Search and Replication in Unstructured Peer-to-Peer Networks

Kaloyan Kanev

Jacobs University

14-04-2008
Decentralized Unstructured P2P

- Decentralized Unstructured P2P - no central server and unstructured topology

- Advantages:
  - Robustness against attacks
  - No single point of failure
  - Simple to implement
  - Partial queries

- Disadvantages:
  - Inefficient searching (undirected and large overhead)
  - Unpopular items are slow or impossible to find

- This paper discusses better search algorithms and replication techniques to improve search efficiency and availability
Introduction
Decentralized Unstructured P2P

Notation and Metrics
Notation
Metrics

Search Methods
Flood and Expanding Ring
Random Walks

Replication Strategies
Replication Distributions
Achieving Square Distribution
Replication Strategies

Experiments and Conclusions
Network Topology
Search Strategies and Query/Replication distributions
Replication Distributions
Replication Strategies
There are $m$ different items each has normalized popularity (i.e. fraction of all queries) $q_i$: $\sum_{i=1}^{m} q_i = 1$

Two types of popularity distribution:
- Uniform (equal popularity): $q_i = \frac{1}{m}$
- Zipf-like: $q_i \propto \frac{1}{i^\alpha}$

Each item $i$ is replicated on $r_i$ nodes for total $R$ objects on the network: $R = \sum_{i=1}^{m} r_i$

Three types of replication distribution:
- Uniform: $r_i = \frac{R}{m}$
- Proportional: $r_i \propto q_i$
- Square-root: $r_i \propto \sqrt{q_i}$
Introduction
Notation and Metrics
Search Methods
Replication Strategies
Experiments and Conclusions

Metrics

- User aspects:
  - $Pr(success)$ - probability of success
  - $\#hops$ - delay before item is found

- Load aspects:
  - $\#msgs$ per node - number of search messages a peer has to process
  - $\#nodes visited$ - how many nodes are visited per query
  - percentage of message duplication ($\frac{total\_\#msgs - \#nodes visited}{total\_\#msgs}$)
  - $peak \#msgs$ - marks hot spots in the network
Flood and Expanding Ring

- **TTL Flood**
  - Flood to all neighbors with predetermined TTL
  - Main problem is large message duplication percent, which increases network load

- **Expanding Ring**
  - Iterative deepening approach
  - Like TTL Flooding, but gives less overhead for queries for popular items
Random Walks

- Instead of flooding to all neighbors, randomly choose one neighbor to forward query to
- To improve speed requesting node initiates several "walkers"
- There is still TTL but it is much larger
- The implementation uses "checking", which means that a walker periodically checks with the requester and terminates if the requester already received query hit
- Another variant of the algorithm keeps state, to prevent using the same path if a host receives duplicated query
Uniform and Proportional - Average search size for m distinct item and n peers: $A_{\text{prop/uniform}} = \frac{m}{\rho}$ where $\rho = \frac{R}{n}$

Square-Root Replication: $A_{\text{optimal}} = \frac{1}{\rho} \left( \sum_i \sqrt{q_i} \right)^2$

Square-Root Replication is achieved for $r_i = \lambda \sqrt{q_i}$ where $\lambda = \frac{R}{\sum_i \sqrt{q_i}}$
Each element is replicated $c \frac{n}{r_i}$ each time query is issued.

$r_i$ can be roughly described by the DE: $\dot{r}_i = q_i c \frac{n}{r_i}$

Logarithmic ratio between two items is $\dot{z} = c \left( \frac{q_i}{r_j^2} - \frac{q_i}{r_i^2} \right)$

There is a fixed point in $r_i = \lambda \sqrt{q_i}$, so perhaps duplicating proportionally to the sites probed would yield square-root distribution.

