A Structured Overlay for Non-uniform Node Identifier Distribution Based on Flexible Routing Tables

Takehiro Miyao, Hiroya Nagao, Kazuyuki Shudo

Tokyo Tech
Background: Structured Overlay

- An application-level network
  - routes a query to the responsible node.
  - enables scalable data store and messaging.

  - e.g. Distributed Hash Tables (DHT)

<table>
<thead>
<tr>
<th>Index range (digest)</th>
<th>Responsible node</th>
</tr>
</thead>
<tbody>
<tr>
<td>ab – dz</td>
<td>192.168.0.2</td>
</tr>
<tr>
<td>ea – gb</td>
<td>192.168.0.3</td>
</tr>
<tr>
<td>gc – …</td>
<td>192.168.0.4</td>
</tr>
</tbody>
</table>

“Shudo”’s tel #?  

“+81 3 5734 XXXX”
Contribution

• A routing algorithm **FRT-Chord#**
  – supports non-uniform node ID distribution.
    • Range queries require it.
    • by **Chord#** [Schütt 2008] -inspired routing table maintenance.

  – has features existing overlays do not provide.
    • Extensibility, arbitrary routing table size, and one-hop property.

• by **Flexible Routing Tables (FRT)** [Nagao 2011] -based design.
Non-uniform node ID distribution

- Traditional structured overlays
  - Node and data ID are generated with a hash function such as SHA-1.
  - Nodes in a routing table are selected based on node IDs.

- To support range queries
  - Data are not hashed. Otherwise a query involves almost all nodes.
  - Load imbalance is caused.

Data Node Self ID space

Data Node Self location, time, temperature, … ID space
Non-uniform node ID distribution

• To support range queries
  – 1) Virtual nodes
  – 2) Making a node ID distribution follow a data ID distribution

– But a non-uniform node ID distribution leads larger hop numbers / longer path length.
Non-uniform node ID distribution

- **Node order** based routing table maintenance
  - **Chord#** [Schütt 2008] does it.
  - cf. Node ID based
  - Efficient lookups = smaller hop numbers / shorter path length by having enough number of pointers to dense areas.

  ![Data Node Diagram]

- Our algorithm **FRT-Chord#** adopts it.

We designed a **Flexible Routing Table (FRT)** based algorithm that perform it. described in next pages
Flexible Routing Tables (FRT) [Nagao 2011]

• A unified framework for structured overlays.
  – A methodology to design a routing algorithm

Ad-hoc extensions to each algorithm

Conflict
Conflict

Extensions

DHT algorithms

One hop
Proximity
Node group

One hop
Proximity
Node group

One hop
Proximity
Node group

One hop
Proximity
Node group

Flexible Routing Tables (FRT)

Arbitrary combination

One hop
Proximity
Node group

Ring (Chord)
XOR distance (Kademlia)

ID distance, topology

Designed without essence recognition

Algorithm characteristics and general actions are separated
Flexible Routing Tables (FRT)

[Nagao 2011]

• Declarative algorithm definition and common actions are separated.

• A routing table is just a list of entries.

• Algorithm definition

  an algorithm designer provides

  $\leq_{RT}$ A total order on the set of all routing table patterns
  Better is higher. “Better” means smaller hop numbers / shorter path length.

  Sticky entries
  Routing table entries not to be removed from the table.
  E.g. successor in Chord

• Common actions

  FRT provides

  – Entry learning A node notices another node and put it to the table.
  – Entry filtering A table overflows, an entry is selected and removed.
Flexible Routing Tables (FRT) [Nagao 2011]

- **FRT-based algorithms**
  - FRT-Chord [Nagao 2011]
  - FRT-2-Chord [Ando 2014]
  - FRT-XOR, that borrows ID space and distance from Kademlia
  - FRT-Chord# (this paper)

- **Extensions**
  - Proximity-aware FRT (PFRT) [Miyao 2013]
  - Grouped FRT (GFRT)
  - Virtual Node Fusion (VNF)

- **Features of FRT**
  - **Extensibility**
    - Algorithms and extensions can be combined arbitrarily.
  - **Arbitrary routing table size**
  - **One-hop property**
    - A query reaches the responsible node in one-hop if \# of nodes \( \leq \) the routing table size.
    - Note that FRT-Chord# itself does not perform one-hop lookup, but 2-hop, that is lowest and the same as Chord and Chord#.

- FRT-Chord# achieves efficient lookups with non-uniform ID distribution while providing the features of FRT.
Evaluation

• Goals: to confirm that
  – Path length does not get longer even with non-uniform node ID distributions
  – FRT-Chord# retains features of FRT

• Compared with Chord and FRT-Chord

• Configuration
  – Routing table size: 16, determined to be fair with Chord
  – Distributed environment emulator of Overlay Weaver 0.10.1
  – Java SE 6 Update 22
  – Linux 2.6.35
Path lengths do not depend on node ID distributions.

- Path lengths get longer for Chord and FRT-Chord as the node ID distribution changes from uniform to Zipf with \( \alpha = 0.7 \) and \( \alpha = 0.95 \).
- FRT-Chord# maintains a constant average path length regardless of the node ID distribution.

# of nodes: 10,000
Node ID distributions and path length

• Zipf distribution with $\alpha = 0.95$

- Chord shows shorter path length.

- FRT-Chord# shows shorter path length.

# of nodes: 10,000

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Average path length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chord</td>
<td>8.30</td>
</tr>
<tr>
<td>FRT-Chord</td>
<td>8.50</td>
</tr>
<tr>
<td>FRT-Chord#</td>
<td>6.98</td>
</tr>
</tbody>
</table>
Node ID distributions and path length

- Uniform distribution

<table>
<thead>
<tr>
<th></th>
<th>Chord</th>
<th>FRT-Chord</th>
<th>FRT-Chord#</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average path length</td>
<td>7.21</td>
<td>6.76</td>
<td>6.97</td>
</tr>
</tbody>
</table>

- Comparable with existing algorithms.

# of nodes: 10,000
Arbitrary routing table size

Larger tables show shorter path lengths.
FRT-Chord# retains this feature: arbitrary ...

A table holds all the nodes.

One-hop property FRT provides
Minimum path length of Chord-derived algorithms is 2.

# of nodes
○ N = 10000
△ N = 1000
□ N = 100

Average path length
Routing table size

0 2 4 6 8
0 50 100 150 200

N = 10000
N = 1000
N = 100

A table holds all the nodes.
Summary

• **FRT-Chord** is a routing algorithm for structured overlays
  – supports non-uniform node ID distributions
    • **Range queries** require this feature.
  – designed along Flexible Routing Tables (FRT)
    • Features: **extensibility**, **arbitrary routing table size**, and **one-hop** property