COP5615- Distributed Operating System Principles

Project 2

Developed by –
Shaik, Fayaz - UFID: 6326-9935
Mullapudi, Aseesh - UFID: 9175-1971

1) Aim –

The aim of this project is to implement Gossip and Push-Sum distributed systems algorithms in FSharp using Akka Actor model for different network topologies like “full”, “imp2D”, “2D”, “line”, to analyze their convergence times.

Input: The input provided (as command line to the project2) will be of the form: project2 numNodes topology algorithm
Where numNodes is the number of actors involved (for 2D based topologies we can round up until you get a square), topology is one of full, 2D, line, imp2D, algorithm is one of gossip, push-sum.

Output: Print the amount of time it took to achieve convergence of the given topology and algorithm.

2) How to Run –

Once Dotnet, Akka.Net and Akka.Fsharp are all installed in the Project environment, extract the zip file, and run the program in the following manner:

Command: dotnet fsi “--langversion:preview” proj2.fsx <numberOfNodes> <topology> <algorithm>

Where numberOfNodes (The number of nodes in the network), topology (one among – full, imp2D, 2D, line), algorithm (one among – gossip, push-sum) are the command line arguments.

3) Methodology –

i) Using Akka Actor Model in FSharp, we created each actor as a node in the network. These nodes communicate with each other using message passing between actors.

ii) In case of Gossip algorithm, the initial gossip message is sent to a random node in the network. Then, the algorithm continues until all the nodes in the network receive the message, "threshold" number of times. The threshold in our implementation has been set to 100.

iii) In case of Push-Sum algorithm, the nodes are initialized with initial 's' and 'w' values. Then one random actor or node starts the push-sum algorithm by setting its 's' and 'w' values to half and then sending (s/2, w/2) to a random neighbor. This process continues until one of the actor's ratio of s/w stops changing more than 10^-10 in 3 consecutive messages it receives.

4) Project Questions Answered:

i) *What is working* –

Both the gossip and push-sum algorithm have been implemented for the given topologies – full, imp2D, 2D, line.

Both the algorithms have been tested on various networks by changing number of nodes in the network as well as with different network topologies and have seen the desired results.

Observations –

i. The convergence time order for both gossip and push-sum algorithms followed the same order for different network sizes and different network topologies.
The order is – full < imp2D < 2D < line

ii. The convergence order remained the same, however, multiple runs of gossip algorithm on different network sizes and different network topologies yielded different convergence times.

iii. Full network has seen the best convergence times and line network has seen the worst time for convergence in both the algorithms.

iv. Once the Push-Sum algorithm is converged, The S/W ratio is almost equal to the true average of the network
Which is –

$$\frac{(numberOfNodes) \ * (numberOfNodes + 1)}{2 * \text{numberOfNodes}}$$

Which is reduced to –

$$\frac{(numberOfNodes + 1)}{2}$$

ii) <u>The largest network that we managed to deal with</u> –

a. <u>Gossip</u> –
   i. FULL – 5000
   ii. Imp2D – 5000
   iii. 2D – 5000
   iv. Line – 5000

b. Push-Sum –
   i. Full – 1000
   ii. Imp2D – 1000
   iii. 2D – 1000
   iv. Line – 1000

