Project Report: Distributed Operating System Principles (COP5615)
Project 2: Gossip and Push – Sum Algorithms

Name: Vikas Chaubey**, UFID:** 3511 5826**, Email:** vikas.chaubey@ufl.edu

Group Members: I don't have any group members. I have done this project alone.

Note: Please use following command for program execution if following error occurs (error FS3302: The package management feature requires language version 5.0 use /langversion:preview))

dotnet fsi --langversion:preview project2.fsx 100 line gossip

Project Code, Design and Implementation:
1) f# code implements four different methods which define four topologies - line, full, 2D, imperfect 2D
2) It implements a supervisor Actor which supervises the creation of topologies, creation of child actor nodes, and selection of algorithm (gossip, push sum) as per given input
3) It implements child actor nodes which actually participate in implementation of gossip and push sum protocols and communicate with each other through message channels.

Execution Flow:
1) When program is run, Supervisor actor is created with arguments passed to run the program.
2) As per input given, supervisor actor creates topology, creates child actors and selects the algorithm which has to be run (gossip or push -sum)
3) Supervisor actor chooses one actor randomly from the list of created actors to start the protocol in case of both algorithms
4) Protocol starts running on child actor nodes, when toplogies are formed, a list of neighbor nodes for each and every cluster node is kept in a map, using that list, a node selects a neighbor randomly and sends the message to it.
5) When the execution is completed, converged node numbers are printed and protocol execution time (in milliseconds) is also printed using stopwatch timer.

Problem 1: Gossip Algorithm for information propagation

Description: A participant(actor) it told/sent a rumor(fact) by the main process. Each actor selects a random neighbor and tells it the rumor Each actor keeps track of rumors and how many times it has heard the rumor. It stops transmitting once it has heard the rumor 10 times.

Implementation and Observations: The program execution was done for up to 10000 nodes for each topology.

When the gossip algorithm is implemented as it is described in the problem statement i.e. each node stops transmitting rumor to neighbors once it has heard the rumor 10 times then following observations were made:

1) Line Topology (In general each node has two neighbors) : When the Gossip algorithm is run with Line topology it converges in the case of small number of nodes for example 10, 20 , 30 etc. but in most cases especially with higher number of nodes, line topology does not converge. That means in most cases topology has blind spots where many actors do not receive the rumor from other nodes, hence line topology does not converge.

Reason for line topology to not converge in many cases : since the rumor is started by a random node and on each reception of rumor the node selects a node randomly from its neighbors , there are high chances that if randomly chosen neighbor is same till node is transmitting messages then other neighbor does not receive the rumor and in cluster if this happens with many nodes, it results into blind spots of actors which do not receive rumor. When node was made to send a message to each of its neighbors at least once the line topology converges in most cases. That signifies that in case of line topology formation of blind spots is happening with original approach.

Solution for convergence of Line Topology: In order to achieve convergence in line topology blind spots have to be avoided hence it is necessary that nodes keep on transmitting messages to neighborhood nodes so that each node has heard the rumor at least once. With that implementation convergence was achieved in line topology even for high number of nodes.

2) Full Topology (Each node can send message to every other node in cluster): In most cases full topology converges successfully, and each node hears the rumor 10 times in most cases. since in Full topology each node has every other node of the cluster as neighbor, it can transmit message to every other node. So even though neighbors are selected for transmission of message randomly there are very high chances that each node receives the rumor at least once. Hence full topology converges in most program runs even for high number of nodes. Full topology runs faster than line topology and converges faster than line topology.

