Lab Distributed Hash Table

3rd of October, 2019

Introduction

Our second assignment is to design a simple distributed hash table. Distributed hash tables are key-value data structures. The keys are distributed across different nodes. The values are located on the same node as the key. We will be implemented a version of the Content-addressable Network (CAN). The original paper can be found here: http://www.eecs.berkeley.edu/~sylvia/papers/cans.pdf

The algorithm

We will implement several features of the CAN algorithm. The tasks are as follows:

1. Joining a CAN (30%)
2. Leaving a CAN (10%)
3. Setting a key to a value (15%)
4. Getting a value for a key (15%)
5. A display of the key space (15%)
6. Demo/readme/style/makefile (15%)

The keys will be pairs of floating point numbers, which can be thought of as points. The points can range from (0,0) to less than (1,1). The key space is divided amongst interlinked nodes. A string value can be associated with a key. Values will be no longer than 63 characters.

Each node must keep track of the keys(points) it is supposed to hold. Nodes also must keep track of their bounds and their neighbors (both network addresses and point bounds). Nodes communicate through messages (you will need to design these messages). Nodes WILL NOT know the network addresses of non-neighbors. Inside of the hash table, messages use points as addresses.

Messages are routed to the neighbor with the closest midpoint to the desired point. If a neighbor contains the point it should receive the message. The operations of setting a value and getting a value utilize the same routing scheme.

Joining a CAN is more difficult. The hostname and port of a node in the CAN must be known to join. This contact node will create a random point to choose an established node to split. The established node splits its area in half, giving half to the new node. If the area of the bounds is square the split will be vertical, if the area is not square the split will be horizontal. Neighbor lists and key/value pairs will need to be updated.

Hints

You can draw the key space using ASCII letters to represent the different nodes and sampling the CAN at different intervals. The draw function might be a good way to debug.

This project is about constructing messages. Think about what each message should include. When asking to join what needs to be known? When accepting a new node what should be sent?

Handing in

This project is due Oct. 18 at 11:59pm. Use the handin command to submit your files: handin cmsc360 lab3 yourfileshere You will also need to demo your project for me. Feel free to discuss with your classmates but WRITE your own code.