Project 2: Gossip Simulator - Bonus

Group Members

- Amit Asish Bhadra (66494087)
- Rishab Aryan Das (05369273)

Additional Data Structures

On top of the data structures defined in the usual Gossip Algorithm, for implementation of Failure model, each actor has 2 more arrays –
alive_status : For each neighbor, this array stores whether the neighbor is alive(1) or if it's blocked temporarily(-1)
num_pings : For each neighbor, this array stores how many consecutive "blocked" pings it receives

Additional Functions

2 additional functions have been implemented to simulate a Failure Model –

Ping(idx): This generates a random number and checks it against a range. It returns 1 if the actor is alive, 0 if the actor is blocked temporarily and -1 if the actor is dead
ReceivePing(idx, status): This is the returned ping from the neighbor actor. Status is the value which determines whether the neighbor is alive, dead or temporarily blocked

Failure Model

Since none of my nodes are dying automatically, I induce failure with the following way –

- During the normal execution of Gossip, all the neighbors are pinged to know the status of the nodes using **Ping(neighbor)**
- The neighbor nodes reply back with their status and their idx using **ReceivePing(idx, status**
- Inside the **ReceivePing** function, the following computations happen –
    - If the status is 1, that means the neighbor node is alive, so set the **alive_status** to 1 and flush any consecutive blocked pings on **num_pings**
    - If the status is 0, that means the neighbor node is temporarily blocked. In such a case increase the **num_pings** for that neighbor by 1. Now check that –
        - If num_pings for that neighbor is more than 5, this means we have more than 5 consecutive pings that the neighbor is blocked. In such a case, we consider that the connection is broken. Hence, remove that neighbor from the node's neighbor list and also remove them from **alive_status** and **num_pings**

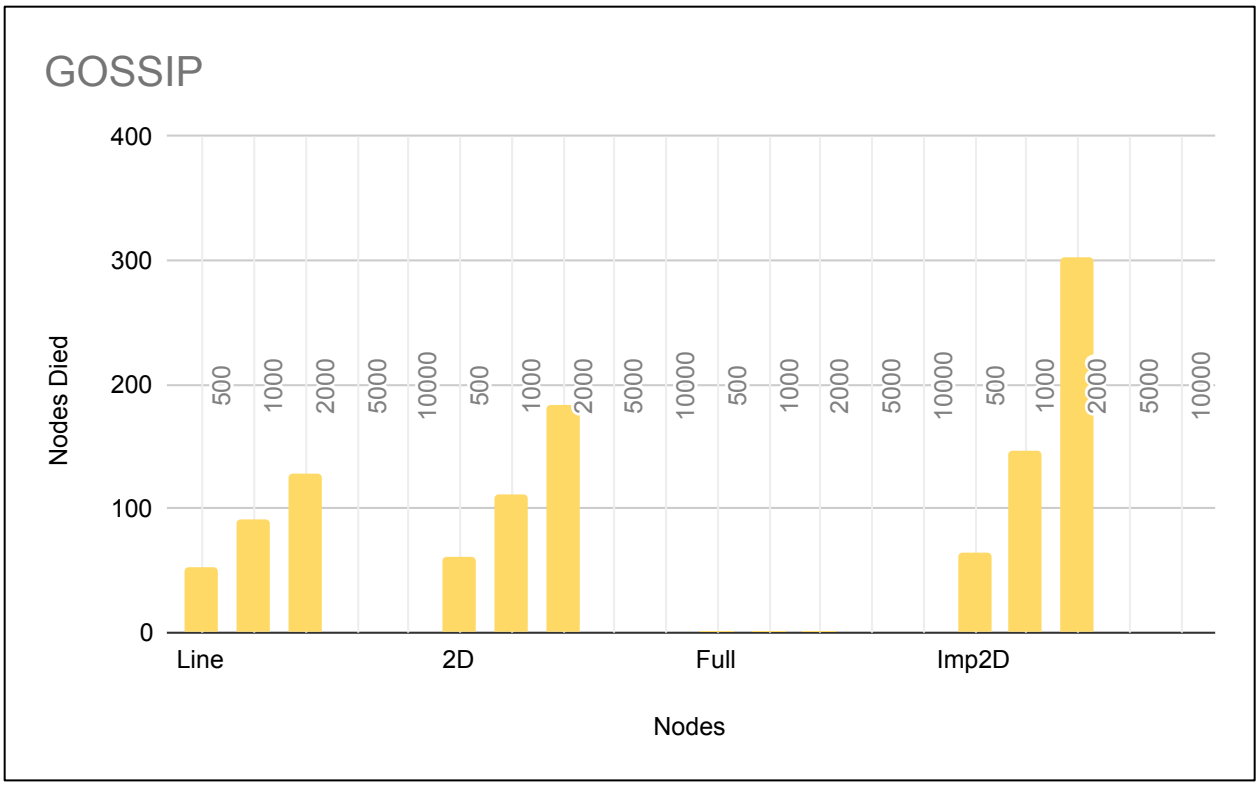
        - If num_pings is less than 5, then do nothing
    - If the status is -1, that means the node is dead. In such a case, we
        - Remove the neighbor from the neighbors array for that node
        - Remove the neighbor from **alive_status** and **num_pings**
        - Converge the node that just died

