HoneyComb<Random 2D< Full < Line

The converge times for **Push-Sum** are as follows:

Number of Nodes	Full
10	6
100	212
500	5306
1000	20457
1500	43071
2000	80926

Number of Nodes	Line
10	6
100	1113
500	108091

Number of Nodes	Random 2D
10	5
100	185
500	4867
1000	18333
1500	40356
2000	69931

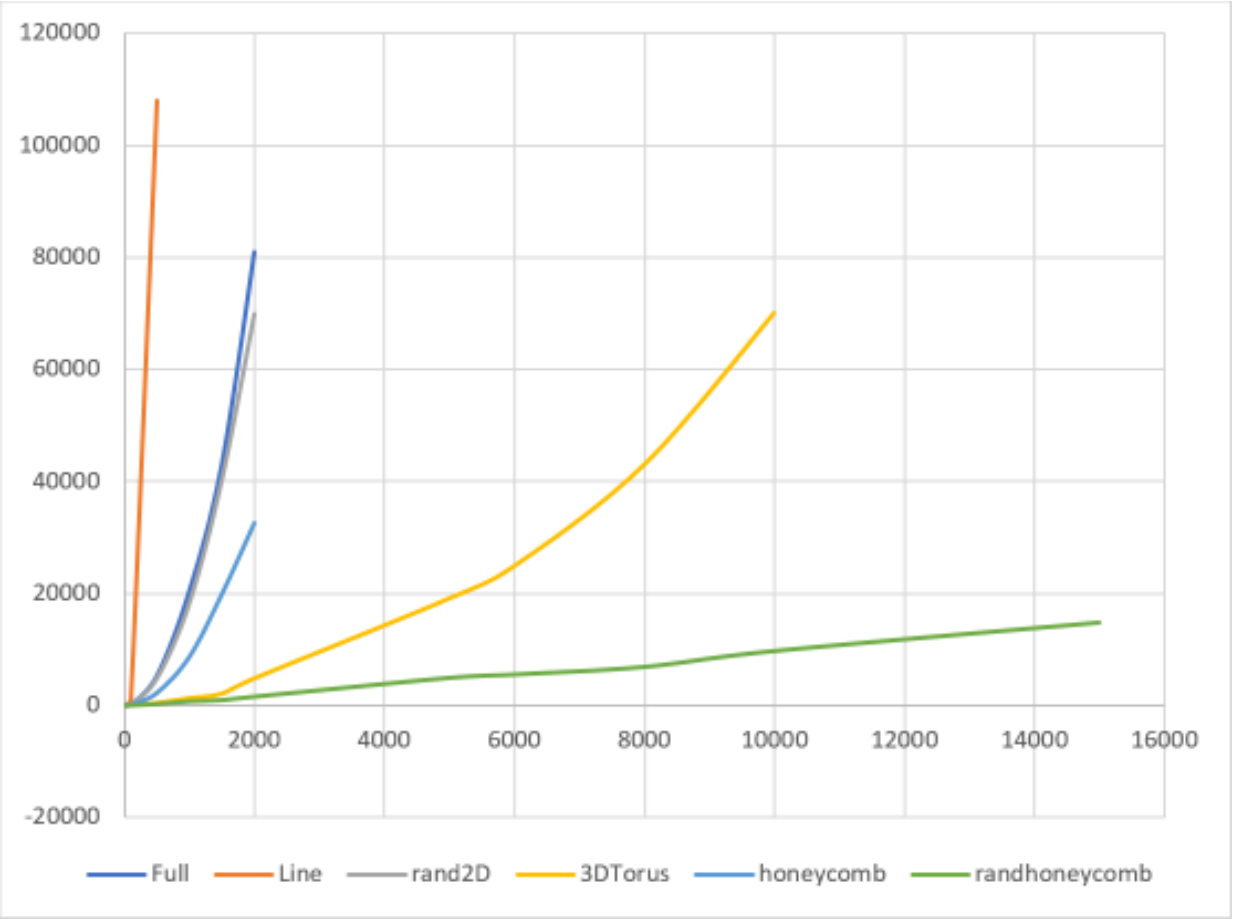
Number of Nodes	3D Torus
10	4
100	67
500	479
1000	1315
1500	2108
2000	4843
5000	19101

6000	24887
8000	42965
10000	70040

Number of Nodes	Honeycomb
10	7
100	105
500	2160
1000	8607
1500	19826
2000	32554

Number of Nodes	Random Honeycomb
10	5
100	43
500	269
1000	741
1500	968
2000	1587
5000	4953
6000	5554
8000	6912
10000	9759
15000	14833

Convergence Time of Push-Sum Algorithm:



Convergence Time of Push-Sum Algorithm with log of number of nodes: