Lecture-3
Information Retrieval in P2P
(Part A)

Zhou Shuigeng

March 23, 2007
Overview(1)

- We discussed about some files sharing techniques in P2P network, such as Chord, CAN, Pastry, and Tapestry, these techniques:
  - Focusing on the availability of storage and archival systems
  - Guaranteeing location of content if it exists, within a bounded number of hops
  - Tightly controlling the data placement and topology within the network
  - Suitable to be used in a cooperating organizations and all computers are trusted computers
We also discussed contents sharing in "loose" P2P systems, such as Napster, Gnutella and Freenet:

- Without strictly controlling the data placement and topology of the networks
- Users from a wide ranges of non-cooperating and non-trusted organizations
- Support for richer (content or semantic based) queries than just identifier lookup
The purpose of IR in P2P is to accept queries from users, and locate and return data.

The desired features of P2P IR systems:
- High-quality query results
- Minimal routing state maintained per node
- High routing efficiency
- Load balance
- Resilience to node failures
- Support different retrieval forms
Overview (4)

- **Retrieval Performance Metrics**
  - Retrieval cost
    - Space cost: routing state maintained per node
    - Bandwidth cost: number of overlay hops per query, or number of messages per query
  - Results Quality:
    - Number of results, relevance, response time
      - recall, precision
Overview (5)

- Distributed IR vs. P2P IR
  - System decentralization and node autonomy
  - System (or network) scale
  - Node dynamism
  - Global meta-data
  - Query results
  - ......
Overview (6)

- P2P searching vs. Web searching
  - Web searching
    - B/S mode
    - distributed crawling, and centralized indexing
    - Searching is separated from services providing
  - P2P searching
    - Peer-to-Peer mode
    - Decentralized crawling and indexing
    - Searching while providing services
Overview(7)

- Different types of P2P systems employ different retrieval methods
  - Non-structured P2Ps
    - Gnutella
  - Structured P2Ps
    - Flat (non-hierarchical) DHT P2Ps
    - Hierarchical DHT P2Ps
    - Non-DHT P2Ps
  - Loosely structured P2Ps
    - Freenet, Power-law network, small-world network
Overview(8)

- Taxonomy of searching in unstructured P2Ps
  - Breadth First Search (BFS) and its variants vs. Depth First Search (DFS) and its variants
  - Deterministic vs. probabilistic
  - Regular-grained vs. coarse grained
  - Blind vs. informed
Overview (9)

- Napster: Single point of failure, hot-spot
  - Uses centralized indices server

- Gnutella: Query-flooding
  - Uses a breadth-first traversal (BFS) with limit $D$. Every node receiving a query will forward the message to all of its neighbors, unless the message has already traveled $D$ hops.

- Freenet: Response time is problematic
  - Uses a depth-first traversal (DFS) with depth limit $D$. Each node forwards the query to a single neighbor, and waits for a definite response from the neighbor before forwarding the query to another, or forwarding results back to the query source.
Random Walk

- **Standard random walk**: one walker’s random walk, a kind of random DFS
- **K-walker random walk**: paralleled random walk
- **Multi-level random walk**
  - Employ different random walk strategies at different levels: the number of walkers are different
Content

4. Dimitrios Tsoumakos and Nick Roussopoulos. Adaptive probabilistic search (APS) for Peer-to-Peer networks, IPTPS'03.
5. Arturo Crespo and Hector GarciaMolina. Routing Indices for Peer to Peer Systems. ICDCS'02
Improving search in Peer-to-Peer Networks

B. Yang and H. Garcia-Molina

Computer Science Department
Stanford University
(Appeared in ICDCS2002)
Contributions

- Find some middle ground between BFS and DFS, while maintaining quality of results
- Three techniques for efficient search in P2P systems were given
  - Iterative deepening
  - Directed BFS
  - Local Indices
- Experiments were conducted to evaluate these techniques
Iterative Deepening (1)

- Basic idea: multiple breadth-first searches are initiated with successively larger depth limits, until either the query is satisfied, or the maximum depth $D$ has been reached.
- A system-wide policy is required that specifies at which depths the iterations are to occur.
- The time between successive iterations in the policy are also must given.
An example: policy $P = \{a, b, c\}$. Time = $W$

- A source node $S$ first initiates a BFS of depth $a$.
- When a node at depth $a$ receives and processes the message, it will store the messages temporarily.
- The query therefore becomes frozen at all nodes that are $a$ hops from the source.
- After waiting for a time period $W$, if the query has been satisfied, $S$ does nothing more; otherwise $S$ will start the next iteration a BFS of depth $b$.

...
Directed BFS

- Basic idea: each peer send query messages to just a subset of its neighbors
- For selection neighbors intelligently
  - Peer maintains simple statistics on its neighbors, such as the number of results received through that neighbor for past queries, or the latency of the connection with that neighbor
  - Some heuristics are used,
    - Select the neighbor that has returned the highest number of results for previous queries
    - Select neighbor that returns response messages that have taken the lowest average number of hops
    - ...

Advanced Distributed Computing
Local Indices

- Basic idea: each node $n$ maintains an index over the data of all nodes within $r$ hops of itself. When a node receives a query message, it can process the query on behalf of every node within $r$ hops.

- A system-wide policy specifies the depths at which the query should be processed. All nodes at depths not listed in the policy simply forward the query to the next depth.
A Novel Strategy for Information Retrieval in the Peer-to-Peer Network

Cheuk Hang Ng, Ka Cheung Sia, Irwing King
Department of Computer Science and Engineering
The Chinese University of Hong Kong

A conference version of this paper is first published as poster paper in WWW02
Contributions

- Providing new routing and searching algorithm makes use of deliberately formed connection between peers and routing of queries intelligently to increase query performance without strict requirement on network topology and location of data placement.

- Technique features:
  - Peer clustering
  - Firework query model
Peer Clustering (1)

- Clustering peers based on the similarities of the contents in these peers.
- Two peers share (roughly) similar content are connected by additional attractive link.
- Therefore, peers with (roughly) similar content are connected together by attractive links, which results in peer clusters.
Peer Clustering (2)

Peer clustering illustration
Peer Clustering (3)

- Here we shift the application domain from image retrieval to text retrieval.
- A peer $p$ contains a number of documents, in which each document can be represented by VSM as a high-dimensional vector, and a peer can be represented by the **merged vector** of all documents vectors in the peer.
- Similarity of two $p$ and $q$ can be measured by the similarity between their corresponding vectors: $\text{Sim}(p, q) = \text{Sim}(\bar{p}, \bar{q})$. 
There are three steps in peer clustering:

- Peers vectors calculation
- Neighborhood discovery
  - When this peer $p$ joins the network, it will connect to another peer randomly chosen by the user. Through the ping-pong messages, it learns the vectors of the set of peers within a certain number of hops $t$ away from it (denoted as $\text{Peer}(p; t)$)
- Similarity calculation and attractive link establishment
  - Connect peer $p$ to a peer $q$ from $\text{Peer}(p; t)$ that has the largest $\text{Sim}(p; q)$ value through an attractive link
Peer Clustering (5)

Example of peer clustering
Firework Query Model (1)

- The aim of Firework Query Model is to reduce the query message traffic
- Basic Process:
  - a query message first walks around the network from peer to peer through random links, by doing this, the message is routed selective towards its target cluster and avoids from passing through peers containing irrelevant data
  - Once it reaches the designated cluster, the query message will be broadcasted by peers through attractive links insides the cluster
Firework Query Model (2)

The Algorithm for firework query model

Firework-query-routing (peer \( p \), query \( Q \))
1. if Sim\( (p; Q) \) >= threshold then
2. reply the query \( Q \) in \( p \)
3. \( \text{TTL}_{\text{new}}(Q) = \text{TTL}_{\text{old}}(Q) \)
4. forward \( Q \) to all attractive-link\( (p) \)
5. Else
6. \( \text{TTL}_{\text{new}}(Q) = \text{TTL}_{\text{old}}(Q)-1 \)
7. if \( \text{TTL}_{\text{new}}(Q) > 0 \) then
8. forward \( Q \) to all random-link \( (p) \)
9. endif
10. endif
Firework Query Model (3)
Extended Peer Clustering (1)

- **Local cluster discovery and cluster vector calculation**
  - In the beginning, every peer performs a local clustering operation on its own collection. This can be done by either K-means method or other methods; Evaluate the vector of each cluster in the peer.

