Attributes of Distributed Hash Tables and Their Ramifications

Andrew Rosen

June 24, 2014

Chapter 1

Introduction

Chapter 2

So What Are Those Attributes Anyway?

2.1 The Coordinate System, Responsibility and Neighbors

The vast majority of overlays only use a single coordinate (and therefore a single dimension). This coordinate is assigned essentially at random through a hash function, typically by hashing the IP address and port. If more than one dimension is used, coordinates can be chosen to map to a particular physical feature (geographical location or available bandwidth)

All DHTs assign responsibility the same way: the node with an ID that is "closest" to a particular key is responsible for that key. What differs is the metric being used to determine which node is closest. One metric used by Chord (others) is "the node with the closest id larger than the key" (presumably, this is much easier to code than "the closest without going over.).

Almost every DHT operates in a modulus space. The space can be either bidirectional (symphony) or unidirectional (Chord). The overlay shape/layout is actually as important as the peer list or routing algorithm. There's typically a distance function, but it's not very interesting.

The domain of the keyspace can be m-bit identifiers, or floats from 0-1. So long as m is the same it is possible to map from one to another.

2.1.1 Closest Metrics

Literal

As in literally the closest. The cost is computation, but the benefit is that the metric is natural and easy to understand

Successors

Predecescors

The node with the ID closest without going over a key is responsible for that key.

Groups

See Pastry

2.2 Finding Peers

Some DHTs build their peer lists by using suggestions from other nodes. The issue with this is that this opens up a new vector for attacks. A malicious node can lie to another node about what peers to use, but even worse, these lies can then propagate to other nodes.

Others DHTs choose their peers by finding the node closest to a particular ID. For example, a node in Chord chooses fingers by finding the node closest to its ID plus some power of 2. In Symphony, peers are chosen by finding the node closest to some ID, chosen randomly over a probability distribution.

2.3 Routing

All routing algorithms follow the same scheme: the next hop brings me closest to my destination. The very nature of DHTs makes the routing process greedy, and the node chooses the best path based on its peer list (It is possible to use a 1-ahead lookup, thereby keeping a record of your peer's peerlist. The cost becomes quite large for more than two, but routing does speed up.). Specifics aside, there are two ways to look at routing. The first way answers the question "Who is responsible for this particular key?" but rather than asking the entire network, it assumes the entire network consists only of it and the peer list. The other way to look at it that routing is the process of greedily eliminating possible recipients, until the final node must be the node responsible.

Almost every DHT has a lg(n) routing time, eliminating half of the remaining nodes each step. However, Kleinberg Small World Networks have inspired networks that have a polylogarithmic time in exchange for smaller peerlists and routing tables.

2.4 Churn and Fault Tolerance

DHTs can choose various ways to implement fault tolerance. First, a DHT must have some mechanism for neighbors to take over for node failure (although it doesn't have to). Next is peerlist maintenance. How are failed lookups handled? How do you handle churn?

2.5 Security

2.5.1 Sybil Attacks

2.5.2 Eclipse Attacks

Chapter 3

The Four Kings

3.1 Shared Attributes

3.2 Chord

3.3 Pastry

Addressing - 128 bit ID, 0 to $2^128 - 1$, assigned randomly using hash. but thought of as base 2^b numbers (typically b=4). This creates a hypercube topology [**?**].

Peerlist - A routing table, neighborhood set and a leaf set. The Routing table consists of log n rows and b-columns each. The 0th row contains peers which don't share a common prefix. The 1st row contains those that share a length 1 common prefix, the 2nd a length 2 common prefix, etc. Since each ID is a base 2^b number, there is one column for each possible difference. The i,j entry of the table contains an ID that shares the same first i digits, with digit i+1 having a value of j (yes, a slot is wasted in each row).

The neighborhood set hold the ID and Ip address of the closest nodes, defined by a metric. It is not used for routing. The leaf set is used to hold the numerically closest nodes. Half of it for smaller and half for larger.

The table is populated at first by a join message to the node responsible for the joining node id. As part of the join message, nodes along the path send their routing tables. After the joining node creates it's routing table, it sends a copy to each node in the table, who then can update their routing tables. Node join cost is $O(log_2^b n)$ messages with a constant coefficient of $3 * 2^b$

When a node leaves the network, its neighbor contacts it's leaf closest to the failed node for its leaf table. That information is used to repair the leaf set. A failed routing node is replaced with another appropriate node for that slot.

Who do we actively back up to? Pastry is only about routing. PAST stores a file to the k closest nodes with id's closest to the file. This allows messages to

make it to any one of the k nodes that can respond to that file lookup (most likely the closest one to the originator)

Eclipse attack would basically work like this - when a node asks the malicious one for peer info, the malicious node replies with IDs it makes up on the spot, each bound to it's IP. These IPs would be spread throughout the keyspace so that any malicious value has a good chance of being chosen.

Routing - Forwarded to (node/peer?) whose shared prefix is longer. If no one has a better shared prefix than the current node, the message is forwarded to the closest node.

Pastry's goal is to minimize the "distance " messages travel, but distance can be defined by some metric, typically the number of hops.

The leaf set is the of nodes closest to the node in the keyspace. The neighborhood set is the of nodes closest to the node according to the distance metric Guarantees routing time is ¡ lg n in typical operation. Guarantees eventual delivery except when half of the leaf nodes fail simultaneously.

Fault Tolerance - A failed node doesn't delay routing, because Pastry's routing table allows it to just send to the next closest node. Damage to the routing table is replaced by contacting other nodes and requesting a suitable replacement

3.4 Tapestry

3.5 CAN

Chapter 4

The Challengers

4.1 Kademlia

Chapter 5

The Small New World

5.1 The Small World

Kevin Bacon

The experiment

Kleinberg

5.2 Voronoi Based Schemes

5.2.1 RayNet

Beaumont et al argues that a loose structure enough for searching. Assume a d-dimension space, each dimension tied to some attribute of an object and each object identified by a unique set of values. Objects should be linked to other objects that are close in the space.

Chapter 6

Momentum

Or who's idea was it that the darn things don't actually move!