PROJECT-3-BONUS REPORT

Project partners:

1. Wins Goyal, UFiD – 73571559, *winsgoyal@ufl.edu*
2. Rahul Wahi, UFiD – 30536162, *rahul.wahi@ufl.edu*

1. Failure model for Tapestry Overlay Network:

1.1 - Failure Model

For the bonus part of this project, we have implemented one of the failure models described in the paper, i.e. the Involuntary Node Deletion. Nodes can enter or exit the Overlay network any number of times, which can be either voluntary (a graceful exit) or involuntary (less graceful, not intentional). This makes the network more dynamic. Dynamic overlay networks generally don't have nodes exiting gracefully. So, we fail some percentage of nodes in the network such that it is their involuntary exit and the nodes fail permanently.

We again use the Supervisor → Genserver pipeline for this project. In the failure model, we pass a new parameter through CLI called 'fail-percentage', along with numNodes and numRequests. Fail-percentage tell us the percentage of nodes in the overlay network that should go inactive, thus not responding to any relevant stimulus. This percentage varies from [0,100]. If all of the nodes in the network fail at a time, the prompt outputs "All nodes failed" and the program exits. We use a hexadecimal 40-digit identifier space to generate the nodeIDs.

As explained in the Paper as well, Surrogate routing plays an important role in maintaining the network stability and resilience to such failure tendencies of the System. We confirm such claims made in the Paper through our simulation and the different input parameters we pass to the simulator. The network, the way it's each routing table is built, maintains redundancy at each slot or level by storing a list of nodes rather than a single node at that level. So, if one or multiple node(s) fail in any process, the network can still easily route to either the target node or to the closest possible node to the target node.

(a) In the former case, the node that is requesting through the route retreats back to the last successful node it made connection with when it encounters a failed (non-responsive) node in the route. This requesting node again goes through different nodes from the forwarding pointers of that last successful node to reach the target node.
(b) In the later case, when the requesting node reaches a non-responsive target node, it find it's closest surrogate node in the route by the same above mentioned mechanism.
(c) When the proposed node from where routing begins to the target is itself inactive, then the network starts the routing from it's closest surrogate node.

The list of nodes stored at every level is the backup plan for the network to cope with such dynamism. This ensure no loss of data, thereby strengthening the reliability and stability of the network.

1.2 - Analysis method

After running the program on CLI, we get one outputs, i.e. maximum number of hops needed to find a target node from a requesting node, when the system is faulty. Please keep in mind that the system has dynamic characteristics like dynamic node insertion (10% of the node randomly enter into the network after it is fully setup) and dynamic node deletion, and the network is able to rewire its routing mesh by redistribution and redundancy at necessary places in the network. To analyze the system resilience, we

input different failPercentage ad numRequests for different number of nodes in the network. We check the pattern for maximum number of hops :→
(a) for low, medium and high failPercentage keeping the network and numRequests same.
(b) for same failPercentage but different numRequests.

In either of the cases, the network should either find the target node or atleast find the closest root node to the target node (if the target node itself failed). And, the maximum number of hops can't exceed the number of levels of a routing table in the network, i.e. $\boldsymbol{\log_b N}$ where b is the radix used to generate nodeIDs by SHA-1 and N is the namespace size.

	numRequests	failPercentage		
	1,10,20,50	5	65	95
numNodes	2000	6	7	5
	1000	5	6	5
	300	6	6	6
	50	4	4	3

Fig. A sample output of the Tapestry simulator, we check Max. hops for the varying parameters for more than half of the runs of the program.

2. Observations & Inferences:

- We test the network for a maximum of 10000 nodes and 100 requests with varying failPercentage. The max. number of hops comes out to be 8 for most of the time when the failPercentage is around the range of medium point, i.e. 50%. When greater than or equal to 90%, the number of hops come out to be <= 8 for most of the time.
- We check that the maximum number of hops varies during different runs of the simulator with different parameters because of the scope of Randomization in the setup of the overlay, dynamic addition and deletion of the nodes. So, it is wise to check the value persists for most of the run times with the same parameter. We say that we get the max. hop value which occurs more than half of the runs of the program.
- We find that, for more than half of the runs of the program, :
  1. When failPercentage is less or zero (<= 20%) → Max hop is less or equal to the maximum that we can get in following point 2.
  2. When failPercentage is medium (around 50%, say 25%-85%) → Max hop is high. Because network is relatively sparse and it is almost equally likely that we don't find the target node active or will encounter the inactive nodes many times.
  3. When failPercentage is high (>= 90%) → Max hop is less again.

  This trend is more observable when the network size is larger.
- When the failPercentage is low enough, the network is not affected much by the node failures. So, the probability of the target node not being inactive is very high or even for any node being encountered while routing. So, more than half of the runs of the program, we get the target efficiently fast, like when we get it when there is no failure at all in the system.
- When the failPercentage is high enough, the network is actually sparse, and the probability of finding the exact node reduces lot more. So, more than half the number of times, we don't find the required target node but some root (surrogate) node that very closely matches the target node. And, because there are lesser number of nodes to route through, so the max. hops also is less.
- The main conclusion to make from the tapestry setup is that it is always able to find the target and outputs the max. hops in the end. So, the data is relatively not lost during failure dynamics.