- **Neighborhood discovery**
  - When this peer joins the network, it will connect to another peer randomly chosen by the user; Through the ping-pong messages, it learns the characteristic of the set of peers within a certain number of hops $t$ away from it ($\text{Peer}(p; t)$).

- **Similarity calculation and attractive link establishment**
  - For each cluster $i$ in the peer $p$, it will connect to another peer $q$ in $\text{Peer}(p; t)$ and having the largest $\text{Sim}(pi; qj)$ value through attractive links.
Illustration of extended peer clustering
LightFlood: An Efficient Flooding Scheme for File Search in Unstructured P2P Systems

Song Jiang, Lei Guo, and Xiaodong Zhang

(Appeared in ICPP’03)
Unstructured P2P Overlay

- P2P overlay
  - Application level network over physical network
  - Self-organized by peers voluntarily

- Characteristics
  - Power-law distribution: a small number of peers have high connectivity
  - Dynamic population: peers come and go frequently
  - Resilient to random node failures
Search in P2P Overlay

- Flooding (Gnutella)
- Expanding ring (ICS’02)
- Random walk (ICS’02, SIGCOM’03)
- Iterative deepening (ICDCS’02)
- Directed BFS (ICDCS’02)
- Super peer (ICDE’03)
- Interest of locality (INFOCOM’03)
Flooding

- Simple and robust
  - No state maintenance needed
  - High tolerance to node failures
- Effective and of low latency
  - Always find the shortest / fastest routing paths
- Fundamental operation for
  - Broadcasting in distributed systems
  - P2P communications
Problems of Flooding

- Loops in Gnutella networks
  - Caused by redundant links
  - Result in endless message routing
- Current solutions by Gnutella
  - Detect and discard redundant messages
  - Limit TTL (time-to-live) of messages
- Unnecessary traffic is still too much
  - The redundant links are still there
Traffic Minimization: Spanning Tree

- Reduce traffic without changing P2P overlay
- How much bandwidth can we save?
  - Average degree of Gnutella nodes: about 3 ~ 5
  - N-node spanning tree
    - N-1 links
    - N-1 messages for a broadcast
  - Estimated traffic reduction: about 67% ~ 80%
- Bandwidth efficiency is not the only objective!
Problems of Spanning Tree

- Long latency for flooding
  - More than 30 hops to cover 95% of nodes
  - Only 7 hops to cover 95% of nodes by Gnutella flooding
  - 5 times slower in a power law based topology

- Weak reliability due to node failures
  - A node failure can disconnect a large portion of network
P2P Overlay (non-power-law)
Flooding in Spanning Tree

HOPS = 0

HOPS = 1

HOPS = 2

HOPS = 3

HOPS = 4

HOPS = 5

HOPS = 6

HOPS = 7

HOPS = 8

HOPS = 9

HOPS = 10

HOPS = 11
Flooding in P2P Overlay
Node Failure
Trade-offs

- Traffic efficiency and routing latency
- Redundancy and robustness
- Flooding in Gnutella gives us some new thoughts.
Observations of Pure Flooding

Coverage Growth Rate

Coverage Increase (times)

Hop 2  Hop 3  Hop 4  Hop 5  Hop 6  Hop 7

Hop in 7-hop Flooding
Observations of Pure Flooding

Redundant Messages Distribution

Percentage of Redundant Messages

Hop in 7-hop Flooding
Motivations

- Pure flooding is efficient in the initial hops
  - Node coverage grows quickly, while
  - Only account for a small portion of redundant msgs
- Most redundant messages are generated in high hops with very low coverage growth rates.
Our Solution