3) 2D - topology (Each node has four neighbors in general) : In 2D grid each node has either three neighbors(edge nodes) or four neighbors(center nodes) , hence topology succeeds to converge in most cases because each node can transmit the message to one of the four or three possible neighbors , unlike line topology where a node has only two neighbors. There is one very important observation - 2D topology performs better than full topology in terms of time convergence even though. full topology connects one node from every other node in cluster, the probable reason is since in 2D topology nodes are arranged more symmetrically and neighbors of nodes are arranged around the node in such a way that their location helps spreading of rumor in all the directions in the 2-dimensional grid. Hence the rumor is spread more efficiently. Also, in full topology the cluster is flooded with messages which increases the load on cluster because one node might receive lot of rumors from other nodes at the same time. Hence the load introduces a delay in processing time in case of full topology whereas in case 2D topology, rumor is spread efficiently in all directions which helps in coverage of whole cluster with ease.

4) Imperfect 2D Topology (Same as 2D topology grid with one additional random neighbor, (4+1) neighbors): Imperfect 2D topology functions in same fashion as 2D topology. Hence topology succeeds to converge in most cases. But in case of gossip, since imperfect 2D topology has got an extra neighbor, it can spread the rumor faster compared to 2D topology. Hence when imperfect topology is used for gossip protocol, it performs better than 2D topology. It was observed that imperfect 2D topology is the fastest in terms of convergence among line topology, full topology and 2D topology.

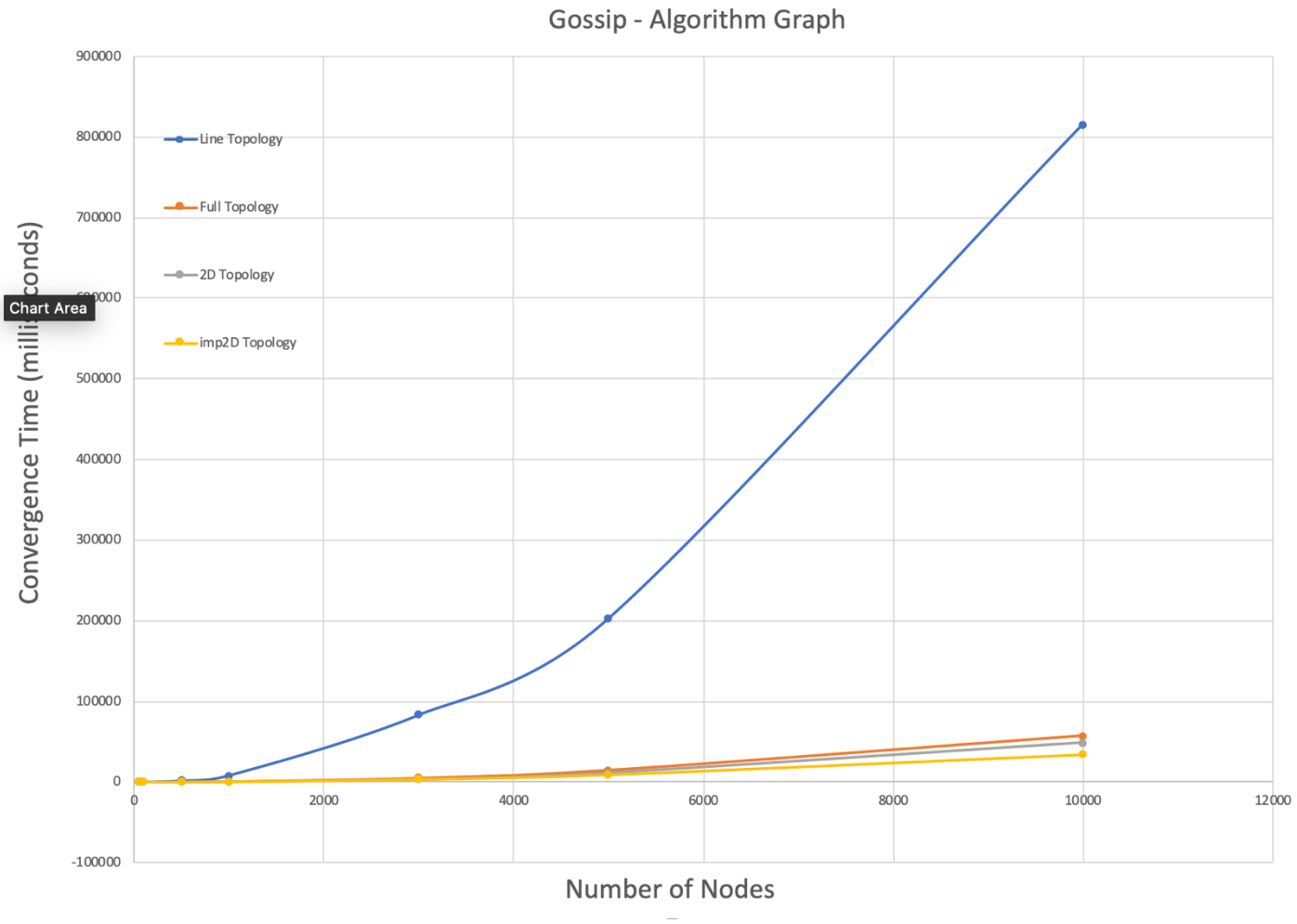
What is Working in case of gossip algorithm?
After implementing above mentioned measures for mentioned challenges, all four toplogies are converging and each node in the cluster is able to receive the rumor 10 times. Convergence of all the toplogies was tested successfully for up to 10000 nodes.

