DOS Project 2 - Gossip Protocol - Bonus

Rupayan Das - 63992113
Pirouz Naghavi - 30158432

Getting Started

Requirements

- **.NET 5 SDK on both machines**

Running the project

- *dotnet fsi ProjectPart2_bonus.fsx <numberOfNodes> <topology> <algorithm> <numOfFailedNodes>*
- eg: dotnet fsi ProjectPart2.fsx 100 full gossip

Implementation

For this part of the project, we added a failure mechanism where a certain number of nodes are randomly selected to be removed/failed on purpose. These nodes can be thought of as dead nodes as they can't recieve or send messages to other nodes. We introduce these dead nodes in the network using a command line parameter. It can be anything between zero to the number of nodes previously specified.

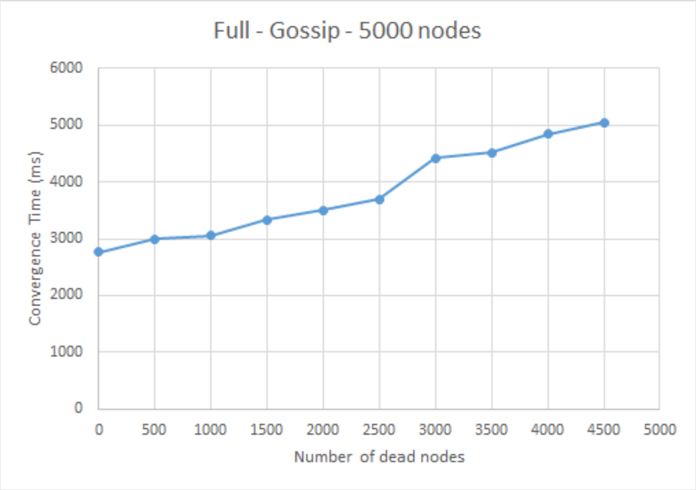
These dead nodes are randomly selected throughout the network. With the exception of the first node which starts the gossip algorithm or the push-sum algorithm all other nodes can be selected as a dead node. This is done so that there is at least one node to start the process, otherwise the algorithm will stop at the first node itself.

With our experiments we aim to find out if the remaining nodes in the system still converge or not. Also, what percentage of the network has to be down to cause the system to not converge. In real life, with distributed systems, it's expected that not all parts of the network will always be functional. We aim to replicate that with our failure model.

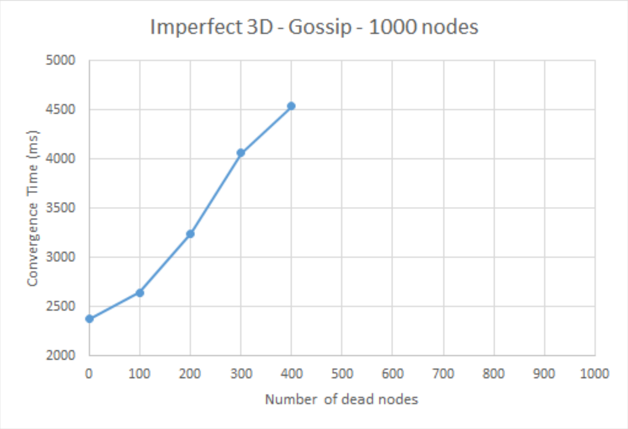
Experiments and Findings

Gossip:

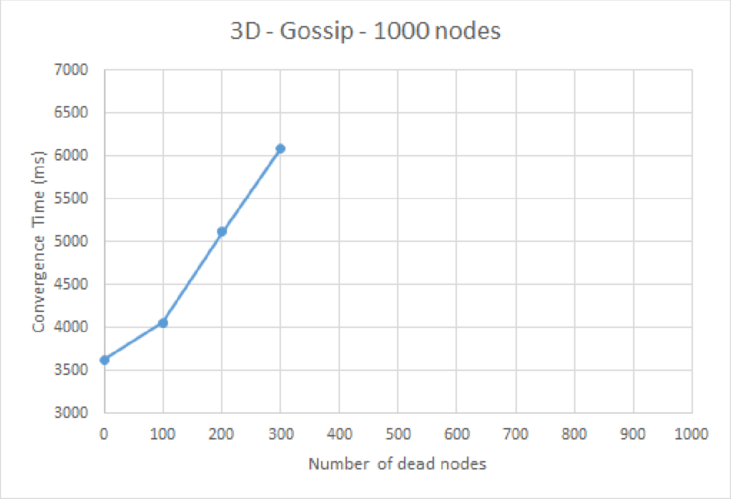
The first experiment we carried out was with the gossip algorithm in full topology. We noticed that as the number of dead nodes increased, the convergence time also increased. It is also evident that even though the number of failed nodes may be 90% of the network, the system still converges.



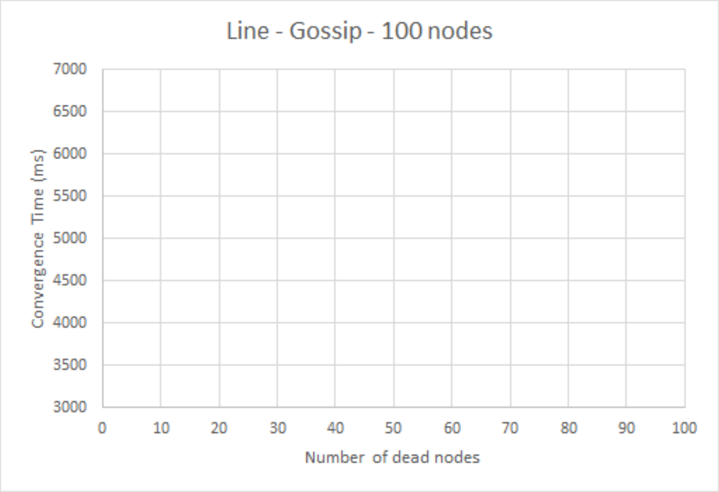
The second experiment was with Imperfect 3D topology for the gossip algorithm. Here we noticed that full network convergence happened only a few times as the number of nodes increased, and after crossing 40% dead nodes, it stopped converging entirely. On deeper inspection we found out that after 40% dead nodes the chances of an actor and it's neighbors all being blocked from receiving messages increases dramatically. Even with an additional random neighbor, a few times the random neighbor itself turned out to be a dead node.



The third experiment was with 3D topology for the gossip algorithm. Here, similar to imperfect 3D convergence time increases as the number of dead nodes increases and after 30% of the network being dead, convergence stops entirely.



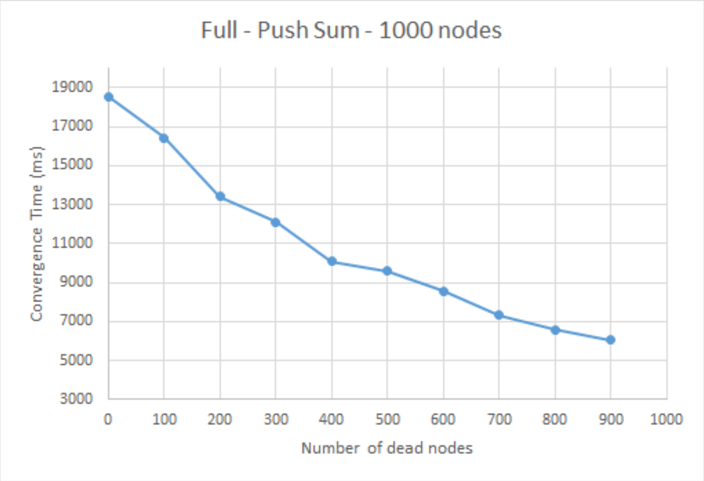
The fourth experiment was with Line topology for the gossip algorithm. With line topology we noticed that even with only 5% of dead nodes, the system doesn't converge. On deeper inspection, it seems that the chances of an alive node being surrounded by two dead nodes as neighbors is high. This causes that alive node to not be able to send or receive any message. Obviously, there were some times that the line topology did converge, due to luck of how the random dead nodes were selected. However, out of the 50 times we ran the experiment for line topology, the convergence rate was less than 3 times.