5) Results –

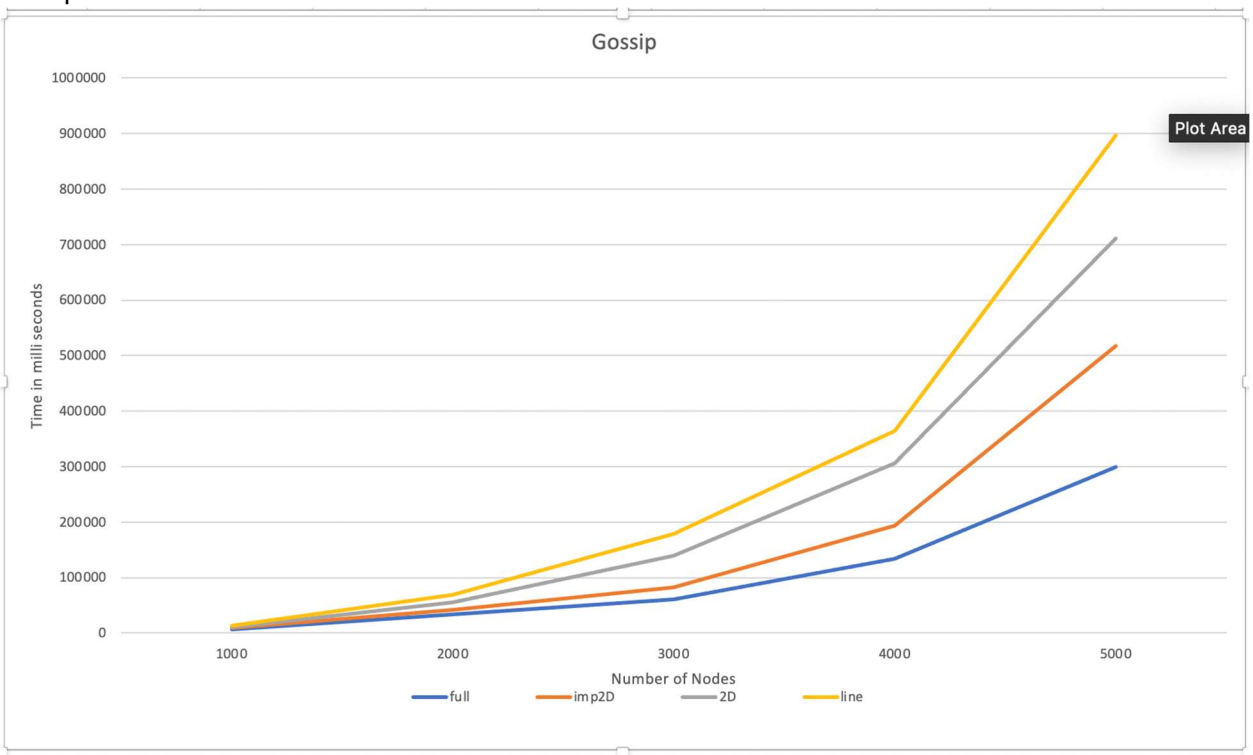
Values for gossip -

num of nodes	full	imp2D	2D	line
50	60.069	69.79	73.684	102.553
100	178.274	169.552	179.28	224.909
500	2012.9	2347.856	2576.752	3607.695
1000	7235.178	9345.666	10507.908	13142.81
2000	33602.285	42496.612	55517.496	68865.312
3000	60926.49	82622.58	139936.025	178125.597
4000	133451.466	193687.84	305964.67	364710.51
5000	299639.667	517010.608	711304.307	897232.431

The time is in milliseconds.

The above values and the graph for gossip algorithm have been drawn with a maximum gossip count of 100, that is, each actor has to receive a rumor 100 times.

