Efficient Search in Unstructured Peer-to-Peer Networks

Vicent Cholvi
Universitat Jaume I
Castellón (Spain)
Vicent.Cholvi@uji.es

Pascal Felber
Institut Eurécom
Sophia–Antipolis (France)
Pascal.Felber@eurecom.fr

Ernst Biersack
Institut Eurécom
Sophia–Antipolis (France)
Ernst.Biersack@eurecom.fr

Categories and Subject Descriptors

C.2.4 [**Distributed Systems**]: Distributed applications; C.4 [**Performance of Systems**]: Performance attributes

General Terms

Algorithms, Performance, Design

Keywords

Peer–to-peer systems

1. INTRODUCTION

The last few years have witnessed the appearance of a growing number of peer-to-peer (P2P) file sharing systems. Such systems make it possible to harness the resources of large populations of networked computers in a cost-effective manner, and are characterized by their high scalability.

The main objective of this work is to develop techniques to render the search process in unstructured P2P file sharing systems more efficient and scalable, by taking advantage of the common interests of the peer nodes and effectively implement a "directed flooding" search protocol.

In this paper we propose *Acquaintances*, an extension to Gnutella-like unstructured P2P networks that uses dynamic topology adaptation to improve search efficiency. As in Gnutella, our search mechanism uses TTL-limited flooding to broadcast queries in a neighborhood. By associating a TTL (time to live) value to the query messages, one can restrict the search diameter, i.e., the size of the flooded neighborhood, and limit the adverse effects of exponential message generation on the network links. However, the probability of finding a file that does exist in the network strongly depends on the chosen TTL value: bigger values increase the success rate but may quickly lead to network congestion.

To minimize this problem, we propose novel techniques to build self-organized communities of peer nodes that share similar interests. These communities are maintained by dynamically adapting the topology of the P2P overlay network based on the query patterns and the results of preceding searches. Some of the neighbor links (which we call *acquaintance* links) are explicitly reserved for building semantic communities and are continuously updated according to some link replacement algorithm; these links allow peers to quickly locate files that match their interests. The other links are mostly static and random; they help maintain global connectivity and locate more marginal content.

SPAA'04, June 27–30, 2004, Barcelona, Spain.
ACM 1-58113-840-7/04/0006.

2. *Acquaintances* EVALUATION

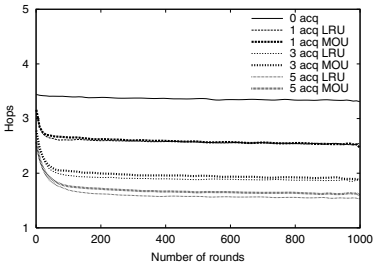
To evaluate the effectiveness of our techniques, we have built a network simulator and conducted extensive simulations under realistic operating conditions. A detailed description of the experimental setup and methodology used for the evaluation of *Acquaintances* can be found in [1].

Acquaintance Links. The first metric used in our evaluation is the *number of hops* necessary to reach the first peer that serves the requested file. The number of hops is a measure of the response time and allows us to choose adequate TTL values to experience a good query success rate without overloading the network.

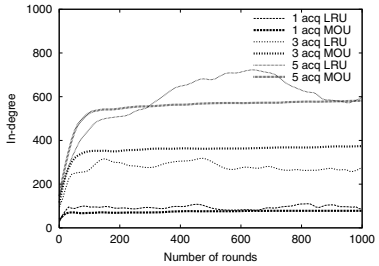
Fig.(a) shows that the system, initially with random connectivity, needs only a few rounds to set up acquaintance links and stabilize in an efficient configuration. Regardless of the acquaintance replacement policy (we considered *Least Recently Used (LRU)* and *Most Often Used (MOU)* policies), the number of hops is reduced ranging from 30% up to 80% depending on the number of acquaintances. We have observed that more popular files are generally found closer to the requester; this behavior can be explained by the fact that popular files have more copies.

The *in-degree* of a peer p is defined as the number of other peers that have chosen p as acquaintance. We have computed the maximum over all peers at the end of each round. As the number of queries received by a peer is clearly proportional to its in-degree, it is important to keep this value within reasonable bounds. Fig.(b) shows that, by introducing acquaintances, we increase the maximum in-degree in the network. This is not surprising as the peers that serve many files are more likely to be chosen as acquaintances by the other peers. Conversely, free-riders should be acquainted with almost no other peer. The *LRU* acquaintance replacement policy exhibits more volatility than *MOU*.

