UrDHT: A Unified Model for Distributed Hash Tables

Andrew Rosen Brendan Benshoof Robert W. Harrison Anu G. Bourgeois
Department of Computer Science
Georgia State University
Atlanta, Georgia
rosen@cs.gsu.edu bbenshoof@cs.gsu.edu rharrison@cs.gsu.edu anu@cs.gsu.edu

***Abstract*—UrDHT is an abstracted Distributed Hash Table (DHT). By completing a few simple functions, a developer can implement the topology of any DHT.**

Current distributed systems suffer from fragmentation , high overhead and inability to scale due to difficulty of adoption. UrDHT is P2P system designed to improve the adaptability of P2P distributed serves.

I. Introduction

Distributed Hash Tables have been extensively researched for the past decade. Despite this, no one has created a cohesive formal specification for building a DHT.

UrDHT is our specification and implementation of an abstract DHT.

- We first discuss our motivation for creating UrDHT and *creating it the way we did* (Section II).
- We give a formal specification for what needs to be defined for a DHT. These attributes have not been formally defined. (Section III)
- We present UrDHT as an abstract DHT and show how a developer can tweak the functions we defined to create new DHT topologies. We show how to reproduce the topology of Chord and Kademlia using UrDHT, which we call UrChord and UrKademlia.
- We conduct experiments showing that UrChord sufficiently approximates a correct implementation of Chord.

II. Motivation

Distributed Hash Tables have been the catalyst for the creation of many P2P applications. Among these are example 1, example 2, another citation, citation, and, most notably, BitTorrent [1].

One issue in the adoption of new P2P applications is the bootstrapping problem. A node can only join the network if it knows another node *that is already a member of the network it is trying to join.*

The other motivation is making it easier for users to create distributed applications. What topology do you use? How do we want our program to communicate over the network?

UrDHT exists to simplify this process, minimizing the distributed application development time and making it easier to adopt by creating a network to bootstrap *other networks*.

III. What Defines a DHT

A distributed hash table provides the following API to the user

A `distance` function
: This measures distance in the overlay formed by the Distributed Hash Table. In most DHTs, the distance in the overlay has no correlation with real-world attributes. This is not the case with UrDHT (see Section IV-B).

A `midpoint` function

An `ownership` definition

A DHT also needs a strategy to organize and maintain two lists of peers: *short peers* and *long peers*. Short peers are the set of peers that define the topology of the network and guarantee that greedy routing works.

Long peers allow the DHT to achieve a better than linear lookup time,

Interestingly, despite the diversity of DHT topologies, all DHTs use the relatively the greedy routing algorithm:

IV. UrDHT

A. UrDHT Components (or maybe logic)

UrDHT is sectioned off into 3 components: database, network, and logic. Database handles file storage and network dictates the protocol for how nodes communicate.

B. Hyperbolic Routing

C. Implementing Chord and Ring Based Topology

D. Implementing Kademlia and Other Tree Based Topologies

Trees are easy to embed in a hyperbolic space.

E. ZHT

V. Experiments

VI. Future Work and Conclusions

References

[1] Bram Cohen. Incentives build robustness in bittorrent. In *Workshop on Economics of Peer-to-Peer systems*, volume 6, pages 68–72, 2003.