We choose probability of Alive: Temporarily Blocked: Dead = 497 : 2 : 1 and take the average of few runs and this is the graph that we get –

Algorithm	Topology	Nodes	Connection Dead	Node died
Gossip	Line	500	0	52
		1000	0	92
		2000	0	128
	2D	500	0	61
		1000	0	111
		2000	0	184
	Full	500	0	500
		1000	0	1000
		2000	0	2000
	Imp2D	500	0	65
		1000	0	146
		2000	0	302

Few key observations –

- For Full, all the nodes are dying. This is because of the continuous pings that we send to each neighbor from every single node. If we increase convergence limit and limit to only x amounts of pings every second, then this can be avoided.



Here,
Node Died: The value returned from the **ReceivePing** was a -1 so the neighbor node must forcibly be removed and converged.

Observations:

- We observe a sort of linear graph which shows that as the number of neighbors increase, the number of "**Node Died**" also increases. This is because we are able to send more pings to a larger number of neighbors together so we have a greater chance of receiving dead status. This is observed in the case of Line, 2D and Imp2D.
- As the number of nodes increase, we also observe a sort of linear increase in the number of nodes that have died. This is with relation to the particular topology but is observed in the case of Line, 2D and Imp2D.
- For Full, the number of neighbors is the entire network and in such a case, our Failure Model fails to scale up. The reason for this is that the number of pings sent among each neighbor to neighbor increases so vastly that the system gets dead status even before they are able to converge. Here I have mentioned NA because I noticed that all the neighbors were getting a "**Node died**" status.

