Distributed Operating System Principles

(Fall 2020)

Project 3

Venkata Vikramaditya Varma Kunaparaju	1537-8659
Prajan Tikayyolla	6690-9943

Project Description:

We have implemented the pastry protocol in this project.

How to Run the Project:

Open the FirstIonideProject and in the terminal run the following command.

dotnet fsi --langversion:preview project3.fsx 10000 100

The first Input is the number of Nodes; the second one is the number of requests each node has to make.

Pastry Protocol:

- All the nodes in the pastry protocol form an overlay network
- Each node is identified with a unique identifier (NodeID)
- Whenever a message is sent to a node, it routes the message to the nearest node based on its key

Implementation:

- We have taken the value of b to be 3 for this pastry implementation
- Joining into the network starts with an assumption that we know a node which is close to the joining node based on proximity metric, here in this implementation we have chosen the proximity node to be a random node from the nodes which are already in the network.
- The proximity node then checks whether the node falls in the leafset range if so then it routes the “join” message to the node which is nearest among the leafset.
- If if does not fall under leafset, it calculates the common prefix with the node id associated with “join” message and checks for a node which has a common prefix length which is 1 greater and routes the message to it.
- If both the conditions are not satisfied, then we check for nearest node among all leaf, routing and neighbourhood nodes and with common prefix length which is greater than or equal to the length between the current node and node associated with “join” message.
- During the process of routing, the intermediate nodes send their states to joining node along with a timestamp attached and the received node updates its state both routing and leaf and maintains the timestamp.
- Once the joining node reaches the closest node, it then sends its state to all nodes encountered in the path, leaf nodes and nodes in the routing table. Along with state it also sends the timestamp to nodes which encountered in its path.
- The intermediate nodes then update their state and checks if its state changed after the received timestamp and if it has changed then again it sends its state and process continues.

- Once the nodes are joined, we then generate a request for each node to calculate the average number of hops it takes to route the node to its closest node.
- Below image shows the path encountered to “route” a node and the states of intermediate nodes.

```
Routing completed for "0561700044434462" with path seq ["7553753060136756"; "0747211044061140"; "0533704135072230"]

Node id : "7553753060136756"
Routing table : [["0747211044061140"; "1560366774326004"; "2576227404714674"; "3753571607415330";
  "4765142471714762"; "5677605445532544"; "6640341744705675"; "7553753060136756"]
 ["7066032641713514"; "null"; "7222467063531255"; "null"; "7470423110747500";
  "7553753060136756"; "7624322753773622"; "7730004046450202"]
 ["null"; "null"; "null"; "null"; "null"; "7553753060136756"; "null"; "null"]]
LeafSet small : [|"7450101532635623"; "7470423110747500"; "7427677716507463"; "7415116640770520"|]
LeafSet Large : [|"7730004046450202"; "7625332670254356"; "7624322753773622"; "null"|]

Node id : "0747211044061140"
Routing table : [["0747211044061140"; "1032464032056707"; "2014170310573713"; "3021364252113400";
  "4013473651636523"; "5015502122465653"; "6001472050340425"; "7020772377301707"]
 ["0040721417160150"; "0173667074535210"; "null"; "null"; "0452632440234043";
  "0533704135072230"; "0673236223513436"; "0747211044061140"]
 ["null"; "null"; "null"; "null"; "0747211044061140"; "null"; "null"; "null"]]
LeafSet small : [|"0673236223513436"; "0533704135072230"; "0452632440234043"; "0643207102055741"|]
LeafSet Large : [|"1121370642452767"; "1032464032056707"; "1034377164043352"; "1174250066116623"|]

Node id : "0533704135072230"
Routing table : [["0533704135072230"; "1032464032056707"; "2014170310573713"; "3021364252113400";
  "4013473651636523"; "5015502122465653"; "6001472050340425"; "7020772377301707"]
 ["0040721417160150"; "0173667074535210"; "null"; "null"; "0452632440234043";
  "0533704135072230"; "0643207102055741"; "0747211044061140"]
 ["null"; "null"; "null"; "0533704135072230"; "null"; "null"; "null"; "null"]]
LeafSet small : [|"0435220070541100"; "0452632440234043"; "0126406033065515"; "0157133551744210"|]
LeafSet Large : [|"1034377164043352"; "0747211044061140"; "1032464032056707"; "1121370642452767"|]
```

Results:

The following is the results table for the average number of hops when the number of requests is 5 and 10, respectively.

Number of Nodes	Requests = 5	Requests = 10
10	1.88	1.89
100	2.81	2.7
500	3.38	3.38
1000	3.68	3.67
5000	4.19	4.13
10000	4.23	4.29

Below is the graphical representation of the above results.