Making Gnutella-like P2P Systems Scalable

Dimitris Mavromatis

Computer Science Department
University of Crete

May 10, 2016
Introduction to P2P

- **Napster**
  - Introduced P2P file-sharing.
  - Distributing Download to peers (Serverless).
  - Centralized search.

- **Gnutella-like**
  - Decentralized Downloads and Searches to peers.
  - Unstructured overlay network.
  - Not scalable.
Introducing Gia

- Gia, a Gnutella-like protocol, improves Gnutella systems for decentralized search algorithms.

**Design Key Components**

- Dynamic Topology Adaptation protocol; Nodes are close to high-capacity nodes.
- Active Flow Control; avoid overloaded hot-spots with the use of flow-control tokens.
- One-hop Replication; nodes maintain pointers to the content of their immediate neighbors; allows high-capacity nodes to answer more queries.
- Search Protocol; uses biased random walks towards high-capacity nodes, instead of the Gnutella’s flooding mechanisms.

- It is believed to achieve a system capacity of three to five orders of magnitude better than Gnutella.
Gia is modeled after the Gnutella protocol.

All clients connect to each other using a three-way handshake.

All messages exchanged by clients are tagged, to their origins with a globally unique identifier (GUID).

GUID is a random sequence of 16 bytes, used to track messages through the network.

Takes into account client capacity and network heterogeneity.
Each Gia client maintains a host cache consisting of a list of other Gia nodes. Cache is updated throughout the lifetime of client.

Each node has a level of satisfaction, $S$:
- $S=0$ if no neighbors (dissatisfied)
- $S=1$ if enough good neighbors (fully satisfied)
- $S$ is a function of capacity, degree, age of neighbors.
- Improve the neighbor set as long as $S < 1$.

High-capacity nodes must be well-connected nodes with high satisfaction factor.
Node $X$ Joins

- New node randomly selects candidate entries from its host cache.
- After, $X$ select the node with maximum capacity greater than its own capacity. If no such node exists it picks a random one.
- Then, $X$ initiates a three-way handshake to the selected neighbor, $Y$.
- During the handshake each node decides whether to accept each other, based on the capacity and degree of the existing neighbors.
If $X$’s total number of neighbors does not exceed the configured bound $max_{n}b_{rs}$, it automatically accepts $Y$.

Otherwise, the node must identify the appropriate existing neighbor to drop and replace.
Flow Control

- Allocate tokens to neighbors based on processing capabilities.
- Each token represents a query, that the client is willing to accept.
- Thus, a high flow of tokens from a node to its neighbors indicates that the client is able to receive a high amount of queries.
- If the traffic load on a client is too high, it allocates tokens at a lower rate.
- Allocation is not proportional to neighbors capacities.
- Neighbors that do not use their assigned tokens are assumed inactive, leftover capacity is distributed to remaining neighbors.
One-hop Replication

- Nodes maintain an index of the content of each of its neighbors.
- Exchanged during connection establishment and are updated periodically.
- Receiving a Query the node searches not only in its content, but also provide matches from the content by all of its neighbors.
-Flushed on node failures; ensuring that information remains mostly up-to-date.
Search Protocol (1/2)

- Uses a biased random walk.
  - Pick highest capacity node to which it has tokens.
  - If no tokens, queues till arrive.
- A unique ID, GUID, is given to each query for nodes keep a list of forwarded queries to avoid path looping.
- In addition each query also has a MAXRESPONSES parameter, which is decremented every time the query finds a match.
Once *Max Responses* or *TTL* reaches zero, the query is discarded.

The query responses are forwarded back to the originator of the query, along with the path associated with the query. If reverse path is lost, we rely on recovery mechanisms.

Book-keeping: Maintain list of neighbors to which a query (unique GUID) has been forwarded.
Simulation Results

- Compare four Systems:
  - FLOOD: Search using $TTL$-scoped flooding over random topology (Gnutella model).
  - RWRT: Search using random walks over random topologies (Suggested by Lv et al.).
  - SUPER: Search using super-node mechanisms. Queries flooded only between super-nodes.
  - GIA: Search using Gia protocol, including topology adaptation, active flow control, one-hop replication, and biased random walks.

- Metric:
  - Success-rate, Delay, Hop-count
  - Collapse point
System Model

- Five levels of capacity assumed (Table 1). Last two levels are designated as super-nodes for SUPER experiments.
- All nodes generate queries at the same rate bounded by their capacities.
- Keyword queries are performed.
- Control traffic consumes resources.
- The replication factor refers to the fraction of nodes at which answers to queries reside.

<table>
<thead>
<tr>
<th>Capacity level</th>
<th>Percentage of nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1×</td>
<td>20%</td>
</tr>
<tr>
<td>10×</td>
<td>45%</td>
</tr>
<tr>
<td>100×</td>
<td>30%</td>
</tr>
<tr>
<td>1000×</td>
<td>4.9%</td>
</tr>
<tr>
<td>10000×</td>
<td>0.1%</td>
</tr>
</tbody>
</table>

Table 1: Gnutella-like node capacity distributions
Simulations Results (1/2)

Figure 1: Success rate, hop-count and delay for increasing load
Figure 2: Hop-count before collapse
Experimental Results

Figure 3: Ran GIA on 83 nodes of PlanetLab for 15 min, Artificially imposed capacities on nodes, Progress of topology adaptation shown
GIA: scalable Gnutella
3–5 orders of magnitude improvement in system capacity
Unstructured approach is good enough!
DHTs may be overkill
Incremental changes to deployed systems
Thank you for your Attention! Questions?