- Combining both merits of pure flooding and spanning tree
- Constructing FloodNet: a tree-like structure over P2P network
- Flooding over P2P network in initial hops
- Flooding over FloodNet in rest hops
Outline

- Building a FloodNet to approximate a spanning-tree broadcast net.
- Analysis of the FloodNet.
- LightFlood protocol.
- Performance evaluation of the protocol.
- Using LightFlood as the infrastructure.
- Conclusion
FloodNet: a Tree-like Sub-overlay

- States maintained in each node
  - Number of neighbors
  - The node degree of each neighbor

- Topology construction
  - Father node: the neighbor with the highest degree
  - Dynamic updating: very low overhead

- A tree-like structure over Gnutella overlay
Constructing FloodNet
Constructing FloodNet
Constructing FloodNet
Property 1: Loop Elimination

- At most one loop in the structure
- Nodes in a loop have the same degree
  - Root candidates
- Endless routing
  - Easy to detect and avoid
- Redundant messages
  - At most one redundant message per flooding
Property 2: Multiple Trees?

- Possible but the number is very small
  - Only high degree nodes can be tree roots
  - Only a few nodes have high connectivity (recall the power law distribution)
  - These high degree nodes may connect each other
- Normally less than 10 trees in Gnutella overlay according to our simulation
LightFlood

- **Low hops: utilizing redundant links**
  - Flooding in P2P overlay
  - Reach many nodes of different trees with small overheads

- **High hops: keep away from redundant links**
  - Flooding in FloodNet
  - Flooding from multiple nodes in parallel
Notation of LightFlood

- 2-stage broadcasting
  - Low hops: the initial $M$ flooding hops
  - High hops: the rest $N$ flooding hops
  Denoted as $(M, N)$ policy
  $(7, 0)$ is same as Gnutella flooding
High hops
Performance Evaluation
Coverage vs. Latency

(4, *) takes only additional 3 hops to reach same coverage as (7, 0)
Traffic Efficiency

- (4, *): 90.8%
- (7, 0): 28.1%
Degradation Due to Node Failures

-Nearly same coverage

(4, 6) with randomly selected peers removal
(7, 0) with randomly selected peers removal
(4, 6) with best connected peers removal
(7, 0) with best connected peers removal
Conclusion

- FloodNet is easy to construct and maintain
  - Using local and neighboring information
  - Dynamically updated with little overhead
- LightFlood is both broadcast effective and bandwidth efficient
  - Large coverage
  - Small routing hops
  - Small amount of redundant messages
- An efficient and effective flooding scheme
Adaptive probabilistic search (APS) for Peer-to-Peer networks

(Appeared in IPTPS’03)

Dimitrios Tsoumakos
Nick Roussopoulos.
Search Model

- Objects are distributed across the network according to a replication distribution
- Each node is connected directly to its neighbors, and keep some soft state for each query the node process
- Each search is assigned an unique identifier
- $K$ walkers are deployed for each search from a requester node
- Evaluation metrics: success rate, number of discovered objects, the number of message and duplicate message produced
APS: the Algorithm(1)

- Each node keep a local index
  - Each entry corresponds to one object it has requested or forwarded a request for, per neighbor
  - Value of each entry indicates the relative probability of this node’s neighbor to be chosen as the next hop in a future request for the specific object
APS: the Algorithm(2)

The search process

- When a search is initiated at a certain node A, it chooses k out of its neighbors to forward the request to.
- For a chosen neighbor, say B, it first searches its local repository and if a hit occurs, this walker terminates successfully.
- Otherwise, B will forward the request to one of its neighbors.
- For node A, the search terminates when all the k walkers terminate.
APS: the Algorithm(3)

- **Neighbor selection for message forwarding**
  - Node chooses its next-hop neighbor(s) by using the probabilities given by its index values
  - When a node chooses one or k peers to forward the request to, it pro-actively increases/decreases the relative probability of the peer(s) it picked.
APS: the Algorithm(4)

