

# DHT Performance Overview

Miika Komu <miika@iki.fi>

# What is a Distributed Hash Table (DHT)?

- Hash Table:
  - value = lookup(key)
  - store(key, value)
- Distributed: storage and lookups of values are distributed among multiple hosts
- Motivation: how do you find a value in a large P2P system in a scalable manner without any centralized servers or hierarchy?

# Properties of DHTs

- Each node...
  - has a unique node ID
    - The value is stored at the node whose ID is closest to the key
    - Closeness = distance function
  - maintains state: a small list of the node IDs (“neighbours”) and the corresponding IP addresses
  - forwards queries for a key to the closest neighbour
- Routing geometries
  - Skiplist, tree-like, multidimensional
- Iterative vs. recursive routing (factor: 0.6)

# Chord (Ring)

- Distance function = numeric difference between two node Ids
- Skiplist like (power of two) routing

# Pastry

- Distance function = number of common prefix bits
- Tree-like routing
  - Two-stage routing protocol (leaf set, routing table)

# Tapestry

- Distance function = number of common prefix bits
- Tree like routing
- Uses “salt” to avoid root node failures

# CAN

- Routing geometry: d-dimensional cartesian coordinate space
- Distance function: adjacent “zone”

# Kademlia (XOR)

- Distance function =  $\text{XOR}(\text{ID1}, \text{ID2})$ 
  - Unidirectional: does not need a two-stage protocol like Pastry
  - Symmetric: no need for a stabilization protocol like in Chord; routing tables are refreshed as a side effect of ordinary lookups

# Performance Evaluation

- Metrics
  - Number of hops
  - Latency
- Things that affect performance
  - Churn
  - Packet loss
  - Proximity Routing
  - Caching

# Performance Bounds/Results

	<b>Lookup</b>	<b>State</b>	<b>Relative Delay Penalty</b>	<b>Median HOP cour</b>
<b>Chord</b>	$O(\log N)$	$O(\log N)$	6	7
<b>Pastry</b>	$O(\log N)$	$O(\log N)$	9	8
<b>Tapestry</b>	$O(\log N)$	$O(\log N)$	N/A	8
<b>CAN</b>	$O(dN^{1/d})$	$O(d)$	6	8
<b>Kamdelia</b>	$O(\log N)$	$O(\log N)$	N/A	8

RDP = 1000 nodes, no failures

HOP = 65536 nodes, no failures

# Optimal Lookup and State

- Beehive achieves  $O(1)$  performance with proactive caching
- Butterfly keeps only  $O(1)$  state
  - Caveat: Median hop count 21

# Resilience and Recovery

- Resilience through flexibility
  - Flexibility in neighbour selection yields better paths than route selection
  - Chord and Kamdelia have the greatest flexibility
  - Tree and butterfly have the least
- Churn recovery
  - Periodic better than reactive

# Summary

- Performance of all DHT algorithms is pretty good
- DHTs can handle churn (P2P environments) and path failovers
- $O(1)$  lookups with proactive caching (DNS?)
- Lot's of papers and implementations available

# Future Work

- Authentication, Authorization, Accounting
  - The impact on performance
- So far, only application level – would this work directly on network level?
- Competing free version of i3?

# Bibliography

- [1] Zhao et al. Tapestry: A Resilient Global-Scale Overlay for Service Deployment
- [2] Rhea et al. Handling Churn in a DHT
- [3] Ratnasamy et al. A Scalable Content-Addressable Network
- [4] Stoica et al. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications
- [5] Rowstron et al. Pastry: Scalable, Decentralized Object Location and Routing for Large-scale Peer-to-peer Systems
- [6] Balakrishnan et al. Looking Up Data in P2P Systems
- [7] Maymounkov et al. Kademlia: A Peer-to-peer Information System Based on the XOR Metric
- [8] Sit et al. Security Considerations for Peer-to-peer Distributed Hash Tables
- [9] Ramasubramanian et al. Proactive Caching for Better than Single-Hop Lookup Performance
- [10] Jain et al. A Study of the Performance Potential of DHT-base Overlays
- [11] Castro et al. Performance and Dependability of Structured Peer-to-peer Overlays
- [12] Li et al. Comparing the Performance of Distributed Hash Tables under Churn
- [13] Ramasubramanian et al. Beehive:  $O(1)$  Lookup Performance for Power-Law Query Distributions in Peer-to-peer Overlays
- [14] Dabek et al. Designing a DHT for Low Latency and High Throughput
- [15] Gummadi et al. The Impact of DHT Routing Geometry on Resilience and Proximity