A Super-Peer Based Lookup in Structured Peer-to-Peer Systems

Yingwu Zhu
ECECS Department
University of Cincinnati
Cincinnati, OH 45221

Honghao Wang
ECECS Department
University of Cincinnati
Cincinnati, OH 45221

Yiming Hu
ECECS Department
University of Cincinnati
Cincinnati, OH 45221

Abstract

All existing lookup algorithms in structured peer-to-peer (P2P) systems assume that all peers are uniform in resources (e.g., network bandwidth, storage and CPU). Messages are routed on the overlay network without considering the differences of capabilities among participating peers. However, the heterogeneity observed in deployed P2P systems is quite extreme (e.g., with up to 3 orders of magnitude difference in bandwidth). The bottleneck caused by very limited capabilities of some peers therefore could lead to inefficiency of existing lookup algorithms. In this paper we propose a super-peer based lookup algorithm and evaluate it using a detailed simulation. We show that our technique not only greatly improves the performance of processing queries but also significantly reduces aggregate bandwidth consumption and processing cost of each query.

1 Introduction

Structured P2P systems, such as Pastry [1], Chord [2], Tapestry [3] and CAN [4], offers applications with a distributed hash table (DHT) abstraction. In such structured systems, each object that is stored into the DHT has a unique key. Thus, these systems provide a DHT interface of *put(key,object)*, which stores the object with the key, and *get(key)* which retrieves the object corresponding to the key.

One of the key problems in structured P2P systems is to provide an efficient location and routing (lookup) algorithm. All of these existing lookup algorithms in structured P2P systems, make an explicit or implicit assumption that all peers are uniform in resources (e.g., network bandwidth, storage and CPU). Messages are routed on the overlay network without considering the differences of capabilities among participating peers. However, Measurement studies such as [5] have shown that the heterogeneity in deployed P2P systems is quite extreme (e.g., with up to 3 orders of magnitude difference in bandwidth). The bottleneck caused by very limited capabilities of some peers could lead to the inefficiency of these existing lookup algorithms. Therefore, it is possible to improve the performance of these existing lookup algorithms by taking into account the heterogeneity nature of P2P systems.

In this paper, we present a super-peer based lookup algorithm designed to enhance the performance of existing lookup algorithms in structured P2P systems. A super-peer based lookup algorithm is one which exploits the heterogeneity (in network bandwidth, storage capacity, and processing power) among participating peers, by assigning greater responsibility to those super-peers who have high network bandwidth, large storage capacity and significant processing power. The super-peer based lookup algorithm can coexist with existing lookup algorithms through a hybrid approach: first try the super-peer based lookup algorithm, then follow with the existing lookup algorithm if needed. We then use a detailed simulation to explore the behavior of this super-peer based lookup algorithm on a uniform distribution of files. Simulations show that our super-peer based algorithm finds files faster and consumes less bandwidth and processing power than the existing lookup algorithm.

The remainder of this paper is organized as follows. Section 2 presents our super-peer based lookup algorithm. Section 3 describes our simulation environment, and Section 4 describes our experimental results. Section 5 gives an overview of related work. We finally conclude in Section 6.

2 Super-Peer Based Lookup

The basic idea behind the super-peer based lookup is to take advantage of the heterogeneity of capabilities (in network bandwidth, storage capacity, and processing power) among participating peers, by assigning greater responsibility to those super-peers. A super-

peer acts as a centralized server to a subset of clients, and clients submit search queries to their super-peer and receive results from it. The goal of the super-peer based lookup is to reduce the resulting pathlength of a query message and the aggregate load generated by the query across the network. Our super-peer based lookup does not intend to replace the current lookup algorithm used in the system. Instead, it acts as a complement to the current lookup algorithm. If the super-peer based lookup fails to route a query message, it has to resort to the existing lookup algorithm. Therefore, the super-peer based lookup and the existing lookup can coexist in the system.

2.1 Existing Lookup Algorithms

In this section, we review the existing lookup algorithms used in structured P2P systems. The lookup algorithms, though different in design considerations, have more commonality than differences. All of them, given a DHT key, route a message to a destination node who is responsible for that DHT key. Each node is associated with an identifier. The keys and nodes' identifiers are pseudorandom, fixed-length bit strings. Each node maintains a routing table consisting of a small subset of nodes in the system (e.g., $O(\log N)$). When a node receives a query message with a key for which it is not responsible, the node routes the message to a neighbor node who is closer to the key than its own identifier. In worst case, a lookup operation requires $O(\log N)$ sequential network messages to search a P2P system of N peers.

