Kademlia on the Open Internet

How to Achieve Sub-Second Lookups in a Multimillion-Node DHT Overlay

Raul Jimenez

Licentiate Thesis in Communication Systems
Outline

- Context
- Introduction
- Background
- Main results
- Conclusion
<context>
• Content distribution platform
• Fully decentralized
• Streaming support
• Web integration
P2P content distribution
</context>
How do I find peers?
Peer discovery (aka *tracker*)

Peers for 787C32A...? --> tracker

P1, P2, P3

---

Peer discovery (aka *tracker*)

Peers for 787C32A...? --> tracker

P1, P2, P3
Peer-to-peer data transfer
Tracker

- Server
  - Centralized
  - Simplicity
  - Statistics
  - Scalability
  - Single point of failure
Tracker

- **Server**
  - Centralized
  - Simplicity
  - Statistics
  - Scalability
  - Single point of failure

- **Distributed hash table**
  - Decentralized
  - Scalability
  - No single point of failure
  - Churn
  - High latency

- Single point of failure
Tracker

- Distributed hash table
  - Decentralized
  - Scalability
  - No single point of failure
  - Churn
  - High latency
<background>
Distributed Hash Table (DHT)
Distributed Hash Table (DHT)

• Key, value store/retrieve service

<table>
<thead>
<tr>
<th>KEY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tel nr.</td>
<td>Name</td>
</tr>
<tr>
<td>Person ID</td>
<td>contact info</td>
</tr>
<tr>
<td>Product ID</td>
<td>stock info</td>
</tr>
<tr>
<td>Content ID</td>
<td>list of peers</td>
</tr>
</tbody>
</table>
Distributed Hash Table (DHT)

- Key, value store/retrieve service
- Same as “normal” hash tables but:
  - Values are stored on nodes
  - Nodes are connected to other nodes, forming an overlay
  - Lookup mechanism to locate the nodes responsible for a given key
- DHT designs:
  - Chord, Tapestry, CAN, Pastry, Kademlia, …
Kademlia
Kademlia

- XOR metric
- Iterative lookup

Kademlia-based overlays on the Internet
  - Mainline DHT
  - KAD
  - Azureus DHT

> 1 million nodes each!
Kademlia's routing tables
Kademlia's routing tables

My nodeID
001100...
Kademlia's routing tables

My nodeID
001100...
Kademia's routing tables

My nodeID
001100...

000...
0010...
001101...
001111...
01...
1...
Kademlia's routing tables

My nodeID
001100...
Kademlia's routing tables

My nodeID
001100...
Kademlia's routing tables

My nodeID
001100...
Kademlia's lookup
Kademlia's lookup

My nodeID 001100...

Lookup target 111000...
Kademlia's lookup

My nodeID 001100...

Lookup target 11000...

0 1 3 5 4 2
Kademlia's lookup

My nodeID
001100...

Lookup target
11000...

Get value for 11000...
Kademlia's lookup

My nodeID: 001100...

Lookup target: 11000...

Get value for 11000...
No value, ask node 1101...

111...
Kademlia's lookup

My nodeID 001100...

Lookup target 11000...

No value, ask node 1101...
Kademlia's lookup

My nodeID 001100...

Lookup target 11000...

Get value for 11000...

No value, ask node 1101...
Kademlia's lookup

My nodeID: 001100...

Lookup target: 11000...

Get value for 11000...

No value, ask node 11000...
Kademlia's lookup

My nodeID 001100...

Lookup target 11000...

Get value for 11000...

VALUE
Kademlia's performance

• In the lab
  • [Li-05]
    - Median: 450 ms
  • [Kaune-08]
    - Median: 250 ms
  • [Rhea-05]
    - Median < 200 ms
    - 99 p < 600
  • [Dabek-04]
    - 100–300 ms

• On the Internet (1M+)
  • [Crosby-07]
    - MDHT median: ~1 min
    - ADHT median: ~2 min
  • [Falkner-07]
    - ADHT median: 13 s
  • [Stutzbach-06]
    - KAD median: 2 s
  • [Steiner-09]
    - KAD median: 1.5 s
<main results>
Main results

- Connectivity properties
- Sub-second lookups
- Scalability & Locality
Connectivity Properties of Mainline DHT Nodes
Why Mainline DHT is so slow?

