Quickly routing searches without having to move content

Brian F. Cooper
Center for Experimental Research in Computer Systems
College of Computing, Georgia Institute of Technology
cooperb@cc.gatech.edu

Abstract

A great deal of work has been done to improve peer-to-peer routing by strategically moving or replicating content. However, there are many applications for which a peer-to-peer architecture might be appropriate, but in which content movement is not feasible. We argue that even in such applications, progress can be made in developing techniques that ensure efficient searches. We present several such techniques. First, we show that organizing the network into a square-root topology, where peer degrees are proportional to the square root of the popularity of their content, provides much better performance than power-law networks. Second, we present routing optimizations based on the amount of content stored at peers, and tracking the "best" peers, that can further improve performance. These and other techniques can make searches efficient, even when content movement or replication is not feasible.

1 Introduction

A large number of optimizations have been proposed to improve the performance and effectiveness of peer-to-peer searches. Many of these proposals involve moving or replicating content to achieve high performance. For example, Cohen and Shenker [4] propose replicating files in order to make them easier to find. Super-peer networks [21, 14] replicate content metadata from leaf peers to super-peers, where the actual search processing is done. Even distributed hash tables [19, 17, 18] move data, as content (or pointers to content) are taken from their original peer and moved to a location in the network based on a hash of the object identifier. Other examples of proposals to move or replicate content for efficiency include [20, 3, 8, 7, 11, 2].

In order for this *content movement* approach to be effective, it must be feasible to move objects around. For example, in the traditional application of multimedia filesharing, it makes sense to move or replicate content: the files and metadata rarely change and are small enough to replicate. However, in many cases content movement may not be feasible. First, the data may be very large, or the index over the data may be very large, and bandwidth and storage requirements for moving content or indexes may be prohibitive. For example, consider a network of digital libraries, each containing multiple gigabytes or terabytes of data. Full text searches can be accomplished efficiently using inverted indexes, but such indexes may be as large as the content itself. In this case, replicating either the content or the indexes will certainly tax network links, and may cause problems if storage is limited at peers. Second, if there are many changes in the system, it will be difficult to keep remote indexes and copies up to date. Frequent content changes, or frequent peer membership changes, will require many updates, again taxing bandwidth resources. Third, many content providers are unwilling to export data or index information for intellectual property reasons. For example, an electronic publisher may be willing to process searches and return results, as long as it can record which searches are being processed over its content or attach copyright notices to the content. Such a publisher will oppose replication, and will probably be resistant to exporting indexing information so that other peers end up processing searches of its content. Not every application has these issues, and in many cases content movement makes sense. However, there are many potential applications where such techniques are not feasible.

Can we still use peer-to-peer search techniques to perform information discovery in these applications? We argue that the peer-to-peer approach can still be used and made efficient. In particular, if we do not proactively move content, but instead leave it at its source, we can avoid the cost of shipping replicas or updates altogether. Unfortunately, existing basic peer-to-peer protocols that do not require content movement, such as Gnutella's original flooding approach, are not scalable or efficient. What is needed is a new set of techniques to optimize peer-to-peer searches without content movement.

As evidence for our argument, we present three techniques that can be used to optimize peer-to-peer searches even when content is not moved. Consider a simple protocol of random walk searches over an unstructured network [1, 11]. Without content movement, the performance of simple random walks can degrade significantly. Our first optimization is to reorganize the overlay network so that random walks can operate efficiently. We propose the *square-root topology*, where each peer's degree is proportional to the square root of the popularity of its content. Our analysis shows that this topology is optimal for simple random walk searches, and simulations show that other search techniques also perform best on the square-root topology[1]. We also provide an adaptive algorithm for forming the square-root topology without using content movement or global information.

We then present two more optimizations to simple random walks in square-root networks. *Biased document count* and *search memory* work to quickly route searches to peers that have the most content, and thus have the highest probability of storing matching content. These optimizations complement the square-root topology to further improve performance. Simulation results show more than a factor of two performance improvement for our techniques over simple random walk searches in power law networks.

Our optimizations are only a starting point, but they illustrate that high performance can be achieved in networks where replicating or moving content is infeasible. There are a few other techniques that also operate without content movement, such as "expanding ring" [11, 20] or "directed breadth first search" [20]. However, more work needs to be done. For instance, our results show that the commonly assumed power-law network is not even the best network for walk-based searches, since the square-root topology is optimal. There are potentially a whole host of new techniques that can be developed to search efficiently without using content movement.

[1]In fact, the square-root topology is often best even when content movement is used; see [6].

In this paper, we first define and analyze the square-root topology (Section 2). Next, we discuss the biased document count and search memory optimizations (Section 3). We present simulation results that show the performance benefit of our techniques (Section 4). We survey related work (Section 5), and then discuss our conclusions (Section 6).