Due to space constraints, we here do not further describe the lookup algorithms. Please see [1, 2, 3, 4] for more detail.

2.2 Super-Peer Overlay Network

A super-peer overlay network is a secondary overlay constructed on super-peers. It is similar to the overlay constructed in structured P2P systems (we refer to it as the original overlay) in [1, 2, 3, 4], except that it is completely composed of super-peers. A super-peer has multiple roles: (1) It is a peer on the original overlay. (2) It is a peer on the secondary overlay. (3) It is a centralized sever to a subset of clients (weak peers on the original overlay). Clients submit queries to their super-peer and receive results from it. We refer to a super-peer and its clients as a *cluster*. Figure 1 illustrates what the topology of a super-peer overlay network might look like.

As a result, each super-peer has two node IDs. One is for the original overlay and the other is for the super-peer overlay. Moreover, each super-peer maintains a routing table for *each* of these two overlays.

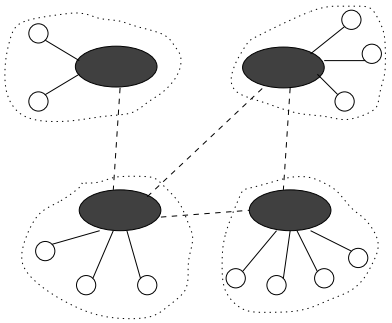

Figure 1: Illustration of a Super-Peer Overlay Network. Black nodes represent super-peers, while white nodes represent clients. A cluster is composed of a super-peer and its clients.

A super-peer maintains an index over its own clients' data. This index, for example, can be composed of tuples $< fileId, C >$ (where the client C owns the file specified by the $fileId$). Then, each tuple is published by the super-peer S over the super-peer overlay in the form of a triple $< fileId, S, C >$. Figure 2 depicts the index maintained in the super-peer node and on the super-peer overlay. The overhead of maintaining such an index at the super-peer is expected to be small in comparison to the savings in query costs the centralized index introduces. For instance, given a query, if the query source node and query destination node of a query are located in the same cluster, the current lookup algorithm in P2P systems however cannot realize it, and the query message might be routed across multiple autonomous systems (AS) before reaching the destination. Some of these overlay hops might even involve transcontinental links since the peers in P2P systems could be geographically distributed, resulting in high latency. In contrast, the centralized index allows the super-peer to directly forward the query to the destination within the cluster and avoid routing the query across multiple ASs, thus minimizing both latency and network hops as well as reducing network traffic.

When a client C joins the system, it will first associate itself to a nearby super-peer S. It will then send the tuples over its data collection to its super-peer S, and the super-peer will add the tuples into its index and publish all the triples corresponding to these tuples over the super-peer overlay. Specifically, If a client inserts a file (identified by the $fileId$), it will send the tuple $< fileId, C >$ to its super-peer,

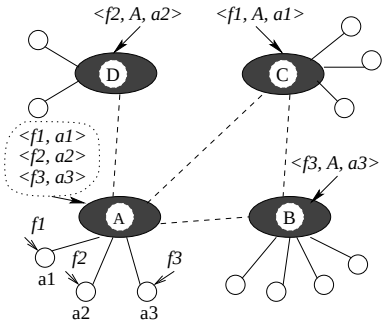

Figure 2: Illustration of a Super-Peer Overlay Network with Files Published. A, B, C, and D represent super-peers, while a1, a2 and a3 represent A's clients. A maintains an index over the files f1, f2, and f3, which are stored at a1, a2, and a3 respectively. A then publishes these tuples $< f1, a1 >$, $< f2, a2 >$, and $< f3, a3 >$ into the super-peer overlay network, in the form of $< f1, A, a1 >$, $< f2, A, a2 >$, and $< f3, A, a3 >$ respectively.

which will add this tuple into its index and then publish the triple $< fileId, S, C >$ over the super-peer overlay. If a client deletes a file, it will send the tuple $< fileId, C >$ to its super-peer, which will remove this tuple from its index and then withdraw the triple $< fileId, S, C >$ from the super-peer overlay. When a client leaves, its super-peer will remove the tuples owned by this client from its index and withdraw all the corresponding triples from the super-peer overlay. Note that a super-peer itself also publishes/withdraws its data indexes to/from the super-peer overlay.