Graph -



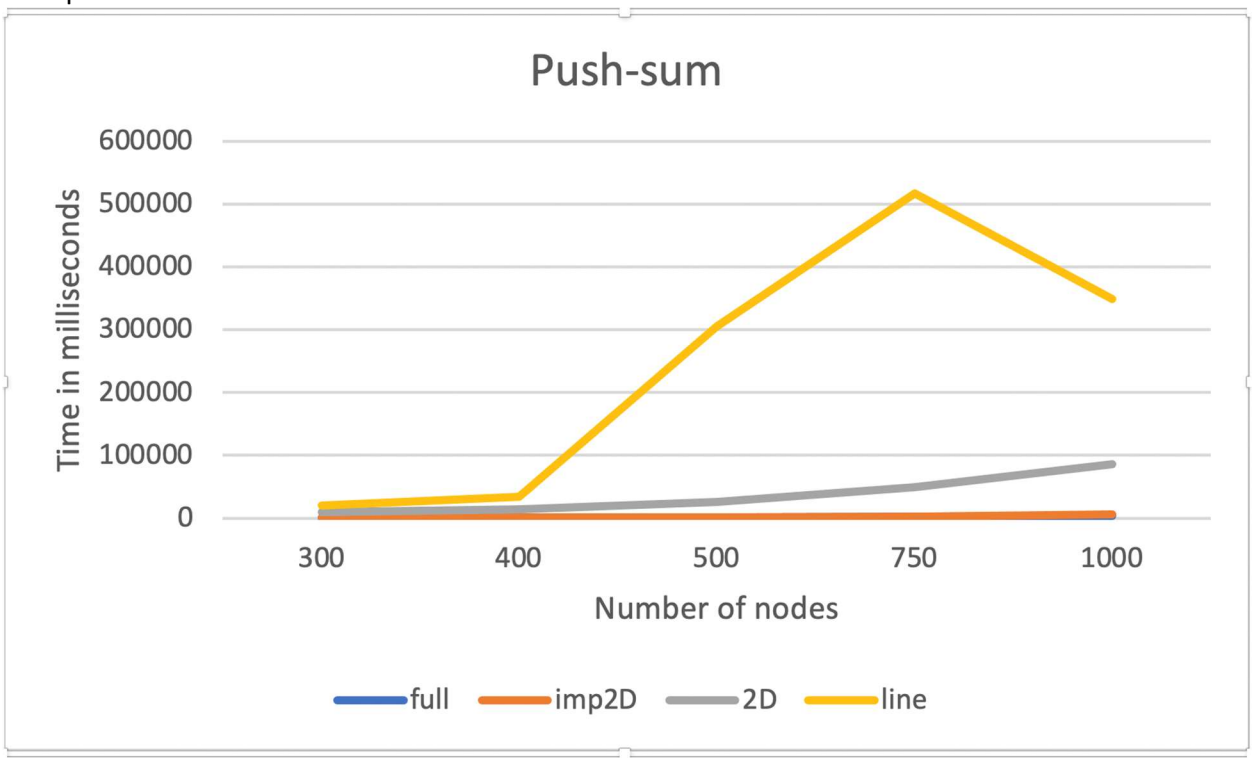
The graph has been plotted for the values in red in the above table.

Values for Push sum -

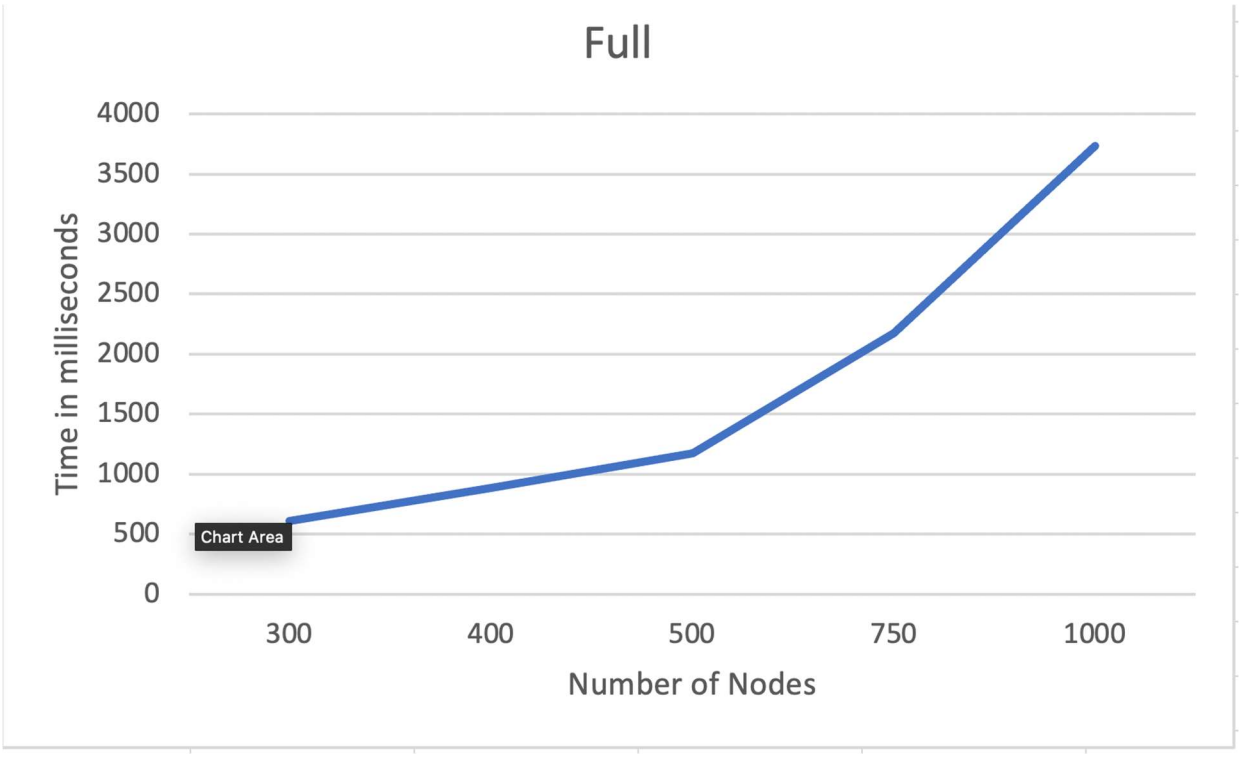
Number of Nodes	Full	imp2D	2D	line
10	21.97	28.71	52.98	98.48
50	89.171	265.421	515.94	3550.44
100	196.41	430.1	1149.647	15384.602
200	467.89	641.863	4040.387	5918.401
300	606.241	1047.713	9138.282	19588.125
400	881.667	1409.35	14280.46	34688.964
500	1171.635	1748.493	25909.451	305796.78
750	2175.42	2783.146	49866.149	517761
1000	3731.404	5995.476	85923.914	349306

