COP5615 Distributed Operating System

Project 3

README

 # Group Members

1. Bhavya Kavdia, UFID: 7017-7277

2. Ishita Trivedi, UFID: 6893-6496

Problem Statement -

We talked extensively in class about overlay networks and how they can be used to provide services. The goal of this project is to implement F# using the actor model and the Chord protocol and a simple object access service to prove its usefulness. We implemented the network join and routing as described in the Chord paper and encoded a simple application that associates a key (same as the IDs used in Chord) with a string.

Maximum no of nodes we managed to simulate for the Chord protocol implementation are 2000.

Note: This file contains information and command about project 3: The Chord Protocol

Input –

1. numNodes - number of nodes to be created in the peer-to-peer network
2. numRequests - Number of requests that each peer must make to other peers

Command to run –

1. To run Project 3:
   dotnet fsi Project3.fsx <numNodes> < numRequests >

Implementation –

- **Steps:**

1. We have applied the Chord protocol by using the join and routing algorithm as specified in the paper. We built the network topology by constructing a network ring of peers and a finger table for each node which maintains routing information for approximately log(n) peers where n is the total no of nodes in the network.
2. The algorithm gets initiated by creating nodes equal to numNodes, assigning each node with a Node ID which is SHA-1 hashed and each node creates requests equal to numRequests entered by the user. Each node then looks up for the keys requested which are also SHA-1 hashed and the program finally counts the average no of hops the nodes take to find the key.

- **Algorithm implementation:**

1. The program starts with the boss actor creating a chord ring like structure for actors which represent peers in the network and for each node we define a predecessor and a successor node. Each node has a finger table where it maintains entries of nodes to hop to when resolving keys to nodes.
2. Once the network is up, each node gets assigned a fixed set of requests, namely key lookups to perform. During this time, a lookup function is called recursively on nodes mentioned in the finger table entry and the hop count gets increased with each recursion. The average of this hop count is calculated which is defined as the total no of hops / total no of requests.
3. We have used SHA – 1 algorithm for hashing the Node IDs and the keys in the network.
4. The program will conclude with each peer node reporting to the boss node when it has processed all the requests assigned to it.

- **Termination:**

  The program terminates when all the actors have successfully created the number of requests as specified by the user through command line arguments. Each peer node creates and sends a request per second and informs the boss node when it has finished processing and sending all the request messages assigned to it.

- **Output:**

  The average number of hops is defined as total number of hops/total requests. Largest Network Simulation Achieved: 2000 nodes, Requests per node: 50, Average Number of Hops: 3.786

- **Table** displaying average hop for different no of nodes and different no of requests per node: -

Number of Nodes	**Number of Requests Per Node**	**Average Hop Count**
10	10	1.512
10	100	1.15
50	50	1.732
50	100	1.823
100	50	2.239
100	100	2.213
150	100	2.202
200	100	2.363
500	100	2.789
1000	100	3.058
1500	50	3.243
2000	50	3.786