Introduction

Types of P2P (unstructured):

• Without an index
  Example: Gnutella
  Flood the network (or a subset of it)
  (+) simple and robust
  (-) enormous cost

• With specialized index nodes (centralized search)
  To find a document, query an index node
  Indices may be built through cooperation (as in Napster where nodes register (publish) their files at sign-in time) or by crawling the P2P network (as in a web search engine)
  (+) lookup efficiency (just a single message)
  (-) vulnerable to attacks (shut down by a hacker attack or court order)
  (-) difficult to keep up-to-date

TOPIC OF THIS PAPER

Routing indices

A. Crespo & H. Garcia-Molina ICDCS 02

Introduction: DISTRIBUTED INDICES

Should be small

Routing Indices (RIs): give a "direction" towards the document

In Fig 1, instead of storing (x, C) we store (x, B): the "direction" we should follow to reach X

Figure 1: Routing Indices

The size of the index, proportional to the number of neighbors instead of the number of documents

Further reduce by providing "hints"

System Model

• Each node is connected to a relatively small set of neighbors
• There might be cycles in the network

Content Queries: Request for documents that contain the words "database systems"

Each node local document database

Local index: receives the query and returns pointers to the (local) documents with the requested content

Figure 2: P2P Example
Query Processing

Users submit queries at any node with a stop condition (e.g., the desired number of results)
Each node receiving the query
1. Evaluates the query against its own local database, returns to the user pointers to any results
2. If the stop condition has not been reached, it selects one or more of its neighbors and forwards the query to them (along with some state information)

Routing Indices

Motivation: Allow to select the "best" neighbor to send a query to

A routing index (RI) is a data structure (and associated algorithms) that given a query returns a list of neighbors ranked according to their goodness for the query

Goodness in general should reflect the number of matching documents in "nearby" nodes

Routing Indices

(reminder) a CRI (compound RI) contains:
(i) the number of documents along each path
(ii) the number of documents on each topic of interest

Example CRI for node A (assuming 4 topics)
Routing Indices

• Computing the goodness

Use the number of documents that may be found in a path.

Use a simplified model:

queries are conjunctions of subject topics

Assumptions (i) documents may have more than one topic and (ii) document topics are independent

Let the query: \( \land R_1 \)

\[ \text{Number of Documents} \times \prod_i \text{CR}(s_i) / \text{Number of Documents} \]

Routing Indices

• Computing the goodness (example)

Let the query DB \( \land L \)


<table>
<thead>
<tr>
<th>Pub</th>
<th>Documents with topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>100 ( \times ) 20/100 ( \times ) 30/100 = 6</td>
</tr>
<tr>
<td>C</td>
<td>100 ( \times ) 0/100 ( \times ) 50/100 = 0</td>
</tr>
<tr>
<td>D</td>
<td>200 ( \times ) 0/100 ( \times ) 150/200 = 75</td>
</tr>
</tbody>
</table>

Note that these are “estimations”:

- If there is correlation between DB and L, path B may contain as many as 20 matching documents.
- If there is strong negative correlation between DB and L, path B may contain no documents on either topic.

Routing Indices

Using Routing Indices

Let A receive a query on DB and L

1. Use the local database
2. If not enough answers, compute goodness of B (6), C (0), D (75) – Select D
3. Forward query to D

Node D

1. Use the local database, returns all local results to A
2. If not enough answers, it cannot forward the query further
3. Returns the query to D (backtracks)

Node D selects the second best neighbor J.
Using Routing Indices
Lookup Savings
Assume a query with stop condition of 50 documents
Flooding: 9 messages
RI: 3 messages

![Routing Indices](image)

Creating Routing Indices
Assume initially no connection between A and D

- (step 1) A must inform D of all documents that can be accessed through node A
- (step 2) Similarly, D must inform A of all documents that can be accessed through node D

How?

Creating Routing Indices (continued)
Step 1: A informs D
A aggregates its RI and sends it to D
How: A adds all documents in the RI per column (i.e., topic)
E.g., 300 + 100 + 1000 = 1400 documents, 30 + 20 + 0 = 50 on DB, etc

Creating Routing Indices (continued)
Step 2: Similarly, D informs A
D aggregates its RI and sends it to A (excluding the row on A, if it is already there)
Again, D adds all documents in the RI per column (i.e., topic)
E.g., 100 + 50 + 50 = 200 documents, 60 + 25 + 15 = 100 on DB, etc

Creating Routing Indices (continued)
Step 1: A informs D
D updates its RI with information received by A
How: D adds a new row for A

Creating Routing Indices (continued)
Step 2: Similarly, D informs A
D aggregates its RI and sends it to A (excluding the row on A, if it is already there)
Creating Routing Indices (continued)
Step 2: D informs A
A updates its RI with information received by D
How: A adds a new row for D

Creating Routing Indices (continued)
Assume initially no connection between A and D

- step 1: A informed D of all documents that can be accessed through node A
- step 2: Similarly, D informed A of all documents that can be accessed through node D

Is this enough?
Step 3: A and D need also inform their other neighbors

Creating Routing Indices (continued)
Step 3: D sends an aggregation of its RI to I (excluding I’s row) and to J (excluding J’s row)
I and J update their RI, by replacing the old row of D with the new one

Maintaining Routing Indices
Similar to creating new indices.
Two cases:
- A node changes its content (e.g., adds new documents)
- A node disconnects from the network

