**COP5615 – Fall 2019**

**Project 2 – Gossip and Push sum simulator**

**(Bonus – Failure Tolerance)**

***Project members:***

* Ankur Sethi (UFID: 9351-2951)
* Yogesh Laxman (UFID: 9451-2517)

***How to run:***

./project2 numNodes topology algorithm failure\_percentage

Where failure\_percentage is the percent of number of nodes to kill permanently.

***Brief Description:***

The aim of the bonus project is to implement a failure model and report any interesting findings.

Process:

* We are taking failure percentage as input and killing a random number of nodes equal to that percentage of the total number of nodes before initiating the algorithms.
* We do this before initiating the algorithms, but after creating the topologies, so that fault tolerance can be observed for each of them. Since the connection between them dies, there convergence becomes difficult.
* As an added output, we now show the convergence spread, to view how many of the nodes received the message correctly despite the failure.

Convergence spread = Number of nodes Converged / Total Nodes \*100

* We checked this output at 5%, 10% and 15% failure.
* We also introduced a timeout factor, so that the application is timeout out after a certain time, when there is no change in the convergence of the active nodes.

***Observations:***

* We saw all topologies performing extremely well, with the exception of “Line”. Line topology performs the worst because when links between the nodes are broken randomly, the message is not able to propagate to the other side and nodes exist as disconnected clusters. Thus, for line, the nodes never converge if any failures exist in the system.
* We checked the convergence spread of all topologies. While for line, the convergence spread falls **below 10%,** the common as observation was that as the failure percentage increases, the spread percentage increases. Besides the faulty nodes, some other nodes are also unable to receive the values, because disconnects in the system that have been introduced.
* Full topology converges for 100% for the active nodes. The faulty nodes obviously remain disconnected and the observed spread was 95%, 90% and 80% and for 5%, 10% and 80% failure nodes respectively.
* The rest of the topologies also perform extremely well and we see 95-100% convergence for the active nodes. 1-5% of the active nodes fail to converge and this should be because of the failed nodes creating disconnects. This is interesting as even t
* As observed without introducing any failure, for lesser number of nodes, around 100-300, the random2D topology does not perform well, because the number of interconnections tend to be on the lower side compared to densely placed nodes in case of higher number of nodes. When we introduce failure into this, the performance falls further, and the convergence spread decreases. Thus, the random2D topology is only recommended for higher number of nodes.
* The rest of the topologies perform extremely well, in the obvious order: honeycomb, random, honeycomb and torus, i.e. in increasing order of the inter-connections. These do no see failure to converge for more than 1-5% of the active nodes.
* Torus is the best performing topology observed across the experiment.
* We encounter the timeout condition fairly often for the push sum algorithm as the nodesmay get stuck with no active neighbors.

**Convergence Spread % vs Number of nodes for all topologies**

 

 

 