2.3 Super-peer Selection and Election

As described earlier, a super-peer takes greater responsibilities by impersonating both a centralized sever to a set of clients, and a pure peer in a super-peer overlay. Therefore a super-peer candidate must meet the following requirements:

- It has significant processing power in order to route a large amount of overlay traffic on a overlay network of super-peers and process the search queries for its clients.
- It has high bandwidth and fast access to the Internet (e.g. with minimal number of IP hops to the wide-area network). For instance, gateway routers or machines are attractive candidates. Queries across the wide-area can be quickly routed to the destination node by taking advantage of the highly connected network infrastructure among super-peers.
- It has high availability. If super-peers tend to be unavailable frequently, this will have significant impact on the effectiveness of the super-peer based lookup algorithm.
- It has large memory and storage capacity. Unlike clients, a super-peer needs to have extra memory and storage to keep the index of files within its cluster and triples published by other super-peers on the super-peer overlay. Moreover, if it has extra storage space to cache files, this will greatly improve the performance of the super-peer based lookup algorithm.

  In the case of the crash or leave of a super-peer, another super-peer can be elected. Due to space constraints, we here do not discuss the detail of super-peer election.

2.4 Super-Peer Based Lookup Algorithm

In this section we describe our super-peer based lookup algorithm.

When a client wishes to deliver a query (specified by a $fileId$) to the network, it first sends the query to its super-peer. The super-peer then submits the query to the super-peer overlay as if it were its own query. The query is therefore routed to a super-peer who is responsible for the $fileId$ of this query.

Upon receiving this query, the destination super-peer can do an efficient lookup via hashtable to find the triple $< fileId, S, C >$. If the super-peer S is available, the query is forwarded to this super-peer, which will contact the client C and return the query result. Otherwise, the query is forwarded directly to the client C.

During each lookup operation, the triple can be cached along the way on the super-peer overlay. This operation could reduce the overlay hops and network traffic of each query across the super-peer overlay, greatly improving the lookup performance.

In summary, the super-peer based lookup acts as a complement instead of a replacement to the existing lookup algorithms. If the lookup of a query couldn't continue on the super-peer overlay due to the failure of super-peers, the query will leave the super-peer overlay (the highway) and take the original overlay (the local way) at some point. The existing location and routing algorithm then takes care of the query and routes it to the destination node through the original overlay.

3 Simulation Experiments

3.1 Metrics

In order to evaluate the effectiveness of our super-peer based lookup algorithm, we must first define some metrics.

- *Relative Hop Penalty (RHP).* We refer to the ratio of the actual IP hops to route a query on the overlay network versus the ideal IP hops on the underlying IP network to the query destination node.
- *Relative Delay Penalty (RDP).* We refer to the ratio of the actual time to route a query on the overlay network versus the ideal network latency on the underlying IP network to the query destination node.
- *Aggregate Bandwidth Cost (ABC).* We refer to the aggregate bandwidth consumed (in bytes) by each query.
- *Aggregate Processing Cost (APC).* We refer to the aggregate processing power consumed (in CPU cycles) by each query.

3.2 Simulation Environment

We constructed our simulator over a physical network topology, produced by GT-ITM [6]. The transit-stub graphs used in our simulations are generated with 6 transit domains of 10 nodes each. Each transit node is connected to 7 stub domains and each stub domain owns 22 nodes on the average, thereby yielding a total of 9,300 nodes per graph. Given these parameters, we generated seven graphs, and conducted our experiments on these seven graphs to ensure that our results are independent of the particularities of any one graph. On top of this physical network, we built the Pastry overlay and chose 60 peers as the super-peers on which we constructed the secondary Pastry overlay.

3.3 Experiment Descriptions

In all these experiments, we published 9,300 unique $fileId$s to the Pastry overlay network uniformly, so that each peer is responsible for a unique $fileId$. For each cluster, all the $fileId$s within the cluster are indexed as a tuple $< fileId, C >$ at the super-peer, which then publishes all these tuples into the secondary overlay network in the form of triple $< fileId, S, C >$. We chose a sample of query source peers, and then arranged for each query source peer to initiate queries of these $fileId$s we have published.

In the RHP experiments, we measure the RHP of the super-peer based lookup algorithm and Pastry on these queries, assuming that both inter-domain links and intra-domain links count as one hop. To account for the fact that inter-domain links incur higher latency than intra-domain links, we further measure the weighted RHP of the super-peer based lookup algorithm and Pastry, where each inter-domain hop counts as 3 hop units.

