Topics in Database Systems: Data Management in Peer-to-Peer Systems

Routing indexes

A. Crespo & H. Garcia-Molina ICDCS 02

Introduction

P2p exchange documents, music files, computer cycles

Goal: Find documents with content of interest

Types of P2P (unstructured):

- Without an index
- With specialized index nodes (centralized search)
- With indices at each node (distributed search)

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
- o through *cooperation* (as in Napster where nodes register (publish) their files at sign-in time) or
- o 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

Introduction

Types of P2P (unstructured):

- With indices at each node (distributed search)

TOPIC OF THIS PAPER

Introduction: DISTRIBUTED INDICES

Should be small

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

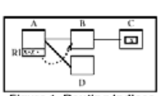


Figure 1. Routing Indices

In Fig 1, instead of storing

(x, C)

we store

(x, B): the "direction" we should follow to reach X

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

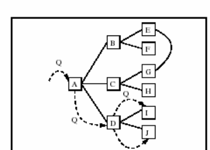


Figure 2. P2P Example

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 be reached, it *selects one or more of its neighbors* and forwards the query to them (along with some state information)

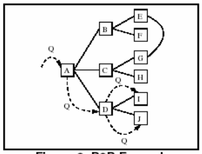


Figure 2. P2P Example

Query Processing (continued)

Queries may be forwarded to the best neighbors in parallel or sequentially

In parallel: better response time but higher traffic and may waste resources

In this paper, sequentially

Compare with BFS and DFS

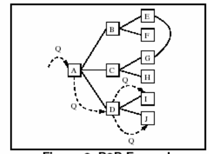


Figure 2. P2P Example

Routing Indices

Motivation:

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

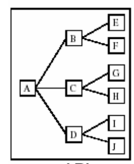
A routing index (RI) is a data structure (and associated algorithms) that

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

P2P system used as example:



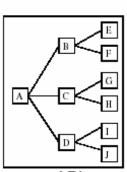
- Documents are on zero or more topics
- Query requests documents on particular topics
- Each node:
  - a local index and
  - a CRI (compound RI) that contains
    - (i) the number of documents along each path
    - (ii) the number of documents on each topic of interest

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)



Path	# docs	Documents with topics: DB	N	T	L
B	100	20	0	10	30
C	1000	0	300	0	50
D	200	100	0	100	150

Routing Indices

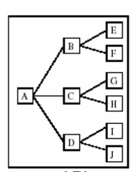
- The RI may be "coarser" then the local index

For example, node A may maintain a more detailed local index, where documents are classified into *sub-categories*

Such summarization, may introduce *undercounts* or *overcounts* in the RI

Examples: overcount (a query on SQL)

undercount (when there is a frequency threshold)



Path	# docs	Documents with topics: DB	N	T	L
B	100	20	0	10	30
C	1000	0	300	0	50
D	200	100	0	100	150

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: $\wedge\, s_i$

NumberofDocuments x $\prod_i$ CRI(s_i)/NumberofDocuments

Routing Indices

- Computing the goodness (example)

Path	# docs	Documents with topics: DB	N	T	L
B	100	20	0	10	30
C	1000	0	300	0	50
D	200	100	0	100	150

Let the query DB ∧ L

Goodness for B

100 x 20/100 x 30/100 = 6

Goodness for C

1000 x 0/1000 x 50/100 = 0

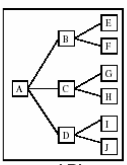
Goodness for D

200 x 100/200 x 150/200 = 75

Note that this are "estimations"

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

Using Routing Indices



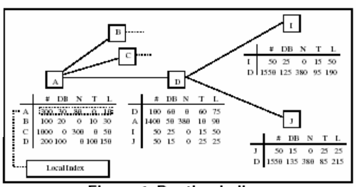


Figure 4. Routing Indices

Assume that the first row of each RI contains a summary of the local index

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

D repeats 1-2-3

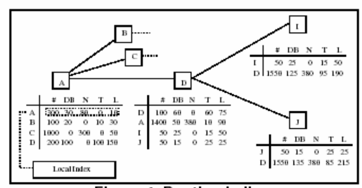


Figure 4. Routing Indices

Using Routing Indices (continued)

Node D

1. Use the local database, returns all local results to A
2. If not enough answers, compute goodness of I (=25), J (=7.5) , – Select I
3. Forward query to I

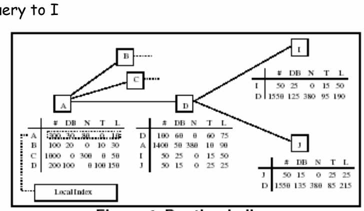


Figure 4. Routing Indices

Using Routing Indices (continued)