Case 1: Assume node I introduces two new documents on topic L
Node I updates its local index
Aggregates all the rows of its compound RI (excluding the row for D) and send this information to D
Then D replaces the old row for I.
D computes and sends new aggregates to A and J
And so on

Case 1: Assume node I introduces two new documents on topic L
- Batch several updates
  Trade RI freshness for a reduced update cost
- Do not send updates when the difference between the old and the new value is not significant
  Trade RI accuracy for a reduced update cost
Maintaining Routing Indices

Case 2: node I disconnects from the network

D detects the disconnection
D updates its RI by deleting I’s row from its RI
D computes and sends new aggregates to its neighbors

In turn, the neighbors updates their RIs and propagate the new information

Note: Node I did not need to participate in the update

Alternative Routing Indices

Motivation:
The main limitation of the compound RI is that it does not take into account the "number of hops" required to find documents

Hop-Count RIs

Store aggregate RIs for each hop up to a maximum number of hops, called the horizon of the RI

Alternative Routing Indices; Hop-Count RI’s

Example: Hop-count index of horizon 2 hops for node W

<table>
<thead>
<tr>
<th>Node</th>
<th>1Hop</th>
<th>2Hops</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#DBs</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>100</td>
<td>5</td>
</tr>
<tr>
<td>Y</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>Z</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>70</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>29</td>
</tr>
</tbody>
</table>

If we define cost in terms of messages
Ratio: Number of documents / messages

Select the neighbor that gives the best number of results per message

Assume a simple model: regular tree cost model
(i) Documents are uniformly distributed across the network,
(ii) the network is a regular tree with fanout F

Then, it takes F^h messages to find all documents at hop h
Divide the expected number of result documents at each hop by the number of messages needed to find them

\[ \sum_{h=0}^{H} \text{goodness}(N[l], Q)/F^{h-1} \]
Alternative Routing Indices: Hop-Count RI's

Let \( F = 3 \), and query for DB

**Goodness for X**
\[
\frac{13}{1} + \frac{10}{3} = 16.33
\]
**Goodness for Y**
\[
\frac{0}{1} + \frac{31}{3} = 10.33
\]

<table>
<thead>
<tr>
<th>Node</th>
<th>1 Hop</th>
<th>2 Hops</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>30</td>
<td>3</td>
</tr>
<tr>
<td>B</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>A</td>
<td>5</td>
<td>2</td>
</tr>
</tbody>
</table>

**Motivation, solve the overhead of Hop-RIs:**
- Increased storage and transmission cost of hop-count RIs
- Limited by the horizon

**Trade accuracy**
One row per path, add together all reachable (!)
\[
\sum_{j=0}^{n} \text{goodness}(N[j], Q)/F^j
\]

\( \text{th height, F fanout of the assumed tree} \)

Alternative Routing Indices: Exponentially Aggregated RI

Weighted sum
For example for path Z and topic N
\[
0 + \frac{40}{3} = 13.33
\]

Cycles in the P2P network

- **This creates problems with updates.**
- For example, assume that node A adds two new documents in its database. When node A receives the update through node C, it will mistakenly assume that more documents are available through node C. Worst, it will propagate this update further.

- **Do Nothing Solution**

  Cycles are not as "bad" with hop-count and exponential RIs
  - Hop-count cycles longer than the horizon will not affect the RI
  - Will stop if we use the regular-tree cost model
  - Exponential RI
    - The effect of the cycle will be smaller and smaller every time the update is sent back (due to the exponential decay)
    - the algorithm will stop propagating the update when the difference between the old and the new update is small enough again, increased cost of creating/updating the RI.
Performance

Compare:

CRI
Hop-Count RI (HRI)
Exponential RI (ERI)
No RI (select one neighbor randomly)

Need to define:

(i) The topology of the network, and
(ii) The location of document results (how documents are distributed)

Cost of the search: number of messages

Experiment 1: Evaluating P2P Search Mechanisms

Compare:

CRI
Hop-Count RI (HRI)
Exponential RI (ERI)
No RI (select one neighbor randomly)

Performance: Network Topologies

1. A tree
2. A tree with added cycles
3. A power-law graph

Performance: Document Results

1. Uniform distribution
2. 80/20 biased distribution

Experiment 1: Comparison of RIs for different document distributions

Experiment 2: Errors (overcounts) in RIs

Categories grouped together
How: Several categories may be hashed to the same bucket
Count in a bucket represents the aggregate number of documents in these categories
A 50% “index compression” means that the number of hash table buckets is half the number of categories, while 83%, 1/6

Experiment 3: Cycles and ERIs

Increase of traffic for two reasons:
1. Loss of accuracy of the RI (detect and recover) we may lose the best route to results (no-op) due to overcounts
2. Increase of number of messages during query processing (detect and recover) to detect cycles (no-op) visit the same nodes

Adding many links – added connectivity, better routes

Note: number of nodes: 600000
Experiment 4: Different Network Topologies

RI's perform better in power-laws
1. Queries are directed towards the well-connected nodes
2. Average path length is lower than in the tree topology
3. No RI
   - Difficult to find the few well-connected nodes
   - Shortest path makes bad decisions on neighbors result in no-result

Experiment 5: Update Cost

1032 queries per minute
Total cost of ERI better of no-RI if less than 36 updates per minute

Open Questions

How can we avoid cycles without losing "good" paths?
- Caching