Gossip protocol execution time order for different topologies, for up to 10000 nodes.

Line Topology > Full Topology > 2D Topology > Imperfect 2D Topology

Gossip Algorithm- Graph: (Execution time vs Number of nodes)

The program was run on a 2.3 GHz 8-Core Intel Core i9, 16 Gb Ram Machine.



Execution Timetable:

		Gossip Algorithm		
	Line Topology	Full Topology	2D Topology	imp2D Topology
Number of Nodes	Convergence Time(Milliseconds)	Convergence Time(Milliseconds)	Convergence Time(Milliseconds)	Convergence Time(Milliseconds)
10	17	13	15	15
50	32	20	24	22
100	98	42	36	33
500	1800	201	199	178
1000	8215	675	694	559
3000	83235	5055	3472	3426
5000	202684	14802	11699	9173
10000	815264	57362	48330	34129

Problem 2: Push sum Algorithm for sum computation and convergence

Description: Each actor A(i) maintains two quantities: s and w. Initially, s = x(i) = i (that is actor number i has value i, play with other distribution desired) and w = 1. Ask one of the actors to start from the main process. Messages sent and received are pairs of the form (s, w). Upon receive, an actor should add received pair to its own corresponding values. Upon receive, each actor selects a random neighbor and sends it a message.
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. At any given moment of time, the sum estimate is s/w ratio where s and w are the current values of an actor. If an actor's ratio s w did not change more than 10−10 in 3 consecutive rounds the actor terminates.

Challenges faced in implementation and their solution:

Challenge. 1: The main challenge faced in order to implement push - sum algorithm is obtaining convergence of the s/w ratio in all the nodes of the cluster. It was challenging because as we have seen in case of gossip algorithm topologies like line topology where it can form blind spots where child nodes do not receive the messages from neighbors, if child nodes will not receive the messages from other nodes, node's s/w ratio will not converge .hence line topology is most prone to not converge.

Solution : As we have observed in the gossip algorithm especially in case of line topology in order to make sure that each node receives message from other nodes ,without forming dark or blind spots where nodes are isolated and not part of communication, the nodes should continue transmission of message (s , w) values even after they converge, It is necessary otherwise blind spots will form, s/w ratios in nodes will not converge as a result. So, a node should transmit messages even after it has converged, though it will not process s/w values after convergence, but it will pass those (s, w) values as they were received to neighbors, so that message flow could be maintained among nodes.