My nodeID 001100...

Lookup target 11000...

Get value for 11000...
Why Mainline DHT is so slow?

My nodeID
001100...

Lookup target
11000...

Get value for 11000...
Why Mainline DHT is so slow?

My nodeID 001100...

Lookup target 11000...

Get value for 11000...
Connectivity properties

Reciprocity

Transitivity

Persistence

A

B

C

A

B

A

B

t0
t1
Reciprocity / Persistence

My nodeID 001100...

Lookup target 11000...
Reciprocity / Persistence

My nodeID 001100...

Lookup target 11000...
Reciprocity / Persistence

My nodeID 001100...

Lookup target 11000...

Get value for 11000...
Reciprocity / Persistence

My nodeID 001100...

Lookup target 11000...

Reciprocity 20% FAIL (immediate check)
Persistence 56% FAIL (after 5 minutes)

Get value for 11000...
Reciprocity / Persistence

My nodeID 001100...

Lookup target 11000...

Reciprocity 20% FAIL (immediate check)
Persistence 56% FAIL (after 5 minutes)

Get value for 11000...

Solution:
Opportunistic → connectivity check → quarantine
Transitivity

My nodeID 001100...

Lookup target 11000...

Checking...
Transitivity

My nodeID 001100...

Lookup target 11000...

Checking... OK
Transitivity

My nodeID 001100...

Lookup target 11000...

Get value for 11000...
Transitivity

My nodeID 001100...

Lookup target 11000...

Get value for 11000...
Transitivity

My nodeID 001100...

Lookup target 11000...

Transitivity 60% FAIL

Get value for 11000...
Transitivity

My nodeID 001100...

Lookup target 11000...

Get value for 11000...

Transitivity 60% FAIL

Solution: Check from different IP (hard!)
Connectivity Properties

- 2/3 of the nodes show “bad connectivity”
- Clear overlap: transitivity & persistence
  - Quarantine could be “good enough”
Sub-Second Lookups in Mainline DHT
Measuring performance
Measuring performance
μTorrent

- 60% of Mainline DHT nodes
- Median lookup latency: \( \sim 650 \text{ ms} \)
- But …
  - \( > 1 \text{ s} \) 27% of lookups
Can we do better?

(while keeping backward-compatibility)
Our MainlineDHT nodes

- Modular architecture
  - **Lookup** module
  - **Routing table management** module
Lookup Parameters

- **Standard** lookup module
  - Same as μTorrent

- **Aggressive** lookup module
  - More parallel queries
  - [Stutzbach-06] [Steiner-09]
Routing Table Management

- Handle *connectivity issues* (quarantine)
- **Continuous** refresh
- Prefer *low-latency* neighbors
- Enlarge *most frequently used* buckets
Performance comparative

- µTorrent + 8 nodes
- 3078 lookups/node
- 80+ hours, 11+ GB
- 4.4 M unique IPs
- Very consistent results
Lookup Latency

Median: 647 ms
75\textsuperscript{th} perc.: \sim 1 s

Median: 164 ms
99\textsuperscript{th} perc.: 566 ms
Scalability & Locality
Scalability and Locality

My nodeID 001100...

Lookup target 11000...

0 1 3 5 4 2
Scalability and Locality

My nodeID: 001100...

Lookup target: 11000...
Scalability and Locality

My nodeID 001100...

Lookup target 11000...
Scalability and Locality

Are nodes overloaded in Mainline DHT?

Can we apply this approach to Mainline DHT?
</main results>
<conclusion>
Contributions

- Connectivity properties
- Sub-second lookups
- Scalability and Locality
- Source code
Future Work

- Integrate scalability and locality into MDHT
- Privacy and security
Future Work

- Integrate scalability and locality into MDHT
- Privacy and security

My nodeID
001100...

Lookup target
11000...
</conclusion>
Thank you!

http://people.kth.se/~rauljc/lic/