Node I

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

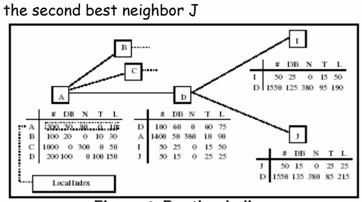


Figure 4. Routing Indices

Using Routing Indices

Lookup Savings

Assume a query with stop condition of 50 documents

Flooding: 9 messages

RI: 3 messages

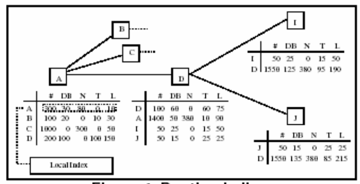


Figure 4. Routing Indices

Using Routing Indices

Storage space

s: counter size in bytes

c: number of categories

N: number of nodes

b: branching factor (number of neighbors)

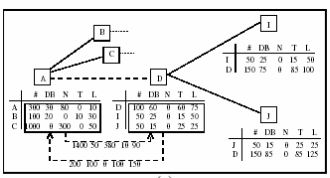
Centralized index c x (t+1) x N

Each node c x (t+1) x b

Total c x (t+1) x b X N

Creating Routing Indices

Assume initially no connection between A and D



(a)

- (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?

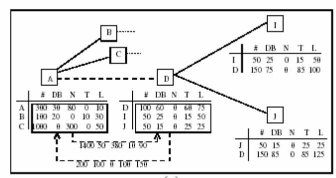
Creating Routing Indices (continued)

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



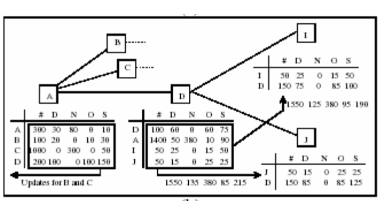
(a)

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



(b)

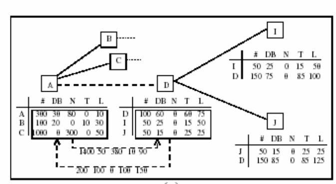
Creating Routing Indices (continued)

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



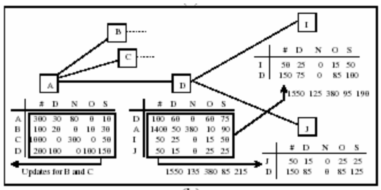
(a)

Creating Routing Indices (continued)

Step 2: D informs A

A updates its RI with information received by D

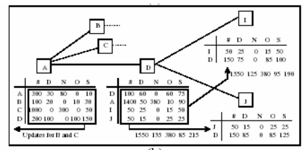
How: A adds a new row for D



(b)

Creating Routing Indices (continued)

Assume initially no connection between A and D



(b)

- 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

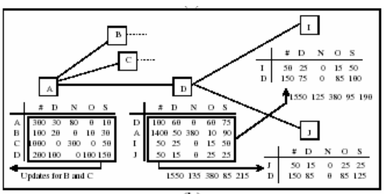
Is this enough?

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



(b)

Note, if I and J were connected to nodes other then D, they would have to send an update to those nodes as well

Maintaining Routing Indices

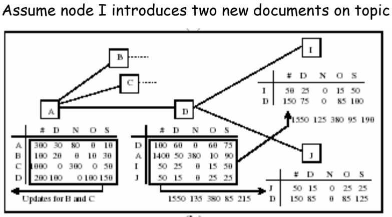
Similar to creating new indices.

Two cases:

- A node changes its content (e.g., adds new documents)
- A node disconnects from the network

Maintaining Routing Indices

Case 1: Assume node I introduces two new documents on topic L



(b)

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

Maintaining Routing Indices

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

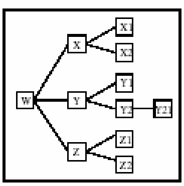
Motivation:

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



Node	1 Hop #	DB	N	T	L	2 Hops #	DB	N	T	L
X	60	13	2	5	10	20	10	10	4	17
Y	30	0	3	15	12	50	31	0	15	20
Z	5	2	0	3	3	70	10	40	20	50

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

Alternative Routing Indices; Hop-Count RI's

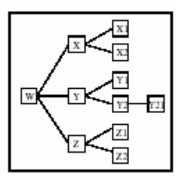
We need a new estimator for the goodness of a neighbor

Assume we have a query on topic DB

Node X gives us 13 documents in one hop, and 23 in two hops

Node Y gives us 0 documents in one hop and 31 in two hops

Which one to choose?