Challenge. 2 : The second challenge was to implement the condition of convergence itself, which is s/w ratio difference in each node for consecutive three turns should be less than 0.0000000001 (10 power(-10)) , for three consecutive turns within the node, then only the node is considered to be converged node. The problem is even though if we resolve the first issue which is related to blind spots. the precision for convergence is so minimal that when the topologies are run for even small clusters like (number of nodes = 50, 100) , the program execution was running for hours without giving any results because precision is very minute , in order to obtain that precision for three consecutive turns , it will require a lot of time. Since we need convergence time output for plotting graphs hence this problem was big.

Solution : Push - sum problem description gives freedom to choose different distribution of s value(in s, w pair) , so as a workaround in order to convergence algorithm for topologies in decent manageable times, Value of s was defined for each node as below (opposed to s = node_id):

s = node_id / 1000000000

This way value of "s" is initialized in decimals , with small values nodes can process data faster and reach the precision point faster, in case when s value was simply taken as node index, the program execution was taking hours and still was not able to converge the s/w ratios in nodes.

What is Working in case of push- sum algorithm?
After implementing above mentioned measures for mentioned challenges, all four toplogies are converging values of s/w ratio among all nodes of cluster within reasonable time. Convergence of all the toplogies was tested successfully for up to 500 nodes.

Implementation and Observations: The program execution was done for 10, 50, 100, 200 and up to 500 nodes for each topology and following observations were made:

1) Line Topology (In general each node has two neighbors): It was observed that if line topology is used to run push sum in its original form (without using preventive measures to avoid blind spots) it never converges, only partial number of nodes were seen to be converged. After implementing continuous message flooding (solution mentioned above to avoid blind spots) It was found that the line topology is not the slowest in terms of convergence of s/w ratio for all nodes. Line topology performs better than 2D and imp2D topologies. It is the case because in case of push - sum the message is not just sent and received but the s, w values in the message are also processed which takes additional time. In current scenario Line topology proves to be second best in terms of time for s/w ratio convergence among all nodes.

2) Full Topology (Each node can send message to every other node in cluster): Since in full topology a node is connected to every other node, hence spread of (s, w) values in the cluster among the child nodes happen fast that is why convergence time for full topology is fastest among all other topologies for convergence of s/w ratio among all nodes.

3) 2D - topology (Each node has four neighbors in general): In 2D grid each node has four neighbors in general, this kind of node arrangement helps the cluster to spread (s, w) value pair message in all directions of grid among all the nodes. In this case 2D topology was observed to be converging slowest among all other topologies. This is happening because in push sum convergence depends on the s, w values processing time as well. In 2D topology each node's neighbors are not sequential hence the value of node index differs widely among the neighbor nodes, which in turn effects the value of s, this disparity is causing more delay in converging the s/w ratios for the given precision. Hence 2D topology is the slowest.