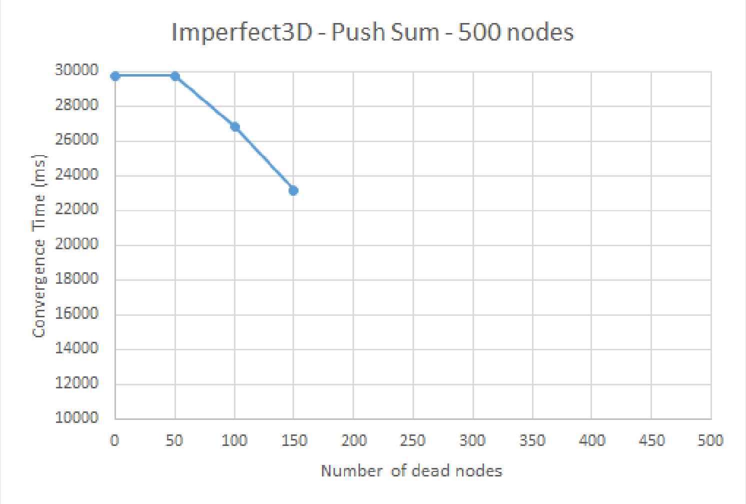
Push-Sum:

We carried out similar experiments with the push sum algorithm for each topology.

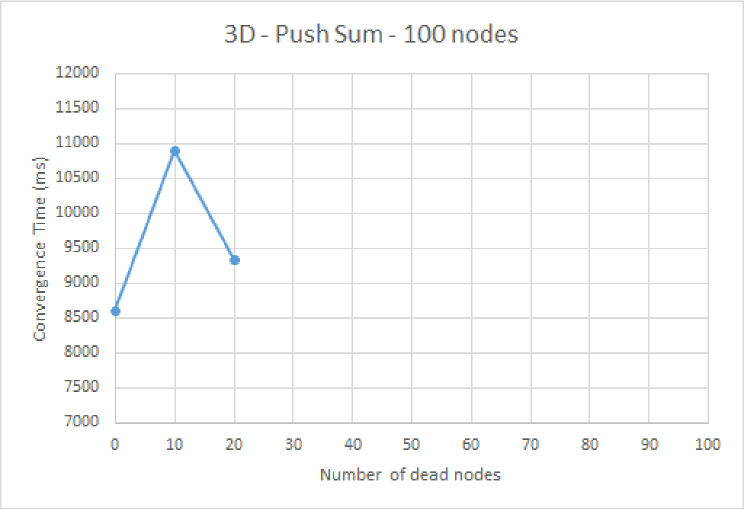
Converse to what happened with gossip, in the case of the push sum algorithm for the full topology, as the number of dead nodes increased, the convergence time for the remaining nodes decreased. It was also noted that unlike before in a completely functional network, here the ratio where the nodes converged would fluctuate a little each time the program was run. It is to be expected since random nodes are removed. Hence, the sum of these nodes aren't added to the final values.



When tested for imperfect 3D topology, the convergence time decreased until there were only 70% nodes remaining. After killing 30% of nodes, the system didn't converge anymore.



As for the 3D topology, convergence stops entirely after more 20% nodes are killed. However, we do see a variance in the convergence time for remaining nodes until then. However, randomness can be attributed for this effect.



Line never converged for any of the test cases we ran. We assume that the same reasons that stopped gossip from converging for line topology apply to the push-sum algorithms as well.