Node	1 Hop #	DB	N	T	L	2 Hops #	DB	N	T	L
X	60	13	2	5	10	20	10	10	4	17
Y	30	0	3	15	12	50	31	0	15	20
Z	5	2	0	3	3	70	10	40	20	50

Alternative Routing Indices; Hop-Count RI's

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

Alternative Routing Indices; Hop-Count RI's

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

$\Sigma_{j=0..h}$ goodness(N[j], Q)/F^{j-1}

Alternative Routing Indices; Hop-Count RI's

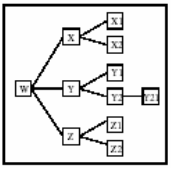
Let F = 3, and query for DB

Goodness for X

13/1 + 10/3 = 16.33

Goodness for Y

0 + 31/3 = 10.33



	1 Hop					2 Hops				
Node	#	DB	N	T	L	#	DB	N	T	L
X	60	13	2	5	10	20	10	10	4	17
Y	30	0	3	15	12	50	31	0	15	20
Z	5	2	0	3	3	70	10	40	20	50

Alternative Routing Indices: Exponentially Aggregated RI

Motivation, solve the overhead of Hop-RIs:

- Increased storage and transmission cost of hop-count RIs
- Limited by the horizon

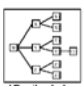
Trade accuracy

One row per path, add together all reachable (!)

$\sum_{j=0..th}$ goodness(N[j], Q)/$F^{j\ 1}$

th height, F fanout of the assumed tree

Alternative Routing Indices: Exponentially Aggregated RI



	1 Hop					2 Hops				
Node	#	DB	N	T	L	#	DB	N	T	L
X	60	13	2	5	10	20	10	10	4	17
Y	30	0	3	15	12	50	31	0	15	20
Z	5	2	0	3	3	70	10	40	20	50

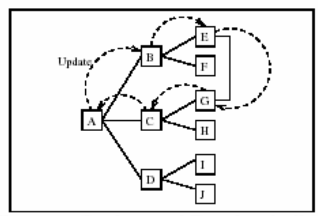
Path	#	DB	N	T	L
X	66.67	16.33	5.33	6.33	15.67
Y	46.67	10.33	3.00	20.00	18.67
Z	28.33	5.33	13.33	9.67	19.67

Weighted sum

For example for path Z and topic N

0 + 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

Cycles in the P2P network

- Cycle detection and recovery

Let the originator of an update or a query include a unique message identifier in the message

If a message with the same identifier returns to a node, then it knows, there is a cycle and can recover

- Cycle avoidance solutions

We may end-up with a non-optimal solution

Cycles in the P2P network

- 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 propagate 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

Performance: Network Topologies

1. A tree
2. A tree with added cycles

   Start with a tree and add extra vertices at random
3. A power-law graph

Performance: Document Results

1. Uniform distribution

   All nodes have the same probability of having each document result
2. 80/20 biased distribution

   assigns 80% of the documents uniformly to 20% of the nodes

   and the remaining 20% of the documents to the remaining 80% of the nodes

Experiment 1: Evaluating P2P Search Mechanisms

Compare

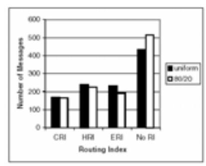
CRI

Hop-Count RI (HRI)

Exponential RI (ERI)

No RI (select one neighbor randomly)

Experiment 1: Comparison of RIs for different document distributions



- The difference in performance between the RIs is a function of the nodes used to generate the index
- 80/20 does not improve the performance of RIs much

Why? The queries were directed to nodes with a high number of documents results but to reach then passed through several nodes that had very few or no document results

For uniform: the queries were directed through good paths where at each node they obtained a few results

- 80/20 penalizes no-RI

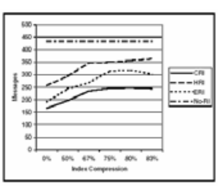
Experiment 2: Errors (overcounts) in RIs

Categories grouped together

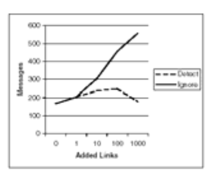
How: Several categories may be hashed to the same bucket

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



Note: number of nodes: 600000

Increase of traffic for two reasons:

1. Loss of accuracy of the RI

   (detect and recover) we may lose the best route to results

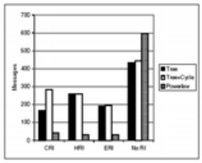
   (no-op) due to overcounts
2. Increase of number of messages during query processing

   (detect and recover) to detect cycles

   (no-op) visit the same nodes

Adding many links – added connectivity, better routes

Experiment 4: Different Network Topologies



RIs perform better in power-laws

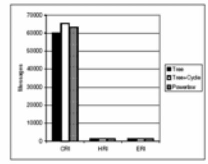
1. Queries are directed towards the well-connected nodes
2. Average path length is lower than in the tree topology

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