- Two index updating schemes
  - Optimistic approach
    - When a walker is successful, the relative probability of the nodes on the path of forwarding will be increased
  - Pessimistic approach
    - When a walker fails, the relative probability of the nodes on the path of forwarding is decreased
An Example

<table>
<thead>
<tr>
<th>Indices</th>
<th>Initially</th>
<th>At walker termination</th>
<th>After index updates</th>
</tr>
</thead>
<tbody>
<tr>
<td>A→B</td>
<td>30</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>B→C</td>
<td>30</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>C→D</td>
<td>30</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>A→E</td>
<td>30</td>
<td>20</td>
<td>40</td>
</tr>
<tr>
<td>E→F</td>
<td>30</td>
<td>20</td>
<td>40</td>
</tr>
<tr>
<td>A→G</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
</tbody>
</table>
Algorithm Improvements

- **Swapping APS (s-APS)**
  - Switching between the optimistic and pessimistic approaches according to whether the success rate of walkers is or not greater than k/2

- **Weighted APS (w-APS)**
  - The updating amount of the relative probability is inversely proportional to the distance between the node and the requesting node
Experimental Results(1)

Success rate vs number of deployed walkers
Experimental Results (2)

Messages per query vs number of deployed walkers
Conclusions

- APS exhibits some characteristics
  - High accuracy
  - Low bandwidth consumption
  - Large number of discovered objects
  - Robust and adaptive behavior in rapidly-changing environments
Routing Indices for Peer to Peer Systems

Arturo Crespo and Hector Garcia Molina
(Appeared in ICDCS’02)

Computer Science Department
Stanford University
Routing Indices: the Concepts

- A 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.
- Allow nodes to forward queries to neighbors that are more likely to have answers.
Related Work

- P2P information search mechanisms
  - Searching without indices (Gnutella)
    - Queries flooding
    - Random walk
  - Searching with one specialized index node (centralized indices, Napster)
  - Searching with some specialized index nodes (centralized indices, Kazza & Morpheus)
  - Searching with indices at each nodes (distributed indices, Local indices, RIs)
About the Paper

- Introducing the concepts of routing indices
- Providing three RIs
  - compound RI
  - hop count RI
  - exponential RI
- Performance evaluation by simulation
Compound RI: A sample (1)

- Each node has a local index
- Nodes also have a CRI containing
  - the number of documents along each path
  - the number of documents on each topic of interest

<table>
<thead>
<tr>
<th>Path</th>
<th>Number of documents</th>
<th>Documents with topics:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Databases</td>
</tr>
<tr>
<td>B</td>
<td>100</td>
<td>20</td>
</tr>
<tr>
<td>C</td>
<td>1000</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>200</td>
<td>100</td>
</tr>
</tbody>
</table>
Compound RI: A sample (2)

- Goodness: the number of documents that may be found in a path
- Estimator:

\[ \text{NumberOfDocuments} \times \prod_{i} \frac{\text{CRI}(s_i)}{\text{NumberOfDocuments}} \]

- Examples
  - Query related to database & language
  - Node B: \(100 \times 20 \times 30 / (100 \times 100) = 6\)
  - Node C: \(100 \times 0 \times 50 / (1000 \times 1000) = 0\)
  - Node D: \(200 \times 100 \times 150 / (200 \times 200) = 75\)
Compound RI: how to use it?

A query at A: documents related to both DB and L
Compound RI: how to create it?

When A and D setup connection:
1) Aggregate local RI; 2) Exchange aggregated RIs
Compound RI: how to create it?

