COP5615 – Fall 2019
Project 2 – Gossip Simulator

Subham Agrawal | UFID - 79497379
Pranav Puranik | UFID - 72038540

Aim - Implementing Gossip and Push Sum Algorithms in Elixir.

Introduction -

- **Gossip**

Gossip Algorithm is similar to how a gossip or epidemic spreads in the real world. Initially, a node is infected with a gossip. It starts to spread (broadcast) this gossip with its neighboring nodes. They spread it to their neighbors and so on. The nodes stop broadcasting when a gossip is heard 10 times.

In contrast to the centralized system of message transmission gossip protocol is fault tolerant: failure of nodes doesn't affect the propagation of the rumor or the protocol.

- **Push Sum**

An actor receives a tuple {rs,rw} and updates its current {cs,cw} by adding the received pair elementwise. It then halves both the values in new tuple. Next, it stores the half, as well as sends it randomly to one of its neighbors. An actor terminates if the its current s/w ratio is less than 10^(-10) three times. When every node receives the message three times, all nodes are converged, and the algorithm is successful.

Implementation -

- **Gossip**

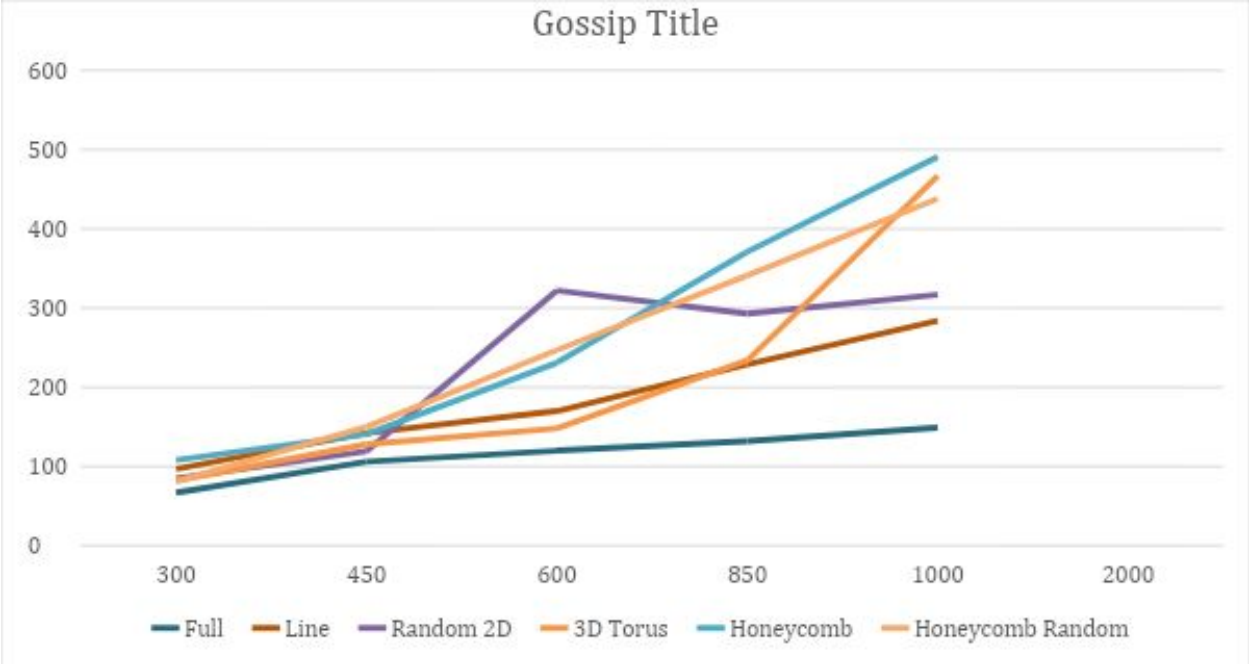
Here, we are creating actors based on the user input number of nodes. We used a decentralized registry while initializing the new actor with its list of neighbors. We then randomly select a node and send it the rumor. This is how the gossip algorithm begins. Once the rumor is heard by an actor, it starts spreading this to its neighbors. Though every actor sends the message until it has listened 10 times, the algorithm converges once when all the actors have heard the rumor at least once.

The maximum possible network capacity for all the topologies for Gossip Algorithm-

Topology	Maximum Nodes	Time for Convergence
Full	10000	4469
Line	10000	3390
Random 2D	2000	946
3D Torus	10000	10422
HoneyComb	10000	14031
HoneyComb Random	8000	10281

Gossip Convergence time vs No of Nodes -

On Linux Ubuntu, intel i5 7th gen, 8 GB Ram.



Data Points for Gossip Algorithm -

No. of Nodes	Topologies					
	Full	**Line**	**Rand 2D**	**3D Torus**	**Honeycomb**	**Random Honeycomb**
50	6	20	-	19	30	20
150	17	45	-	39	39	50
300	67	97	85	82	108	81
450	106	142	119	128	141	150
600	120	170	322	148	231	247
850	132	229	293	234	371	341
1000	149	284	317	467	491	438
2000	267	494	-	776	891	668

Interesting Observations for Gossip –

- In random 2d, the time decreases after about 850 nodes. With more number of nodes connected to each other as nodes are congested,
- Full network is the fastest as there are many nodes connected to each other.
- Honeycomb with Random Neighbor performed slightly better than the simple Honeycomb due to an extra random node.
- Line usually takes the most amount of time, but surprisingly it didn't in our case. We guess this happened because the random function picked the next node (ie the n+1 th node) many times.