2 The square-root topology

In "unstructured networks," such as that in Gnutella, the topology of the network is built up over time as peers choose neighbors essentially randomly. Without any outside interference, such networks tend toward a power-law distribution, where the number of neighbors of the i^{th} most connected peer is proportional to $1/i^{\alpha}$. Here, α is a constant that determines the skew of the distribution. For such networks, random walk searches have shown to be effective [1, 11]. A simple random walk search starts at one peer in the network, and is processed over that peer's content. That peer then forwards the search to a random neighbor, who processes and forwards the query again. In this way, the search "walks" randomly around the network, until it terminates, either because enough results have been found or because a time-to-live (TTL) has been reached [11].

Consider a peer-to-peer network with N peers. Each peer k in the network has degree d_k (that is, d_k is the number of neighbors that k has). The total degree in the network is D, where $D = \sum_{k=1}^{N} d_k$.

We define the square-root topology as a topology where the degree of each peer is proportional to the square root of the popularity of the peer's content. Formally, if we define g_k as the proportion of searches submitted to the system that are satisfied by content at peer k, then the square-root topology has $d_k \propto \sqrt{g_k}$ for all k.

We now show that a square-root topology is optimal for random walk searches. Imagine a user submits a search s that is satisfied by content at a particular peer k. Of course, until the search is processed by the network, we do not know which peer k is. How many hops will the search message take before it arrives at k, satisfying the search? We can model the search process as a Markov chain. Each state in the Markov chain represents a peer, and the transitions between states represent a search being forwarded from a peer to one of its neighbors. For simple random walk searches, the probability of transitioning from peer i to peer j is $1/d_i$ if i and j are neighbors, and 0 otherwise. Under this formulation, Markov chain theory tells us that the expected number of hops for an arbitrary search to reach its goal peer is proportional to the goal peer's degree:

Lemma 1 *If the network is connected (that is, there is a path between every pair of peers) and non-bipartite, then the expected number of hops for search s to reach peer k is D/d_k.*

This result is shown in [13].

To simplify our analysis, we assume a peer forwards a search message to a randomly chosen neighbor, even if that search message has just come from that neighbor or has already visited that neighbor. Lv et al [11] notes that avoiding previously visited peers can improve the efficiency of walks. Simulation results show that the square-root topology is still best; experiments are discussed in Section 4.

If a given search requires D/d_k hops to reach peer k, how many hops can we expect an arbitrary search to take before it finds results? For simplicity, we assume that a search will be satisfied by a single unique peer; this assumption is relaxed in simulation studies in Section 4. We define g_k to be the probability that peer k is the goal peer; $g_k \geq 0$ and $\sum_{k=1}^{N} g_k = 1$. The g_k will vary from peer to peer. The proportion of searches seeking peer k is g_k, and the expected number of hops that will be taken by peers seeking peer k is D/d_k (from Lemma 1), so the expected number of hops per search over all searches (called H) is:

$$H = \sum_{k=1}^{N} g_k \cdot \frac{D}{d_k} \tag{1}$$

How can we minimize the expected number of hops taken by a search message? It turns out that H is minimized when the degree of a peer is proportional to the square root of the popularity of the peer's content. This is the square-root topology.

Theorem 1 *H is minimized when*

$$d_k = \frac{D\sqrt{g_k}}{\sum_{i=1}^{N} \sqrt{g_i}} \tag{2}$$

Proof sketch We use the method of Lagrange multipliers to minimize equation (1). Our constraint is that $\sum_{k=1}^{N} d_k = D$. Taking the gradient of our constraint, and also of equation (1), and setting them equal to each other gives us a series of N equations of the form $-D \cdot g_k \cdot d_k^{-2} \cdot \hat{\mathbf{u_k}} = \lambda \hat{\mathbf{u_k}}$ where λ is the Lagrange multiplier and $\hat{\mathbf{u_k}}$ is a unit vector. Solving for d_k, and substituting back into the constraint equation (to eliminate λ), gives us the statement of the theorem. The full proof is in [6]. □

Theorem 1 shows that the square-root topology is the optimal topology over a large number of random walk searches. Our analysis shows that D, the total degree in the network, does not impact performance: substituting equation (2) into equation (1) eliminates D. Thus, any value of D that ensures the network is connected is sufficient. Also, the actual topology does not matter, as long as peers have the proper degrees. The result in Lemma 1 is independent of which peers are connected to which other peers.