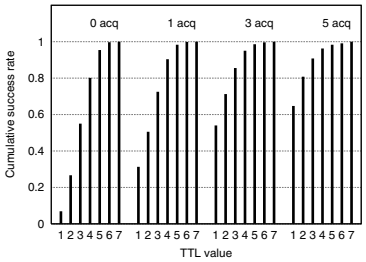
The metric used to study the effect of the TTL value is the *cumulative success rate*, i.e., the cumulative probability of success in the TTL-limited neighborhood of the requester. High success rates for small TTL values indicate that search



(a) Number of hops to the closest peer that serves the requested file.



(b) Maximum in-degree over all the peers in the network.



(c) Cumulative success rate, averaged over the last 800 rounds of the simulation, with a *MOU* acquaintance replacement policy.

is more efficient. Fig. (c) shows that more acquaintances leads to higher success rates for any TTL value.

Friends Awareness. We now consider the case where each peer maintains an index of the files stored on its friends (i.e., the other peers of which it is an acquaintance) and uses this knowledge to answer to queries on their behalf, when possible. Our experiments show that, even with a single acquaintance, the number of hops needed to reach the first peer that serves, or has a friend that serves, the requested file drops to the optimal value 1 after a few rounds. This occurs regardless of the acquaintance replacement policy (*LRU* or *MOU*). As almost all requests are satisfied after a single hop, peers have no incentive to change their acquaintances; we have indeed observed that the number of promotions with both the *LRU* and *MOU* acquaintance replacement policies is almost null.

The experiments also show that the in-degree of the busiest peer grows quickly to almost attain the total number of peers in the system. This indicates that one peer acts as a central hub that all other peers choose as acquaintance; it is chosen initially because it serves many files, and later because it has many friends and consequently can answer to almost all queries. This snowball effect leads the system to spontaneously self-configure as a system with a central "index" peer, like Napster. The major difference with a centralized system is that, if the central index fails or leaves, the system quickly reconfigures and chooses another index peer.

A configuration with a central index has the major drawback of overloading the index peer (experiments show a significant degradation of the load distribution). In addition, the index peer must have enough resources to maintain the state of all its friends, i.e., the list of almost all the files served by all the peers in the P2P network. To overcome these drawbacks, we adopt a less extreme approach and we bound the maximum number of friends that a peer needs to keep track of. We have run simulations with this limit set to 25. Our experiments show gains on the number of hops needed to reach the first peer that answers a query (with respect to the case where peers keep no state) ranging from 8% to 16% with the *LRU* acquaintance replacement policy, and from 25% to 35% with *MOU*. The lower performance of *LRU* can be explained by the higher volatility of the network connectivity, which leads peers to only perform short-term optimizations. Our results also shows that the increase of the number of requests answered by the busiest peer is moderate even with the *MOU* acquaintance replacement policy, which confirms that friends awareness does not cause hot-spots.

Dynamic TTL. We finally evaluate the effectiveness of using a dynamic TTL mechanism which decrements the TTL value twice when a query falls within the interests of the peer being traversed during query flooding. As queries require fewer hops to be satisfied within the scope of a semantic community, we expect this optimization to have little impact on the success rate while significantly decreasing the query traffic.

We have measured the *total number of query messages* sent across the network using a TTL of 6 as base value for the experiments, and observed a important reduction of the query traffic when using the dynamic TTL optimization, by factors of 4 to more than 8.

We have also measured the *query failure rate*, i.e., the proportion of requests that yield a negative result although the requested file is available in the system. Unsurprisingly, the percentage of failures with the dynamic TTL optimization is high (10%) with no acquaintances, because it is designed to take advantage of the semantic communities that are created by the acquaintance links. When using acquaintances, the failure rate decreases (4% to 6%) but remains significantly higher than with a static TTL value. In contrast, when increasing the TTL value to 7, the dynamic TTL optimization exhibits much lower failure rates (less than 0.1%). This optimization can thus at the same time improve the success rate and reduce the query traffic, when using an adequate TTL value.

3. CONCLUSIONS

Results demonstrate that, when extending a basic Gnutella-like network with *Acquaintances*, one can significantly improve the search efficiency and reduce the network load. A more detailed description of our results can be found at [1].

[1] V. Cholvi, P. A. Felber, and E. W. Biersack. Efficient search in unstructured peer-to-peer networks. Technical Report RR-03-090, Institut Eurècom, 2003.