Project 2 - Failure Model
COP5615

Rahul Prabhu
Sanil Sinai Borkar

October 4, 2015

1 Description

The program was run with an additional parameter which indicated the percentage of the network that needs to go down. The following command was run at the command prompt:

```
$ run <num_of_nodes> <topology> <algorithm> <percent_failure>
```

There are 4 command-line arguments as given below:

1. **num_of_nodes**: The number of nodes in the network topology. It is a positive integer value.
2. **topology**: The network topology. It should be one of 'full', 'line', '3D' or 'imp3D'.
3. **algorithm**: The algorithm. It should be either 'gossip' (Gossip Protocol) or 'push-sum' (Push-Sum Algorithm).
4. **percent_failure**: Percent of the total number of nodes that need to fail.

If the number of nodes that need to fail is a real number, it was 'floored' to the nearest integer.

We tested the performance in case of both temporary and permanent failures. Temporary failure was achieved by sending a message to the designated actor asking forcing it to go to sleep for a specific amount of time. Permanent failure was achieved by maintaining the current 'state' of an actor. It was initialized to the “ALIVE” state, and was switched to “DEAD” state to simulate its death. Once “DEAD”, it never came back up.

2 Gossip Protocol

The Gossip Protocol took a lot more time to converge than the normal implementation. Figure 2 shows the percentage coverage a network reached on different failure rates and different topologies.

For each combination of parameters, we ran the program a few times. The average of all the runs was then considered for our analyses and the graph plots.

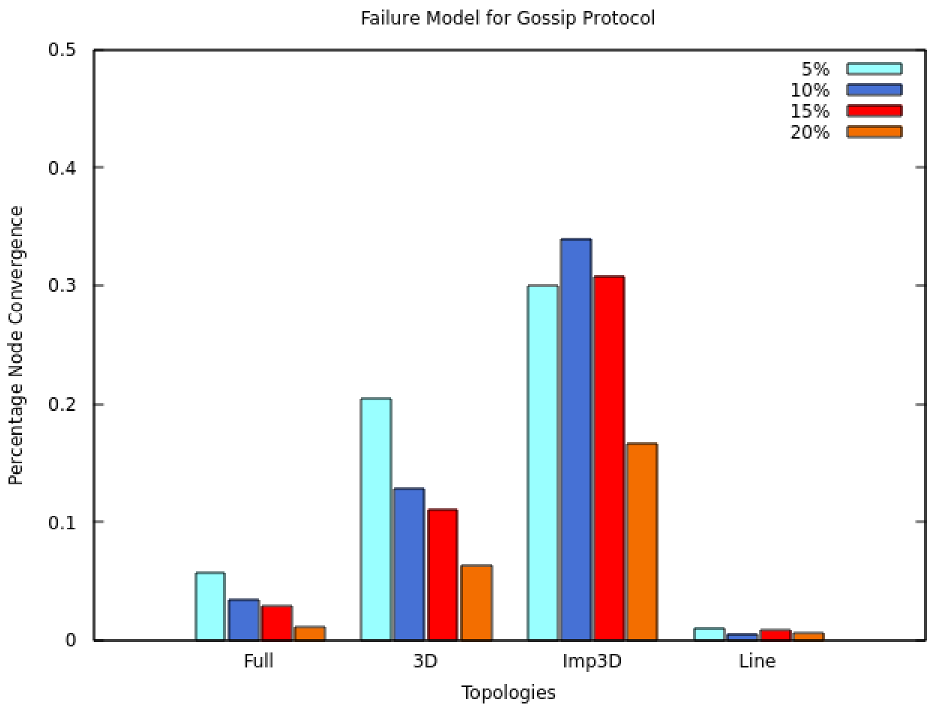

Figure 1: Gossip Protocol Run on various topologies with different percentage of nodes failing.

3 Push-Sum Algorithm

As expected, the Push-Sum took a lot more time to converge to the correct average value than the normal implementation. Figure **??** shows the time taken for convergence on different failure rates and different topologies.

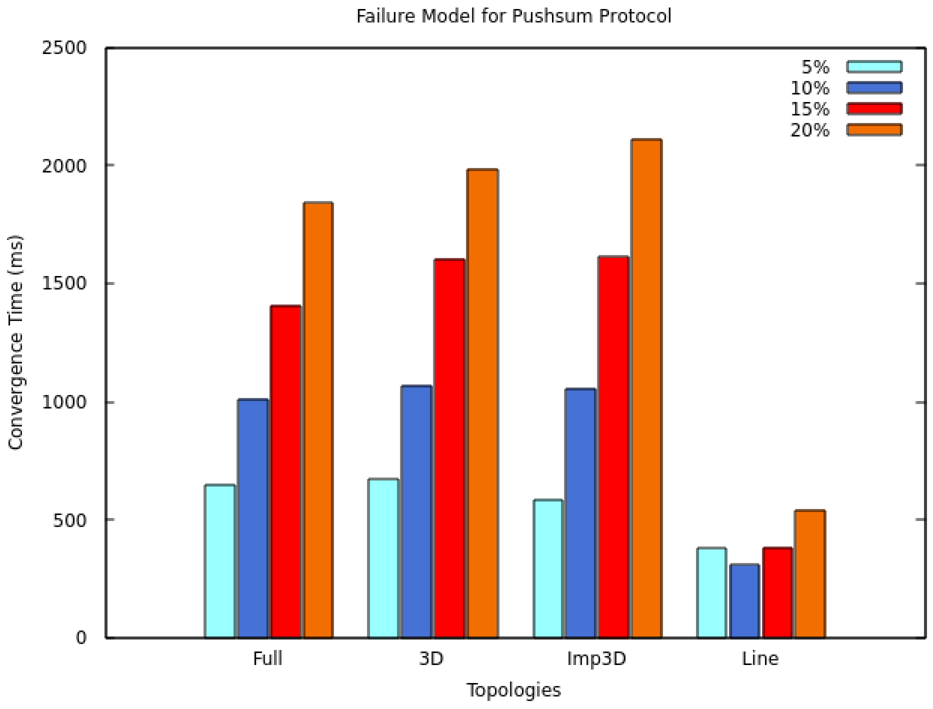

Figure 2: Push Sum Protocol Run on various topologies with different percentage of nodes failing.

For each combination of parameters, we ran the program a few times. The average of all the runs was then considered for our analyses and the graph plots.

4 Observations

In case of temporary failures, once the node was up and running, the protocol continued running and eventually converged. However, this process took a longer time.

In case of permanent failures, the protocol execution was stopped. Since the node failed, there was a portion of the network that was disconnected. As a result, convergence was not achieved.

The time does not monotonically decrease as we increase the number of failure nodes as it actually depends on which nodes fail. This is evident from the graph plots of the Gossip Protocol for permanent nodes failures, particularly in case of line and imperfect 3D topologies.

Out of all the topologies that we worked with, the line topology is the most inflexible, prone to disconnection, and most unpredictable topologies. Since a node can send a message only to 2 of its neighbors (left and right), if one of these nodes go down, it takes a very long time for the algorithm to converge. Also, the output for the same number of nodes is highly inconsistent.