To construct the square-root topology, each peer k must estimate the popularity of its content (g_k) by dividing Q^k_{match}, the number of queries processed so far that matched the peer's content, by Q^k_{total}, the total number of queries processed by the peer. Since D is unconstrained, we choose $D = d_{max} \cdot \sum_{i=1}^{N} \sqrt{g_i}$, and substituting this equation into equation (2) gives the ideal degree of a peer as $d_k = d_{max} \cdot \sqrt{Q^k_{match}/Q^k_{total}}$. The d_{max} value is a constant we choose and fix as part of the peer-to-peer protocol. Each peer continually tracks its queries and calculates its ideal d_k, and then adds or drops connections to achieve its ideal degree (rounding d_k as necessary). In order to keep the network connected, we also choose a constant d_{min}, which is the minimum number of connections a peer can have.

3 Optimizations to random walks

The square-root topology is optimal for simple random walk searches. But are simple random walk searches the best search strategy for the square-root topology? Previous work [11, 4, 1] has shown that content movement can improve simple random walks significantly. However, we can still optimize random walks for cases where content movement is

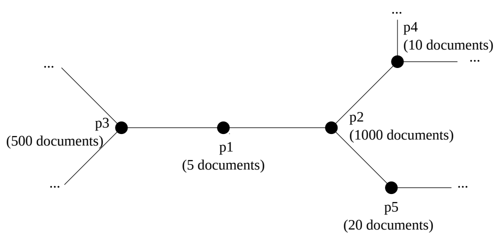

Figure 1: Search memory example.

not feasible. In this section, we describe two optimizations that work together to improve search efficiency for random walks in square-root networks.

3.1 Biased document count

With the *biased document count* technique, peers forward searches to the neighbors that have the most documents. Then, searches are quickly processed over a large amount of content, increasing the probability of finding matches. This technique is similar to biasing random walks toward high degree peers [1], which quickly routes searches to peers that know many other peers (and consequently know about a large amount of content). When it is too expensive for peers to track their neighbors' content, we can do the next best thing: forward queries to peers that have the most content themselves.

3.2 Search memory

Search memory tracks the "best" peers the search has seen so far, and forwards the search directly to those best peers. Consider for example the network fragment shown in Figure 1. Imagine that a search is at peer p_1. This peer has two neighbors, p_2 (with 1,000 documents) and p_3 (with 500 documents). Using the biased document count technique, peer p_1 would forward the query to p_2. Peer p_2 has neighbors p_4 (with 10 documents), p_5 (with 20 documents) and p_1 (with 5 documents). Under the biased document count strategy alone, the search would next be forwarded to p_5. However, if the search message tracks that it has seen, but was not forwarded to, peer p_3 with 500 documents, peer p_2 can determine that p_3 is a better choice than any of its neighbors. Peer p_2 would then send the message to p_3 using UDP or a temporary TCP connection.

Searches are likely to encounter many possible peers along their path, and remembering document counts for all of them will significantly increase the size of the search message. For example, consider a system where peers are identified by their 32 bit IP address and 16 bit port number, and a 16 bit document count is "remembered" for each peer. In our simulations of search memory in a 20,000 peer network, the average search message had to remember 7,460 peers and counts, adding 58 KB on average to the search message size. Since peer-to-peer searches otherwise require a few hundred bytes at most, adding 58 KB per message will prohibitively increase the bandwidth used by search messages.

We can approximate search memory at much lower cost by remembering only the best n peers. For example, if a search message remembers 10 peers, this adds only 80 bytes to the message. Our experimental results (reported in the next section) show that even this limited search memory can result in performance improvement.

With the search memory optimization, search messages are not strictly routed according to the overlay topology. However, the overlay is still important as a mechanism for discovering peers; a search message learns about new peers because they are the neighbors of the current peer. Thus, the square-root topology is still a good network organization, because it ensures the probability that a search message learns about a new peer is in proportion to the popularity of the content at the peer.

4 Experimental results

In this section we present simulation results to confirm our analysis for scenarios where queries may match content at multiple peers. We use simulation because we wish to examine the performance of large networks (i.e., tens of thousands of peers) and it is difficult to deploy that many live peers for research purposes on the Internet.

Our primary metric is to count the total number of messages sent under each search method. We used a discrete-event peer-to-peer simulator that we have developed to model networks with 20,000 peers storing a total of 631,320 documents. A total of 100,000 searches were submitted to random peers in the sys-

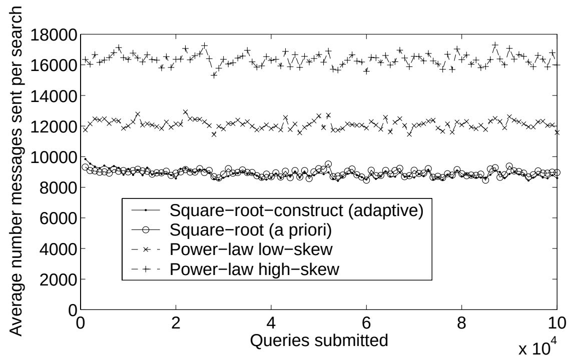

Figure 2: The square-root topology versus power-law topologies.

tem, and each query sought to find 10 results. Because the square-root topology is based on the popularity of documents stored at different peers, it is important to accurately model document and peer popularity; we use a content model based on traces of real documents, peers and queries [5].