3) A and D notify their neighbors (B, C, I, J) of RIs updating
4) B, C, I, J update their our RIs
**Compound RI: Algorithms**

RI Creation/Update Algorithm
Input: I
Output: An updated RI (as a side effect)

// Creation phase
RI[0] = Summary(I)
For each neighbor j:
    Send(j, RI[0])

// Update phase
OldRI = RI
While (true):
    Wait for a batch of aggregated RIs to arrive and/or for the local index to change
    For each aggregated RI (if any), A_k, received from neighbor k:
        RI[k] = A_k
    If local index changed:
        RI[0] = Summary(I)
    If OldRI is different enough than RI:
        OldRI = RI
        For each neighbor j in RI:
            A = Aggregate(RI[(0,...,j-1, j+1..)])
            Send(j, A)
Compound RI: Algorithms

Answer Query
Input: A query $Q$, a client $C$, and a node $F$
Output: Query answers, Query forwards

Send local answers for $Q$ to $C$
If $StopCondition()$:
    return
$Rank = [ ]$
For each neighbor $j$ except $F$:
    $Rank.Append([j, Estimator(RI[j])])$
Sort tuples in $Rank$ by second attribute
for $t$ in $Rank$:
    Send query to neighbor $t[0]$
If $StopCondition()$:
    return
Hop Count Indices (1)

- **Compound RI’s limitation**
  - Considering no the number of hops query forwarding required to find documents

- **Hop count RI**
  - storing aggregated RIs for each hop up to a maximum number of hops (i.e., horizon of the RI)
  - A cost model based on hop count is required
### Hop Count Indices (2)

**Hop count indices of node W**

<table>
<thead>
<tr>
<th>Neighbor</th>
<th>1 Hop</th>
<th></th>
<th>2 Hops</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#</td>
<td>DB</td>
<td>N</td>
<td>T</td>
</tr>
<tr>
<td>X</td>
<td>60</td>
<td>13</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>10</td>
<td>10</td>
<td>4</td>
</tr>
<tr>
<td>Y</td>
<td>30</td>
<td>0</td>
<td>3</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>31</td>
<td>0</td>
<td>15</td>
</tr>
<tr>
<td>Z</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>70</td>
<td>10</td>
<td>40</td>
<td>20</td>
</tr>
</tbody>
</table>

![Graph Diagram]
Hop Count Indices (3)

- Goodness = \( \frac{N \text{-documents}}{N \text{-messages}} \)
- For regular-tree network
  - Documents distributed uniformly across the network
  - Each node has fanout \( F \)
  - Cost model:

\[
goodness_{hc}(\text{Neighbor}_i, Q) = \sum_{j=\overline{1},..,k} \frac{\text{goodness}(N_{i[j]}, Q)}{F^j-1}
\]
Exponentially Aggregated RI

- Drawbacks of hop count RI
  - Additional storage and transmission cost
  - Have no information beyond the horizon
- Exponentially aggregated RI
  - Can overcome these shortcomings
  - But a cost of potential loss in accuracy
Exponentially Aggregated RI

- Storing the result of applying the regular tree cost formula to a hop count RI goodness
- Each entry of the ERI for node N contains a value computed as

\[ \sum_{j=1}^{TH} \frac{\text{goodness}(N[j], T)}{I_j} \]
Exponentially Aggregated RI

An example

<table>
<thead>
<tr>
<th>Path</th>
<th>#</th>
<th>DB</th>
<th>N</th>
<th>T</th>
<th>L</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>66.67</td>
<td>16.33</td>
<td>5.33</td>
<td>6.33</td>
<td>15.67</td>
</tr>
<tr>
<td>Y</td>
<td>46.67</td>
<td>10.33</td>
<td>3.00</td>
<td>20.00</td>
<td>18.67</td>
</tr>
<tr>
<td>Z</td>
<td>28.33</td>
<td>5.33</td>
<td>13.33</td>
<td>9.67</td>
<td>19.67</td>
</tr>
</tbody>
</table>
Differences between ERI and HCRI

- Hop count RI does not have any information beyond the horizon
- Exponential RI can keep information for all nodes accessible from each neighbor in the RI
- Experiments shows that the exponential RI outperforms the hop count RI in most cases
Semantic Overlay Networks for P2P Systems

Arturo Crespo and Hector Garcia Molina

Dept. of Computer Sci.
Stanford University
Semantic Overlay Network
Thank You!