In the RDP experiments, we measure the RDP of the super-peer based lookup and Pastry on these queries, by assigning link latencies of 20 ms for transit-transit links, 5 ms for stub-transit links and 2 ms for intra-stub domain links.

In the ABC experiments, we calculate the aggregate bandwidth consumed under the super-peer based lookup and Pastry on these queries. We first need to estimate the size of a query message. We based our calculation of the query message size on the Gnutella network protocol [7]. In particular, a query message in Gnutella consists of a Gnutella header, a query string, and a field of 2 bytes to describe the minimum bandwidth (in kb/second) that should be provided by any responding node of this message. A Gnutella header is 22 bytes, and a TCP/IP and Ethernet header is 58 bytes. The query string in Gnutella is a variable string, and we here instead assume that it only contains a $fileId$, which is a 160-bit string in Pastry. Total query message size is therefore 102 bytes. We then calculate the aggregate bandwidth cost of the super-peer lookup and Pastry, given this query message size.

For APC experiments, we first have to estimate the processing cost of processing a 102-byte query in one node. We employ the method used in [8] and yield 2280 CPU cycles for processing such a query on a 930 MHz processor (Pentium III, running Linux Kernel Version 2.2). Given this processing cost, we then calculate the aggregate processing cost of the super-peer based lookup and Pastry.

4 Results

In this section, we utilize our experiment results to justify the claims we made in the introduction: that the super-peer based lookup algorithm finds files faster and consumes less bandwidth and processing power than the existing lookup algorithm.

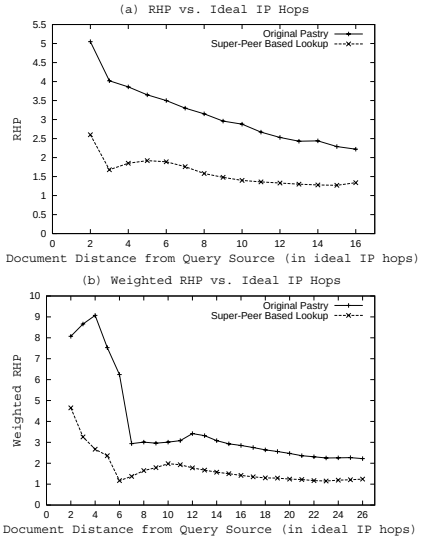

Figure 3: RHP vs. Ideal IP Hops.

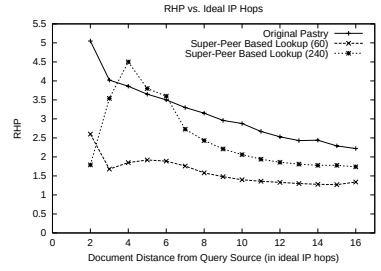

Figure 4: RHP vs. Ideal IP Hops. The RHP for the super-peer based lookup is measured under 60 and 240 super-peers respectively.

4.1 RHP

Figure 3 shows the RHP of our super-peer based algorithm and original Pastry as a function of the query source's distance from the queried document. Note that the super-peer based lookup has significant improvements over original Pastry, achieving a much lower RHP and reducing the routing overhead by about 34-81%.

Figure 4 shows the RHP for the super-peer based lookup algorithm with 60 and 240 super-peers on the secondary overlay network. Note that the super-peer based lookup with 60 super-peers has big improvements over the super-peer based lookup with 240 super-peers. Hence, the fewer the super-peers on the secondary overlay, the less routing overhead on the secondary overlay in locating super-peers for query destinations there will be and the more performance gains the super-peer based lookup can achieve.

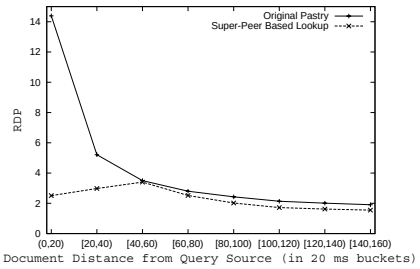

Figure 5: RDP vs. Ideal Latency.

4.2 RDP

Figure 5 plots the RDP of the super-peer based algorithm and original Pastry as a function of the query source's distance from the queried document. Note that the super-peer based lookup achieves improvements in RDP by about 10-43%.