(check bonus report interpretations)

- **Push Sum**

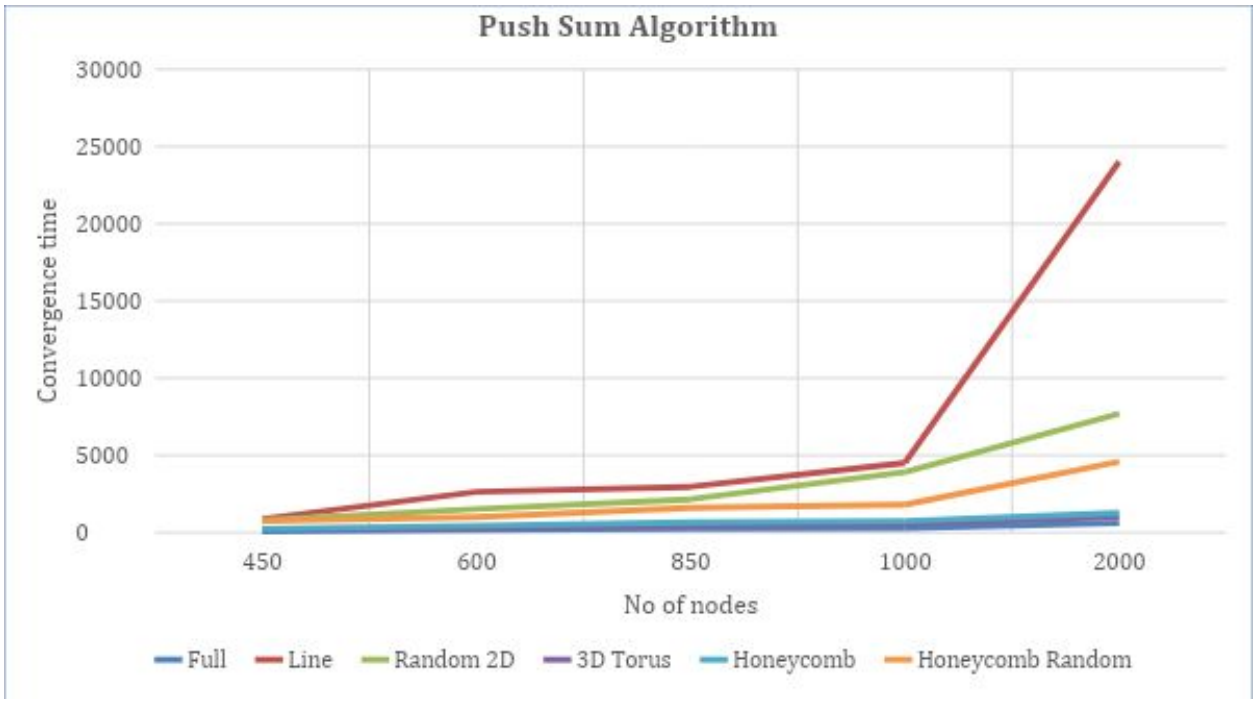
Just like the gossip algorithm, we use the node registry to initialize the new actors with its neighbors. Before sending a message to its neighbor, it first checks if the process is in the registry. If it's there, then message is sent, otherwise, it removes the node from its neighbor list. Neighbors to whom we send the message are selected randomly from neighbor's list. If the change of s/w remains less than 10^(-10) three consecutive times, the node exists and is

marked as converged. We have used ets table for updating the convergence counter in both gossip and push sum algorithms.

The maximum possible network capacity for all the topologies for Push Sum Algorithm-

Topology	Maximum Nodes	Time for Convergence
Full	10000	3344
Line	2000	24032
Random 2D	1500	23203
3D Torus	10000	14844
HoneyComb	10000	28922
HoneyComb Random	8000	27172

Push-Sum (Convergence time vs No of nodes)



Data Points for Push Sum Algorithm -

No. of Nodes	Topologies					
	Full	**Line**	**Rand 2D**	**3D Torus**	**Honeycomb**	**Random Honeycomb**
150	31	297	-	63	94	187
300	63	516	-	187	172	344
450	94	875	791	250	235	750
600	172	2641	1531	275	438	1000
850	235	2967	2157	438	656	1609
1000	297	4516	3921	547	750	1812
2000	610	24032	7688	1000	1265	4593

Observations Push Sum–

- From the push sum algorithm, we can conclude that Line topology is slowest amongst all the topologies. The time it took to converge for the Line topology clearly increases after 1000 nodes. This may be due to fixed neighbors which limits the number of message propagation.
- The order of convergence of the network topologies from the below data point can be stated as:

  T(Full) < T(3Dtorus) < T(honeycombs) < T(rand2D) < T(Line)

  Full is still the fastest.

(check bonus report interpretation section)