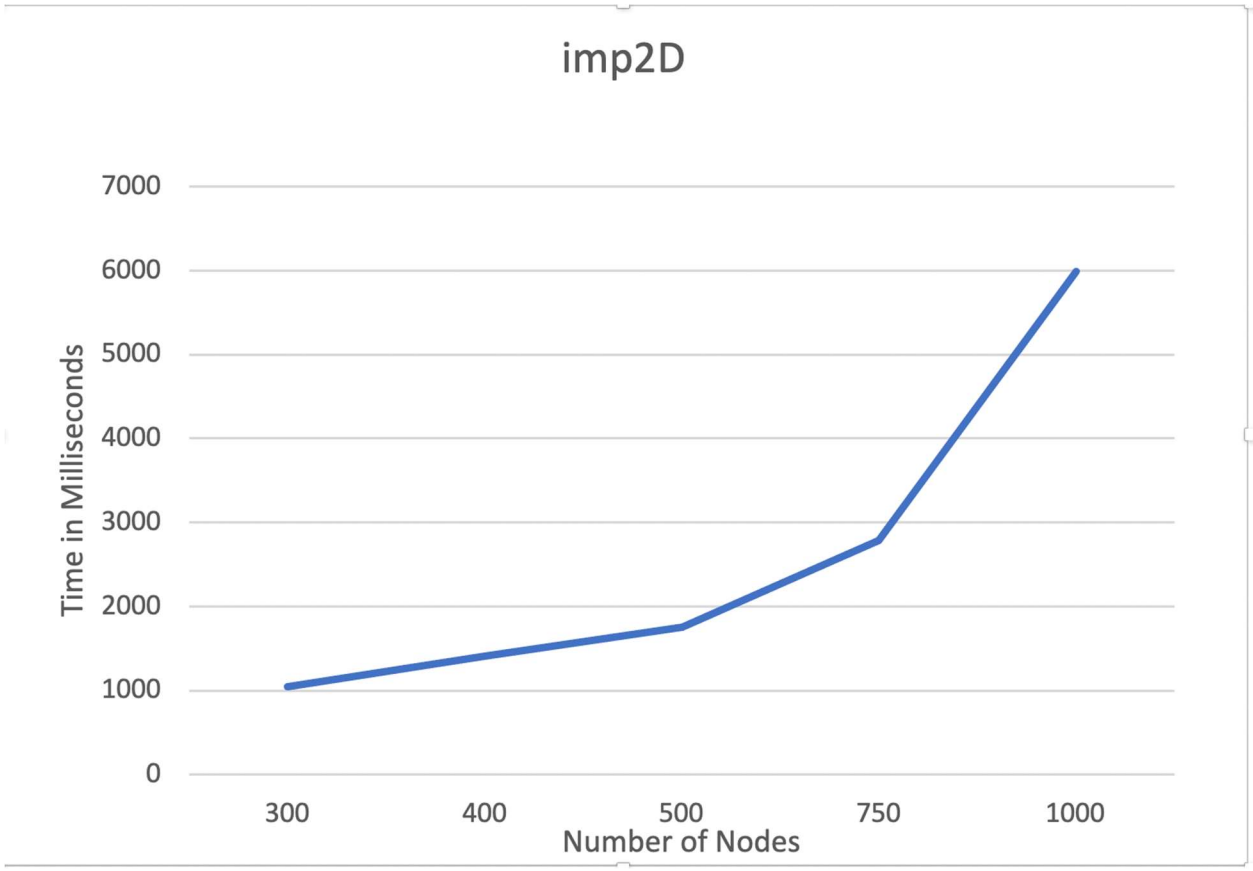
The time is in milliseconds

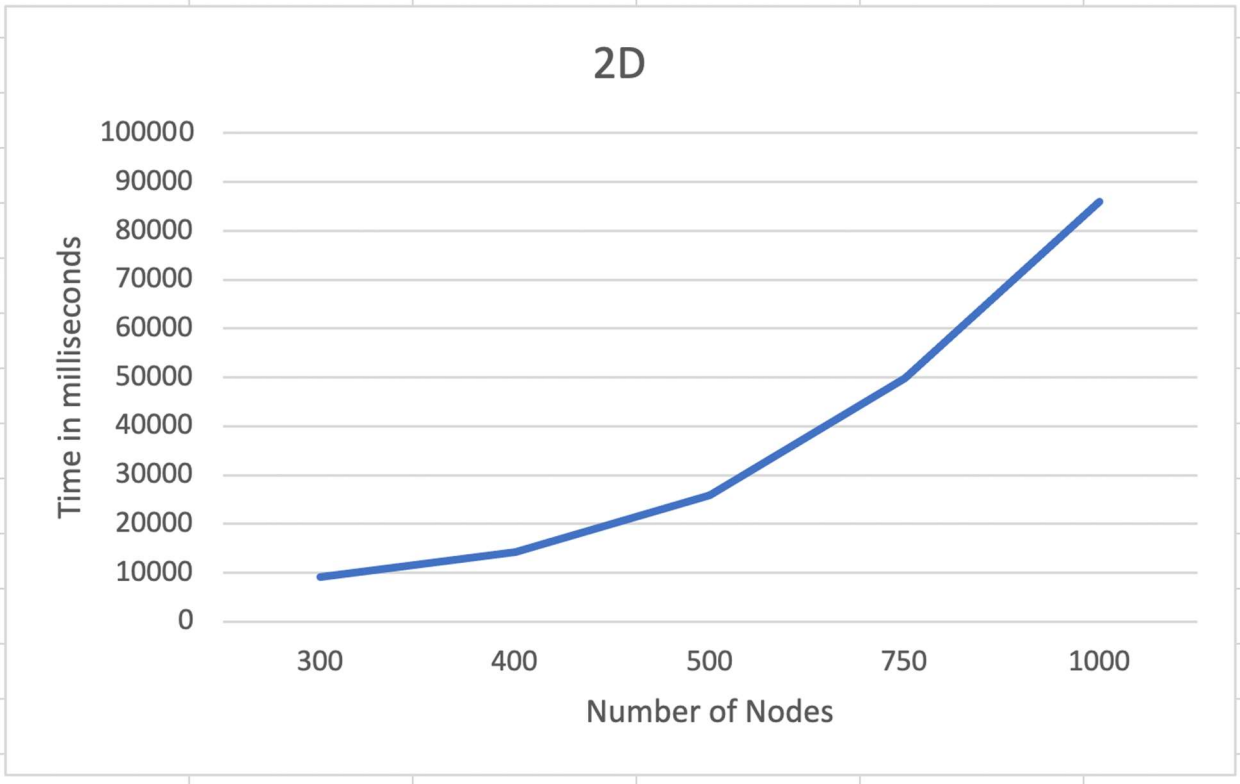
Graphs for Push sum -

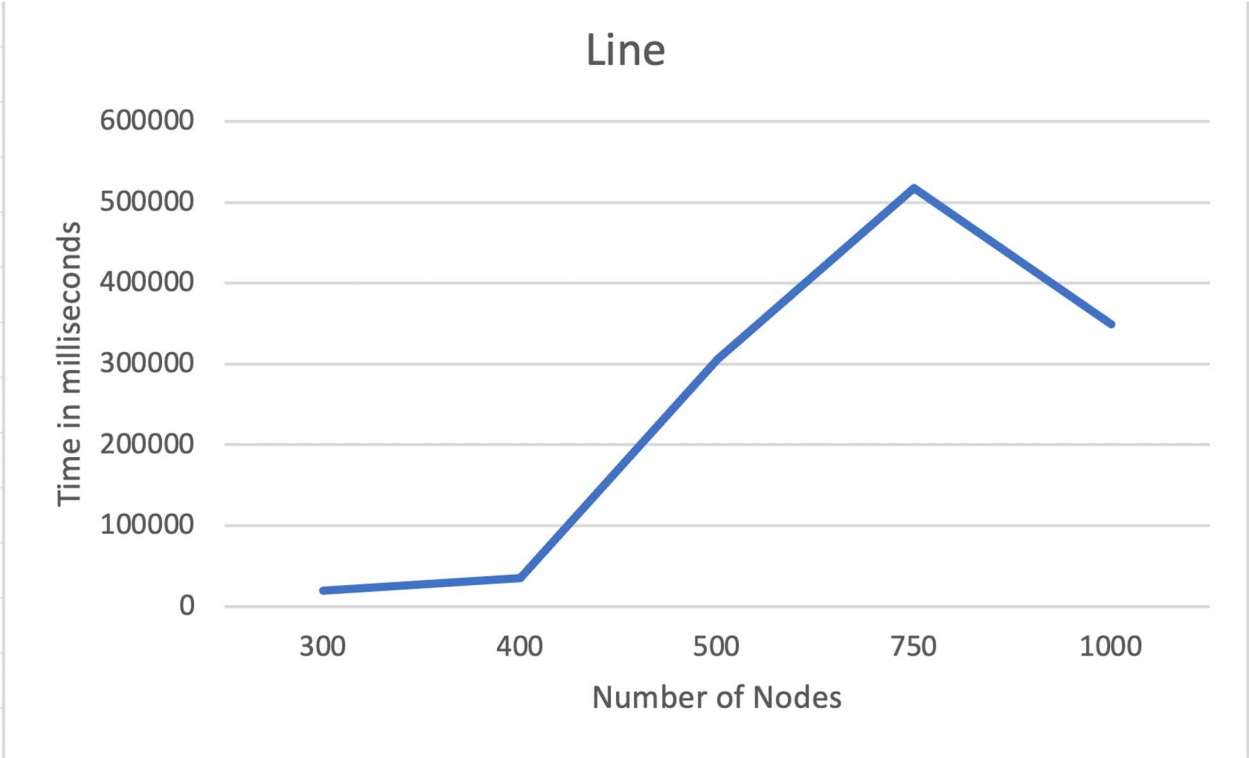


Since the lines for full and imp2D are overlapping, we have separate graphs of all the topologies for a better idea.




imp2D
Time in Milliseconds
7000
6000
5000
4000
3000
2000
1000
0
300
400
500
750
1000
Number of Nodes


2D
Time in milliseconds
100000
90000
80000
70000
60000
50000
40000
30000
20000
10000
0
300
400
500
750
1000
Number of Nodes



A general observation for push sum algorithm using the line topology – the more time it takes to converge, the better the s/w ratio in the end. As you can see the above graph, time taken for 750 nodes to converge is more than the time take for 1000 nodes, the former had a final s/w value nearer to the global average ie (n+1)/2.

Modifying the line topology a bit as in, connecting the last neighbor to the first node (ring) gives a better result in terms of the final s/w value.