The Returned Alive Status

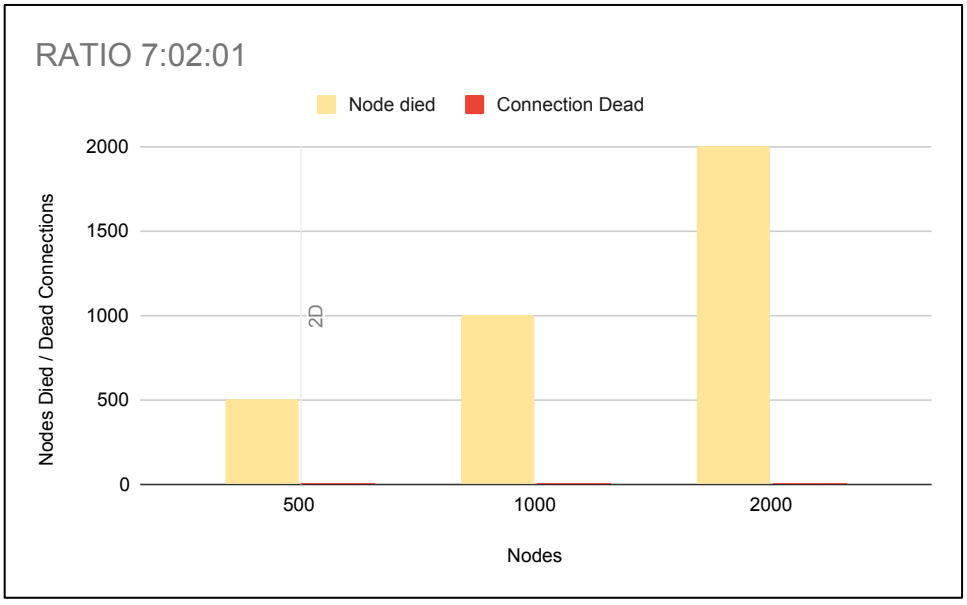
This is very important and it decides how the Failure Model reacts to the topology and the number of nodes.

We choose a random upper index, suppose n and do a RNG from 0 to n. Now, we specify a range –
For eg:

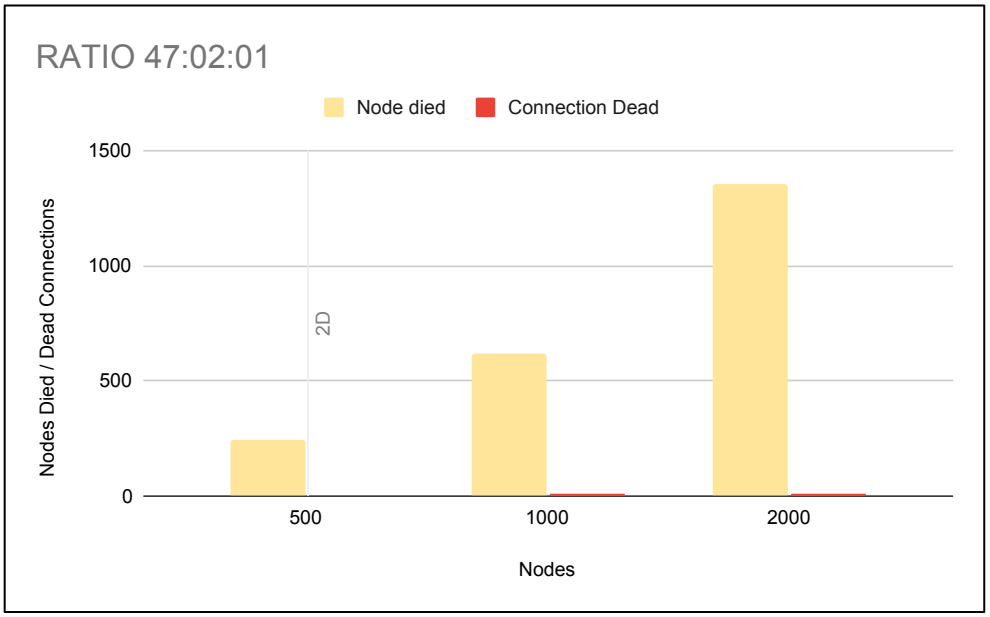
(0, n-3): return alive (1)
(n-2, n-1): return blocked (0)
(n): return dead (-1)

Choosing n and the range of the active status is directly dependent on how many nodes converge normally and how many nodes fail or if connections break.
Keeping the value of converge = 10 constant, and choosing the topology = 2D, we change the range and these are the results:

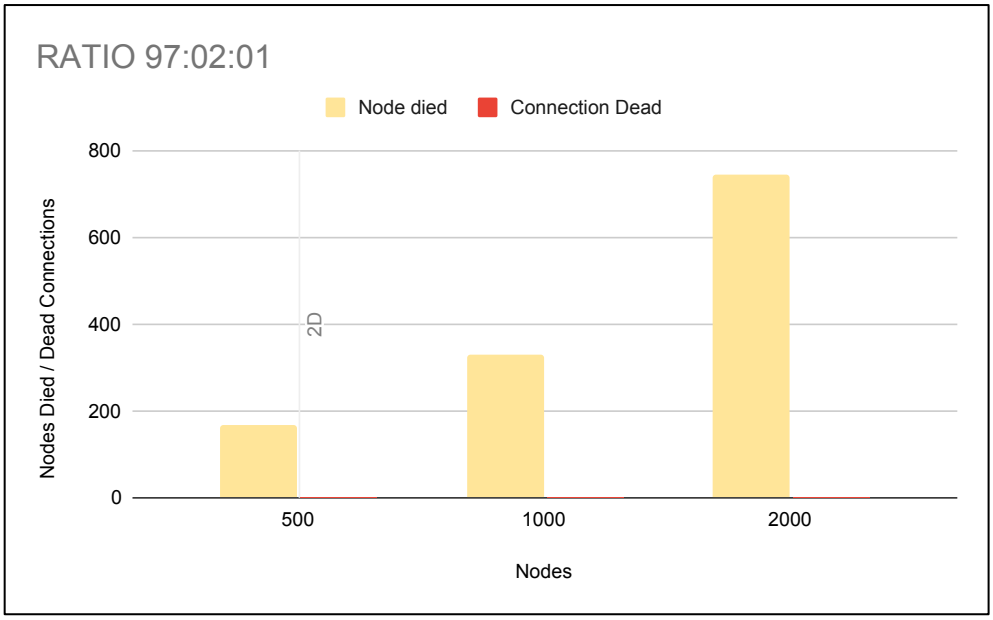
- Alive:Blocked:Dead = 7:2:1



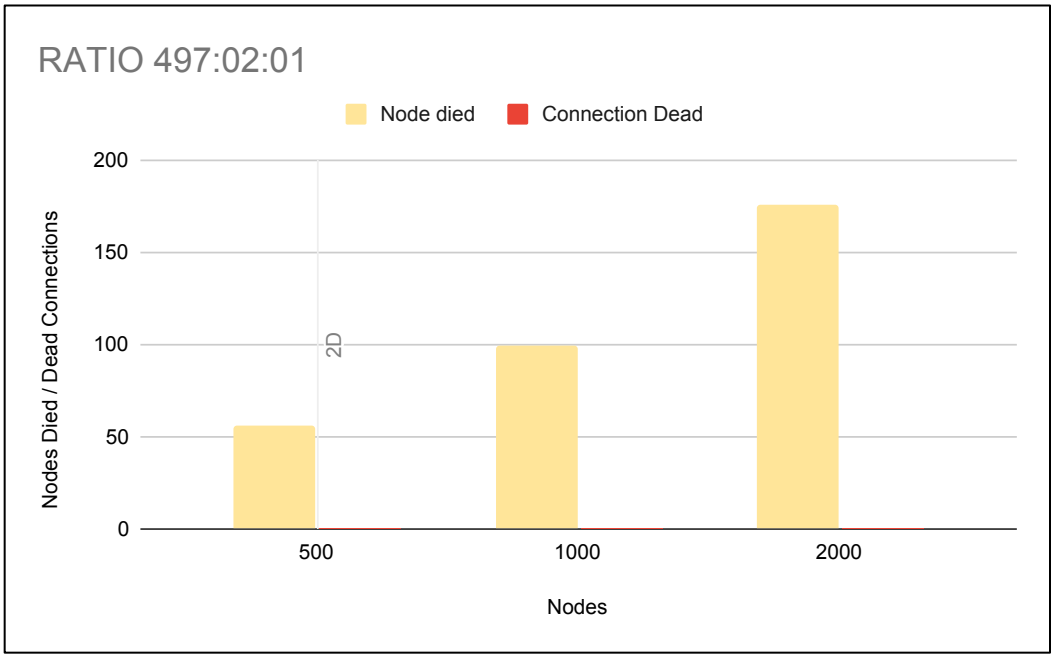
- Alive:Blocked:Dead = 47:2:1



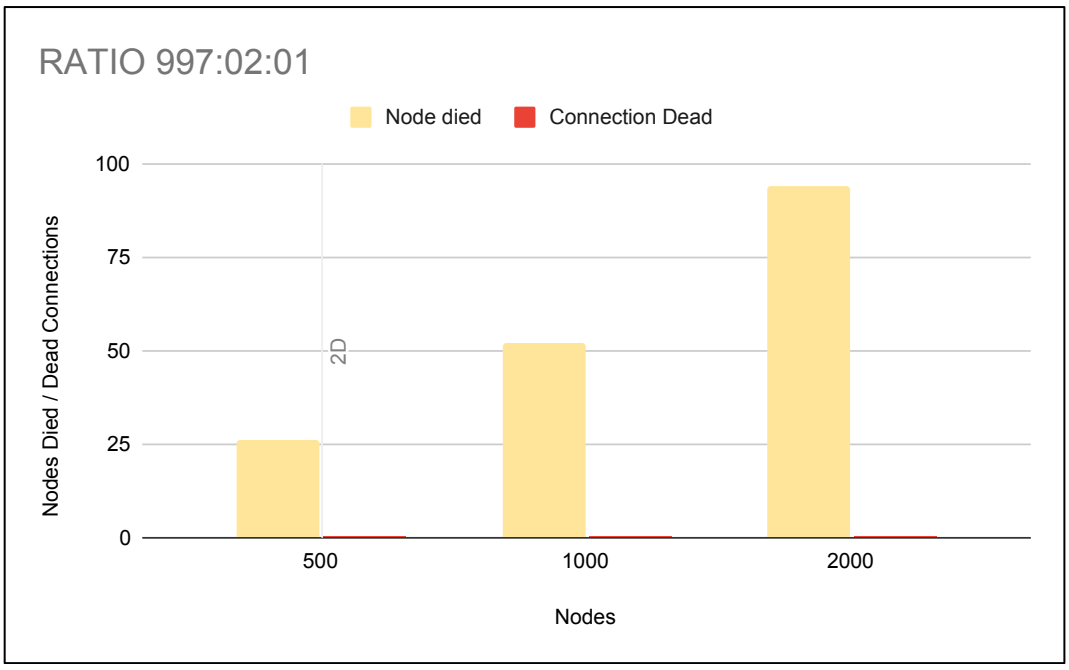
- Alive:Blocked:Dead = 97:2:1



- Alive:Blocked:Dead = 497:2:1



- Alive:Blocked:Dead = 997:2:1



In my tests, I have seen the same pattern repeat for other topologies as well (except Full).

Observations:

- When the ratio of Dead is quite large, it overpowers the network and ultimately kills all the nodes. In full, although the ratio is kept constant, we see this for almost all the number of nodes because of so many pings traveling across and it only takes 1 dead ping to completely kill a node.
- As the ratio of Dead decreases and Blocked increases, we see that there are some ratios, where the connection of the nodes are considered dead but the nodes themselves may not die. This is observed in certain optimal case when Alive is not too high
- As the ratio of Alive increases, it overpowers blocked. As we approach the 3rd ratio, where the sum total is 100, we see that node connections aren't broken, but nodes do die. This is because the chance of getting an Alive message is much higher than getting 5 consecutive blocked messages. Hence, we see nodes dying only due to the Dead ping message.
- As we keep on increasing the ratio of Alive, we see fewer nodes dying from Node failure(or Dead Pings) and more nodes converging normally according to the Gossip Algorithm.
- As more number of nodes die, according to our Gossip implementation, the time it takes to converge decreases because we do not need to wait for these nodes to finish reaching visited=10.