Routing Indices
For Peer-to-Peer Systems

presented by
Wei Xie
University of Toronto

Outline

- Introduction to Peer-to-Peer System
- Routing Indices
- Cycles in Peer-to-Peer Network
- Parallel Alternative of Routing Indices
- Conclusion
Peer-to-Peer System

- Formed by a large number of nodes that can join or leave the system at any time
- Distributed computing nodes of equal roles or capabilities exchange information directly with each other

Famous P2P Systems

- Napster
- Gnutella
- Freenet
- GIOSS

Mechanism of Searching in P2P system

- mechanisms without an index
  - flooding network with queries
- mechanism with specialized index node
  - index node becomes bottleneck
- mechanism with indices at each node
  - routing indices
- mechanism with super nodes
  - every node stores metadata about its neighbors within depth n
Routing Indices

Select the best neighbors to send query to
Key factor is how to define and compute the goodness suitable to a specific query

- Compound Routing Index (CRI)
- Hop-count Routing Index (HRI)
- Exponentially Aggregated Routing Index (ERI)

**Compound RI**

- summary of detailed index
- compact data structure
- goodness: the number of documents that could be found in a path
- send query in depth-first fashion
- savings in the number of messages
- modest storage space
- introduce errors of overcount or undercount

- maintenance?
- effect of errors?
Hop-count RI

- take into account of numbers of hops
- regular-tree model to compute goodness
- need local network structure to decide value of horizon
- higher storage and transmission cost
- potential loss in accuracy
- complicated maintenance algorithm

\[ \text{goodness}_{\text{hop-count}}(\text{Neighbour}, Q) = \sum_{a \in \text{all}} \text{Hop}(a) \]

Figure 6. A sample Hop-count RI for node W

Exponential RI

- keep info. for all nodes accessible from each path
- dynamic data structure
- need complete information of local network
- unrealistic assumptions of regular tree model

Figure 7. A sample Exponential Routing Index for Node W

Regular Tree
Algorithms for creating/update RI, answering queries

```
// Creation phase
RI[R] = Normalize(R)
For each neighbor j:
Send(j, RI[j])

// Update phase
oldRI = RI
While (true) Wait for a batch of aggregated RI to arrive and/or
For the local index: Update
For each aggregated RI of any, A_k, received from neighbor in
RI[R] = Normalize(A_k)
If local index changed:
oldRI = Normalize( )
If oldRI is different enough than RI:
for each neighbor j in RI:
A = Normalize(R)(j, j) - 1 + k
Send(A, oldRI)
```

Figure 7: RI Usage Algorithm

Effect of errors

assumption:
- documents are organized into categories
- index is hash table of categories

results show:
- loss is modest
- similar results to undercounts and mixed errors

Figure 16: Effect of Overcounts
Cycles in the P2P System

- No-op
- Cycle avoidance
- Cycle detection and recovery

signature: unique message id
ignore some update messages

Performance Evaluation

Topology
- Tree
- Tree with added cycles
- Power-law graph

Data Distribution
- Uniform
- 80/20
Parallel Alternative of Routing Indices

for each node i in data structure
if | goodness$_{highest}$ - goodness$_i$| <= threshold
{
    send Q to node i
    apply RI algorithm
}

threshold represents the possibility similarity
of finding results in different paths
value of threshold is decided by many factors:
number of results (stop condition)
response time limitation

Better response time!

Conclusion

- Propose simple and effective strategies
- Compact data structures and algorithms
- Do not need to know global network architecture
- Reduce the number of messages greatly
- Sequential algorithms, but can be changed to parallel ones easily
- Response time is worse than non-index systems
- Paper is easy to understand
Thanks for your patience!

Happy Halloween!