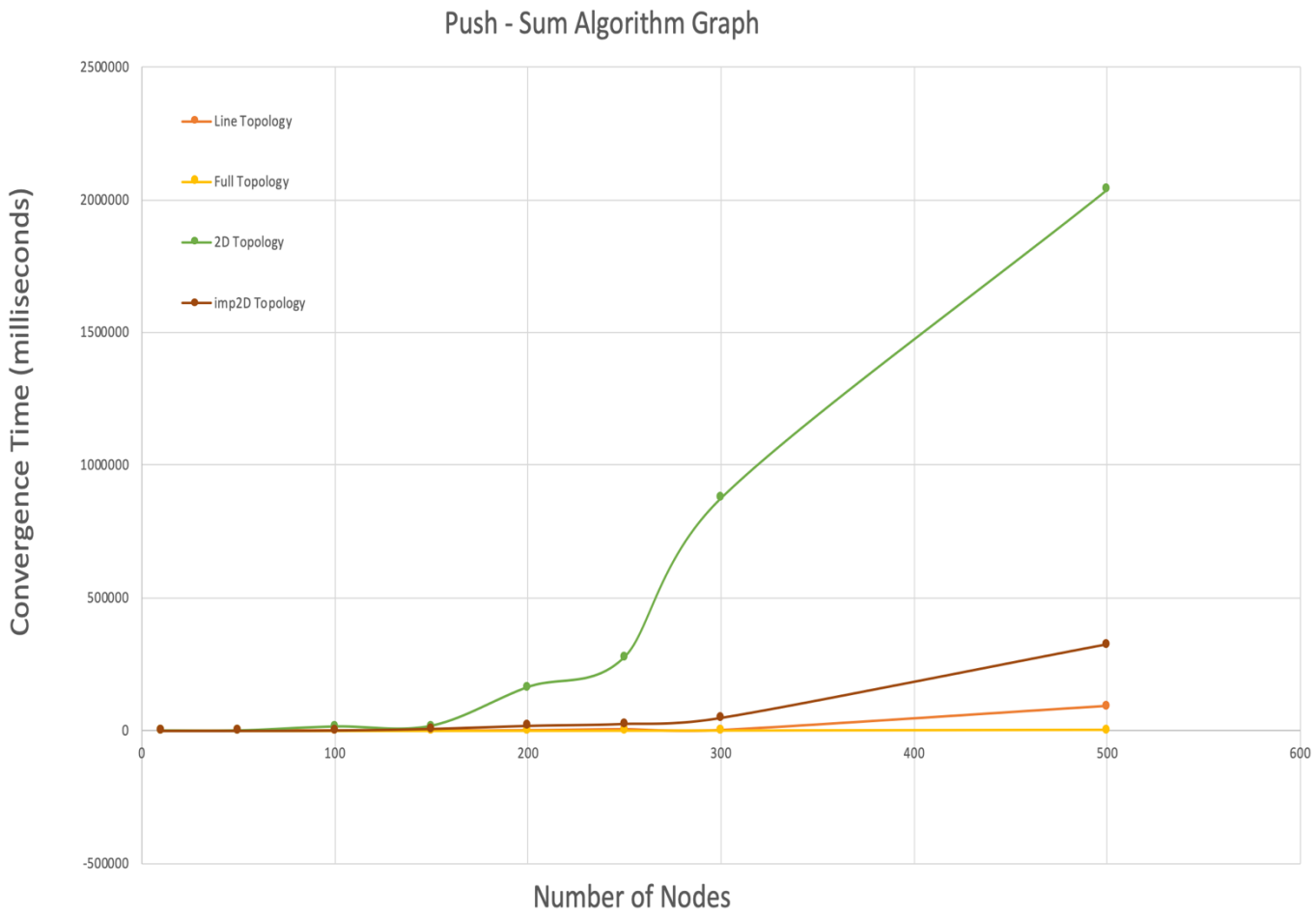
4) Imperfect 2D topology (same as 2D topology grid with one additional random neighbor (4+1) neighbors): Imperfect 2D topology functions in same fashion as 2D topology but with nodes having one extra neighbor in case of imperfect 2D topology. In this case the s/w ratio among the nodes converges faster than 2D topology but slower than line and full topology.

Push-Sum protocol execution time order for different topologies, for up to 500 nodes.

2D Topology > Imperfect 2D Topology > Line Topology > Full Topology

Push- Sum Graph: (Execution time vs Number of nodes)

The program was run on a 2.3 GHz 8-Core Intel Core i9, 16 Gb Ram Machine.



Execution Timetable:

	Push-Sum Algorithm			
	Line Topology	Full Topology	2D Topology	imp2D Topology
Number of Nodes	Convergence Time(Milliseconds)	Convergence Time(Milliseconds)	Convergence Time(Milliseconds)	Convergence Time(Milliseconds)
10	25	21	25	25
50	144	85	702	181
100	1364	189	15885	1692
150	982	368	18572	7758
200	1934	504	164448	19049
250	5759	752	277682	26445
300	2686	1074	876633	49057
500	93436	3085	2040000	325431