A FIFO is used for removing old cache entries to maintain steady state (deletion ration equals insertion ratio).
Replication Strategies

- Owner replication - store only on requesting node
- Path replication - store on all nodes on the path between requester and provider
- Random replication - for path of length $p$ between provider and requester, store on $p$ nodes from the set of all nodes visited by all $k$ walkers
Network Topology

<table>
<thead>
<tr>
<th></th>
<th>#nodes</th>
<th>total #edges</th>
<th>avg. node degree</th>
<th>std. dev.</th>
<th>max degree</th>
<th>median degree</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLRG</td>
<td>9230</td>
<td>20599</td>
<td>4.46</td>
<td>27.9</td>
<td>1746</td>
<td>1</td>
</tr>
<tr>
<td>Random</td>
<td>9836</td>
<td>20099</td>
<td>4.09</td>
<td>1.95</td>
<td>13</td>
<td>4</td>
</tr>
<tr>
<td>Gnutella</td>
<td>4736</td>
<td>13022</td>
<td>5.50</td>
<td>10.7</td>
<td>136</td>
<td>2</td>
</tr>
<tr>
<td>Grid</td>
<td>10000</td>
<td>19800</td>
<td>3.96</td>
<td>0.20</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

**Figure**: Key statistics of the network topologies
Network Topologies with TTL

Figure: TTL Flooding success rate and average load
Network Topologies with TTL

Figure: TTL Flooding message duplication and node coverage
### Search Strategies and Query/Replication distributions

<table>
<thead>
<tr>
<th>distribution model</th>
<th>metrics</th>
<th>50% (queries for hot objects)</th>
<th>100% (all queries)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>flood</td>
<td>ring</td>
</tr>
<tr>
<td>query/replication</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Uniform / Uniform</td>
<td>#hops</td>
<td>2.39</td>
<td>3.40</td>
</tr>
<tr>
<td></td>
<td>#msgs per node</td>
<td>4.162</td>
<td>0.369</td>
</tr>
<tr>
<td></td>
<td>#nodes visited</td>
<td>4556</td>
<td>933</td>
</tr>
<tr>
<td></td>
<td>peak msgs</td>
<td>64.9</td>
<td>6.4</td>
</tr>
<tr>
<td>Zipf-like / Proportional</td>
<td>#hops</td>
<td>1.60</td>
<td>2.18</td>
</tr>
<tr>
<td></td>
<td>#msgs per node</td>
<td>2.961</td>
<td>0.109</td>
</tr>
<tr>
<td></td>
<td>#nodes visited</td>
<td>3725</td>
<td>357</td>
</tr>
<tr>
<td></td>
<td>peak msgs</td>
<td>43.8</td>
<td>2.0</td>
</tr>
<tr>
<td>Zipf-like / Square root</td>
<td>#hops</td>
<td>1.88</td>
<td>2.70</td>
</tr>
<tr>
<td></td>
<td>#msgs per node</td>
<td>3.874</td>
<td>0.208</td>
</tr>
<tr>
<td></td>
<td>#nodes visited</td>
<td>4404</td>
<td>621</td>
</tr>
<tr>
<td></td>
<td>peak msgs</td>
<td>62.5</td>
<td>3.8</td>
</tr>
</tbody>
</table>

**Figure:** Static simulation results for gnutella graph
Replication Distributions

**Figure:** Utilization, Average Search Size, and Relative Allocation
Replication Distributions

**Figure:** Distribution of replication ratios
Replication Strategies Compared

Figure: CDF of % success per number of hops
Replication Strategies Compared

<table>
<thead>
<tr>
<th></th>
<th>Owner Replication</th>
<th>Path Replication</th>
<th>Random Replication</th>
</tr>
</thead>
<tbody>
<tr>
<td>avg #msgs per node</td>
<td>56542.6</td>
<td>19155.5</td>
<td>14463.0</td>
</tr>
<tr>
<td>factor of improvement</td>
<td>1</td>
<td>2.95</td>
<td>3.91</td>
</tr>
</tbody>
</table>

Figure: Average number of messages for different replication strategies
Conclusions

- Random graphs perform better than Power Law Random Graphs
- Random walk search strategy can decrease the average load with 2 orders of magnitude
- Using random replication strategy, as described in the paper, achieve close to optimal distribution of replication and improves overall network performance