Non-Transitive Connectivity and DHTs

Mike Freedman
Karthik Lakshminarayanan
Sean Rhea
Ion Stoica

WORLDS 2005
Distributed Hash Tables…

- System assigns keys to nodes
- All nodes agree on assignment
- Chord assigns keys as integers modulo $2^{160}$
- Assigns keys via successor relationship
- Each node must know predecessor
Distributed Hash Tables…

- Used to store and retrieve (key, value) pairs
- Any node can discover key’s successor, yet without full knowledge of network
  - Implies some form of routing
Distributed Hash Tables…

- All have implicit assumption: full connectivity
Distributed Hash Tables...

- All have implicit assumption: full connectivity
- Non-transitive connectivity (NTC) not uncommon

\[ B \leftrightarrow C \ , \ C \leftrightarrow A \ , \ A \leftrightarrow B \]

- A thinks C is its successor!
Does non-transitivity exist?

- Gerding/Stribling PlanetLab study
  - 9% of all node triples exhibit NTC
  - Attributed high extent to Internet-2

- Yet NTC is also transient
  - One 3 hour PlanetLab all-pair-pings trace
  - 2.9% have persistent NTC
  - 2.3% have intermittent NTC
  - 1.3% fail only for a single 15-minute snapshot

- Level3 ↔ Cogent, but Level3 ↔ X ↔ Cogent

- NTC motivates RON, Detour, and SOSR!
Our contributions

- We have built and run Bamboo (OpenDHT), Chord (i3), Kademlia (Coral) for > 1 year

- Vanilla DHT algorithms break under NTC

- Identify four main algorithmic problems and present our solutions
Our goals

■ Short-term
  ■ Inform other developers about NTC solutions
  ■ Important: DHTs are being widely deployed in Overnet, Morpheus, and BitTorrent

■ Long-term
  ■ Encourage new designs to directly handle NTC
  ■ (This topic is far from solved)
Key space defines an identifier distance

Routing ideally proceeds by halving distance to destination per overlay hop
DHTs 101: Routing

Iterative

Recursive
DHTs 101: Routing tables

- successors / leaf set: ensure correctness
- fingers / routing table: efficient routing
  - $O(\log(n))$ hops, generally
Problems we identify

- Invisible nodes
- Routing loops
- Broken return paths
- Inconsistent roots
NTC problem fundamental?

Traditional routing

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>S → R</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>A → R</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>B → R</td>
<td>R</td>
<td></td>
</tr>
</tbody>
</table>
DHTs implement greedy routing for scalability

Sender might not use path, even though exists: finds local minima when id-distance routing

<table>
<thead>
<tr>
<th>Traditional routing</th>
<th>Greedy routing</th>
</tr>
</thead>
<tbody>
<tr>
<td>S → R</td>
<td>S → R</td>
</tr>
<tr>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>A → R</td>
<td>A → R</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>B → R</td>
<td>C → R</td>
</tr>
<tr>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
Problems we identify

- Invisible nodes
- Routing loops
- Broken return paths
- Inconsistent roots

(First discuss how problems apply to iterative routing, then consider recursive routing.)
Iterative routing: Invisible nodes

- Invisible nodes cause lookup to halt
Iterative routing: Invisible nodes

- Invisible nodes cause lookup to halt
- Enable lookup to continue
  - Tighter timeouts via network coordinates
  - Lookup RPCs in parallel
  - Unreachable node cache
Many proposals for maintaining routing tables
- E.g., replace nodes with larger RTT

Must first prevent routing table pollution
- Only add new nodes upon contacting *directly*
- Do not immediately remove nodes from hearsay
Inconsistent roots

- Nodes do not agree where key is assigned: inconsistent views of root
  - Can be caused by membership changes
  - Also due to non-transitive connectivity
    - May persist indefinitely
Inconsistent roots

- No solution when network partitions
- If non-transitivity is limited:
  - Consensus among leaf set?
    - [Etna, Rosebud]
    - Expensive in messages and bandwidth
  - Link-state routing among leaf set?
    - [Pastry 1.4.1]
- Can use application-level solutions!
Inconsistent roots

- Root replicates (key, value) among leaf set
  - Leaflets periodically synchronize
  - Get gathers results from multiple leaflets
  - [OpenDHT, DHash]
- Not applicable when require fast update (i3)
Recursive routing

- Invisible nodes
  - Must also prevent routing table pollution
  - Easier to achieve accurate timeouts
  - Harder to perform concurrent RPCs

- Inconsistent Roots
  - Similar solutions

- (Routing Loops)

- One new problem…
Broken return paths

- Direct path back from R to S fails
  - Source-route reverse path
  - Use single intermediate hop
    - RON, Detour, SOSR…
Summary

- Non-transitive connectivity exists
  - DHTs must deal with it

- Discovered problems the “hard way”
  - OpenDHT / Bamboo, i3 / Chord, Coral / Kademlia
  - Presented our “from the trenches” fixes

- NTC should be considered during design phase
Thanks…

Watch Our Real, Large Distributed Systems…

coralcdn.org
opendht.org
i3.cs.berkeley.edu