4.3 ABC and APC

Due to space constraints, we here do not present the figures about ABC and APC. But the results show that the super-peer based lookup reduces both ABC and APC by 40-50%.

.

5 Related Work

Structured P2P systems ([1, 2, 3, 4]) assumes that all peers are uniform in resources. However, ignoring the heterogeneity in peer capabilities could lead to inefficiency of existing lookup algorithm. To account for the heterogeneity nature of P2P systems, Zhao et al.[9] present the initial architecture of a brocade secondary overlay on top of a Tapestry network to improve routing performance. CFS [10] also proposes to hosting a number of virtual servers on a physical sever, to account for varying resource capabilities among nodes.

The lookup algorithm of structured P2P systems performs less optimally as the queries document lies closer to the query source. To tackle this problem, [11] proposes a new, probabilistic location and routing by using an *attenuated Bloom Filter*, to improve the lookup performance. Recent work [12, 8, 13] strives to improve search efficiency in unstructured P2P networks. Moreover, Yang et al. [14] conduct some re-

search on a super-peer network, presenting us practical guidelines and a general procedure for the design of an efficient super-peer network.

6 Conclusions

In this paper we have presented and evaluated a super-peer based lookup algorithm in structured P2P systems. Compared to existing lookup algorithms in such systems, the super-peer based lookup algorithm not only greatly improves the performance of processing queries, but also significantly reduces aggregate bandwidth consumption and processing cost of each query. Furthermore, the super-peer based lookup algorithm can coexist with the existing lookup algorithm rather than replacing it. Due to the fact that the heterogeneity in P2P populations is quite extreme, we believe our super-peer based lookup algorithm can have a positive impact on current structured P2P systems.

References

[1] A. I. T. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *Proceedings of the 18th IFIP/ACM International Conference on Distributed System Platforms (Middleware 2001)*, (Heidelberg, Germany), Nov. 2001.

[2] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *Proceedings of ACM SIGCOMM*, (San Diego, CA), pp. 149–160, Aug. 2001.

[3] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph, "Tapestry: An infrastructure for fault-tolerance wide-area location and routing," Tech. Rep. UCB/CSD-01-1141, Computer Science Division, University of California, Berkeley, Apr. 2001.

[4] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and Shenker, "A scalable content-addressable network," in *Proceedings of ACM SIGCOMM*, (San Diego, CA), pp. 161–172, Aug. 2001.

[5] S. Saroiu, P. K. Gummadi, and S. D. Gribble, "A measurement study of peer-to-peer file sharing systems," in *Proceedings of Multimedia Computing and Networking*, (San Jose, CA), Jan. 2002.

[6] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *Proceedings of IEEE INFOCOM*, vol. 2, (San Francisco, CA), pp. 594–602, IEEE, Mar. 1996.

[7] "The gnutella protocol specification." http://dss.clip2.com/GnutellaProtocol04.pdf, 2000.

[8] B. Yang and H. Garcia-Molina, "Efficient search in peer-to-peer networks," in *Proceedings of the 22nd IEEE International Conference on Distributed Computing Systems*, (Vienna, Austria), July 2002.

[9] B. ZHAO, Y. DUAN, L. HUANG, A. JOSEPH, and J. KUBIATOWICZ, "Brocade: Landmark routing on overlay networks," in *Proceedings of 1st International Workshop on Peer-to-Peer Systems (IPTPS).*, Mar. 2002.

[10] F. DABEK, M. KAASHOEK, D. KARGER, R. MORRIS, and I. STOICA, "Wide-area cooperative storage with cfs," in *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP '01)*, (Banff, Canada), pp. 202–215, Oct. 2001.

[11] S. C. Rhea and J. Kubiatowicz, "Probabilistic location and routing," in *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, (New York, NY), June 2002.

[12] Q. Lv, P. Cao, and E. Cohen, "Search and replication in unstructured peer-to-peer networks," in *Proceedings of 16th ACM Annual International Conference on Supercomputing*, (New York, NY), June 2002.

[13] A. Crespo and H. Garcia-Molina, "Routing indices for peer-to-peer systems," in *Proceedings of the 22nd IEEE International Conference on Distributed Computing Systems*, (Vienna, Austria), July 2002.

[14] B. Yang and H. Garcia-Molina, "Designing a super-peer network," in *Proceedings of the 19th International Conference on Data Engineering (ICDE)*, (Bangalore, India), Mar. 2003.