Presentation Report  
Making Gnutella-like P2P Systems Scalable 

Dimitrios Mavrommatis 
May 30, 2016 

Design Components 

Gia, is a Gnutella-like protocol, which improves Gnutella systems for decentralized search algorithms. The key design components consist of: 

• **Dynamic Topology Adaptation protocol** 

The topology adaptation algorithm is the core component that connects the Gia client to the rest of the network. Each Gia client maintains a host cache consisting of a list of other Gia nodes (their IP address, port number, and capacity). The goal of the topology adaptation algorithm is to ensure that high capacity nodes are indeed the ones with high degree and that low capacity nodes are within short reach of higher capacity ones. To achieve this goal, each node independently computes a level of satisfaction (A value of S = 0 means that the node is quite dissatisfied, while S = 1 suggests that the node is fully satisfied.). 

To add a new neighbor, a node (say X) randomly selects a small number of candidate entries from those in its host cache that are not marked dead and are not already neighbors. From these randomly chosen entries, X selects the node with maximum capacity greater than its own capacity. If no such candidate entry exists, it selects one at random. Node X then initiates a three-way handshake to the selected neighbor, say Y. If, upon accepting the new connection, the total number of neighbors would still be within a preconfigured bound max nbrs, then the connection is automatically accepted. Otherwise, the node must see if it can find an appropriate existing neighbor to drop and replace with the new connection. 

• **Flow Control**
To avoid creating hot-spots or overloading any one node, Gia uses an active flow control scheme in which a sender is allowed to direct queries to a neighbor only if that neighbor has notified the sender that it is willing to accept queries from the sender.

Each client allocate tokens for neighbors based on processing capabilities. A 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 allocate tokens at a lower rate. Allocation is proportional to neighbors capacities and when a neighbor does not use its assigned tokens; he is assumed inactive and the tokens are reallocated.

• **One-Hop Replication**

To improve the efficiency of the search process, each node maintains an index of the content of each of its neighbors. Indexes are exchanged during connection establishment and are updated periodically.

when a node receives a query, it can respond not only with matches from its own content, but also provide matches from the content offered by all of its neighbors. When a neighbor is lost, either because it leaves the system, or due to topology adaptation, the index information for that neighbor gets flushed.

• **Search Protocol**

Gia search protocol uses a biased random walk: rather than forwarding incoming queries to randomly chosen neighbors, a Gia node selects the highest capacity neighbor for which it has flow-control tokens and sends the query to that neighbor.

A node remembers the neighbors to which it has already forwarded queries for a given GUID. If a query with the same GUID arrives back at the node, it is forwarded to a different neighbor.

Each query has a MAX RESPONSES parameter, the maximum number of matching answers that the query should search for. In addition to the TTL. Query responses are forwarded back to the originator along the reverse-path associated with the query.

**Simulation**

Four systems selected for the comparison:

mavromat@csd.uoc.gr  Dimitrios Mavrommatis
www.csd.uoc.gr/~mavromat  ID: 924 Page 2
• FLOOD: Search using TTL-scoped flooding over random topology (gnutella model)

• RWRT: Search using random walks over random topologies

• SUPER: Search using super-node mechanisms. Queries flooded only between super-nodes.

• GIA: Search using Gia protocol with the design mechanisms mentioned before.

Five levels of capacity assumed where the last two levels are designated as super-nodes. All the nodes generate queries at the same rate bounded by their capacities. The results that were observed can be seen at fig.1, fig.2 and fig.3 (the replication factor that is mentioned inside the figures refers to the fraction of nodes which contain the answer to the queries.)

Conclusion

GIA achieved the scalability that original Gnutella lacked and it achieved 3–5 orders of magnitude improvement in system capacity. The results have shown that unstructured approach is good enough and DHTs may be overkill.
Figure 1: Success rate, hop-count and delay for increasing load

Figure 2: Hop-count before collapse

Figure 3: Ran GIA on 83 nodes of PlanetLab for 15 min, Artificially imposed capacities on nodes, Progress of topology adaptation shown

mavromat@csd.uoc.gr
www.csd.uoc.gr/~mavromat

Dimitrios Mavrommatis
ID: 924 Page 4