COP5615: Distributed Operating Systems

Project 3 - Bonus Report

Subham Agrawal | UFID - 79497379

Pranav Puranik | UFID - 72038540

Aim - Implementing failure model for Tapestry Algorithm.

Introduction -

Tapestry is a peer-to-peer overlay network routing algorithm. It uses a distributed hash table for each node to make routing faster. Tapestry is efficient, scalable, and self-repairing.

Implementation -

- Node ids are generated using **sha1 hashing** algorithm. The hash value has 8 digits and we are using the hexadecimal naming scheme.
- For each node, we are **storing the closest nodes** in the routing table by iterating over all other hash generated.
- Each cell of the routing table stores **multiple (c = 3) references** where the first one is the closest node and other are for backup, used in case of failure.
- Therefore, the size of routing table will be at most c x 8 x 16 (backup nodes x hash length x base-value).
- In addition to routing tables, each node also stores r**everse references (backpointers)** to other nodes that point at it.
- The last node is **dynamically added** to the network. To achieve this firstly, the new node contacts its root node to perform an **acknowledged multicast** in order to update itself in routing table of its **needtoknow nodes**. Secondly, it does **backpointer traversal** to find the best/closest set of nodes to fill its routing table with .
- While routing the messages, next node is selected from the routing table by looking at the closest match. This **surrogate routing** is used to deliver the message to its receiver.

- Each node sends "numRequests" (given as input) messages to other nodes and keeps the **maximum number of hops**.
- A counter node stores the maximum count of hops for the entire network. After completing all the requests, the nodes send their maximum number of hops to this process.

Results-

We executed the code for a range from 50 to 5,000. The testing was performed on Intel i5 4th Generation processor with 12GB Ram, 512 SSD.

NumNodes	NumRequests	NumNodesToFail	Without Failure	With Failure
50	3	10	2	2
100	3	20	2	3
500	3	100	3	3
1000	3	200	3	4
2000	3	400	4	12
5000	3	1000	4	17

Observation and Inference -

In a network like Internet, many nodes exit due to node and link failures or network partitions, and may enter and leave many times in a short interval. These things happen in a chaotic manner. Developing redundancy into routing tables and object location references, Tapestry algorithm ameliorates object availability and routing. Tapestry retains nearly a 100% success rate at routing messages to nodes and objects.

- More number of hops are required for a message to reach its destination in case of node failures in the network. The number of hops is proportional to nodes failed.
- Since the hops increase, it takes more time for a message to be delivered.
- Deleting a lot of nodes sometimes deletes all the backup nodes as well. This breaks the network. and creates a hazard. New elements should be added in place of failed nodes to avoid this scenario. Whenever a node understands that it's neighbour.

References -

- **matrix.ex is a module used to store and easily access Routing tables in the tapestry network, available on the website-** **https://blog.danielberkompas.com/2016/04/23/multidimensional-arrays-in-elixir/**
- **https://pdos.csail.mit.edu/~strib/docs/tapestry/tapestry_jsac03.pdf**
- **http://cs.brown.edu/courses/cs138/s17/content/projects/tapestry.pdf**