**COP 5615 Distributed Operating Systems Fall 2019**

**Project 3 – Tapestry Algorithm**

**October 25, 2019**

**Anirudh Mukundan Raghavan (UFID: 6416-8277)**

**Aditya Karlekar (UFID: 8888-9598)**

**Instructions to run the program**

1. Extract the tgz file proj3.tar.gz.
2. Extract the tar file proj3.tar generated from the step above, into a folder called proj3.
3. Go to the folder *proj3*.
4. Run the command mix run project3.exs numNodes numRequests.

* Where numNodes is the number of nodes we want in the peer to peer system and numRequest is the number of requests we wish each node to make.
* For example, for 100 nodes and 10 requests, the command will be as follows:
* mix run project3.exs 100 10

**Project Implementation:**

* The Tapestry algorithm is implemented as described in the paper “Tapestry: A Resilient Global-Scale Overlay for Service Deployment” by B. Y. Zhao, L. Huang, J. Stribling, S. Rhea, A. D. Josheph and J. D. Kubiatowicz, published in IEEE Journal on Selected Areas in Communications, Vol 22, No 1, January, 2004.
* The program takes number of nodes as the input and creates a peer to peer network.
* The input also specifies the number of messages each node must send.
* The nodes are provided unique ids by using SHA-1 hash function, using the PID’s of each node.
* The messages are the node ids of the destination and are generated randomly.
* For each node, a routing table is created consisting of the node ids of its neighbors. We decide the correct position of each node in the routing table by using prefix matching, as specified in the paper. If two or more nodes can be fit into one position in the routing table, we insert the node which is closest to the root node. We define closeness by finding the absolute value of the difference between the two nodes and the root node.
* At the time of network creation not all nodes join the network. Once the network is established, we add in an additional node to the network. The added node generates its own routing table as well as the routing tables of existing nodes are updated to reflect the change in the network. (Network Join).
* We send messages from each node by using the protocol described in the paper. We select the destination node and root node randomly from the network. We compare the destination node hash with the source node hash. Based on the common prefix found, we locate the closest node to the destination node in the source hash table. We then hop to that node in the routing table. This is done until the source and destination nodes are the same. Then, we store the number of hops taken for this message. We have not implemented surrogate routing in our project. Since we are finding nodes which are already present in the network, we have decided to not implement surrogate routing.
* Finally, the maximum number of hops taken by all messages is printed.

**What is working:**

* Creating of network as per the input: The program successfully creates a peer to peer network based on the number of nodes specified in the run command.
* Generating Routing Table: For each node in the network we generate the routing table consisting of addresses of its neighbors. We use the scheme described in the Tapestry paper to fill the routing table.
* Message Generation: The message is the node id for node present in the network.
* Sending Requests: The program sends the specified number of requests from each node within the network.
* Network Join: At the time of network creation not all nodes join the network. Once the network is established, we add in an additional node to the network. The added node generates its own routing table as well as the routing tables of existing nodes are updated to reflect the change in the network.
* Output: The program outputs the maximum number of hops required to reach the destination by any message within the network.

The largest network we managed to deal with:

* Number of Nodes: 5000
* Number of Requests: 1
* Maximum Hops: 6

**Observations:**

We observe that as the number of nodes increases, the number of hops remains below the order of growth mentioned in the paper.

|  |  |  |
| --- | --- | --- |
| Number of Nodes | Number of Requests | Maximum Hops |
| 100 | 1 | 3 |
| 100 | 10 | 3 |
| 200 | 1 | 4 |
| 200 | 10 | 4 |
| 500 | 1 | 4 |
| 500 | 10 | 4 |
| 1000 | 1 | 5 |
| 1000 | 10 | 5 |
| 2000 | 10 | 5 |
| 3000 | 1 | 5 |
| 4000 | 1 | 6 |
| 5000 | 1 | 6 |