First, we conducted an experiment to examine the performance of random walk searches in the square root topology. We generated two square-root topologies: one constructed *a priori* with global knowledge, and another constructed adaptively using only local information at peers (with $d_{max} = 160$, $d_{min} = 3$ and $d_k = 4$ when a peer first joins the network). We compared these topologies to low-skew ($\alpha = 0.58$) and high-skew ($\alpha = 0.74$) power-law networks, both generated using the PLOD algorithm [15].

Figure 2 shows the number of messages per search, calculated as a running average every 1,000 queries. As the figure shows, the adaptive square-root topology quickly converges to the ideal *a priori* square root topology (after about 8,000 queries). The square-root topology is significantly better than the power-law topologies, requiring 26 percent fewer messages than the low-skew network, and 45 percent fewer messages than the high-skew network.

Other results (not shown) indicate that the square-root topology is in fact better than a power-law topology for several other types of peer-to-peer routing techniques, and when statekeeping [11] is used. In fact, the square-root topology is often best even when content movement is allowed. Detailed results are reported in [6].

Next, we conducted an experiment to measure the

Routing	*Topology*	*Msgs per search*
Random walk	Power-law (high skew)	16340
Random walk	Power-law (low skew)	12110
Random walk	**Square-root**	8850
Doc count	**Square-root**	7780
Doc count + memory	**Square-root**	7030

Table 1: Results for optimizations

effect of the biased document count and search memory optimizations for searches in the square-root topology. Table 1 shows the results averaged over 100,000 queries. As the table shows, using the biased document count and limited memory optimizations provided good performance, with 21 percent fewer messages than random walks in the square-root topology. Even though we used limited memory, we achieved high performance; for comparison, unlimited search memory only reduced the message cost by a further 3 percent in our experiments. The combination of all three of our techniques (square-root topology, biased document count and limited memory) results in 42 percent fewer messages than random walks in the low skew power-law topology, and 57 percent fewer messages than random walks in the high-skew power-law topology. Clearly, it is possible to achieve high performance even without content movement.

Other optimizations may be possible to further improve performance, and examining such optimizations is worthy of further study.

5 Related work

A variety of techniques for efficient peer-to-peer searches have been proposed. Many investigators have proposed ways to move or replicate content, or replicate indexes over content, in order to improve performance [11, 4, 20, 8, 3, 2, 7, 19, 17, 18, 21, 14, 4]. For applications where content movement is too expensive or resisted by peers, other techniques must be developed. There have been several proposed techniques that do not use content movement, such as expanding ring [11] or directed breadth first search [20]. We argue that these techniques are just a starting point, and that there is unexplored potential for further significant performance enhancements.

Some investigators have looked at building efficient topologies for peer-to-peer searches. For example, Pandurangan et al [16] discuss building low diameter networks for efficient flooding. However, random walk searches have been shown to be more scalable than flooding [11]. Lv et al [12] presented a dynamic algorithm for load balancing when random walks are used. It may be possible to combine these techniques with our square-root topology in order to take both popularity and peer capacity into account.

Several investigators have examined peer-to-peer systems analytically, including models for peer behavior [9], download traffic [10], and so on. To our knowledge, there have been no published analytical results on the optimal topology for random walk searches.

6 Conclusions

We have argued that new techniques must be developed to deal with networks where it is infeasible to move or replicate content. Although many of the most effective techniques developed so far utilize content movement, we believe that progress can be made on efficient searching while leaving content at its original peer. We have presented three techniques as support for our assertion, and as a starting point for further investigation. First, we have shown that for simple random walk searches, the optimal topology is a square-root topology, not a power-law network. This topology can be constructed using purely local information at peers. Second, biasing searches towards peers with a large amount of content further improves performance. Third, adding search memory allows messages to be quickly routed to the best peers. These techniques show the viability of further research into routing in unstructured networks, even when we cannot move or replicate content.

References

[1] L. Adamic, R. Lukose, A. Puniyani, and B. Huberman. Search in power-law networks. *Phys. Rev. E*, 64:46135–46143, 2001.

[2] M. Bawa, R. J. Bayardo Jr., S. Rajagopalan, and E. Shekita. Make it fresh, make it quick — searching a network of personal webservers. In *Proc. WWW*, 2003.

[3] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker. Making Gnutella-like P2P systems scalable. In *Proc. SIGCOMM*, 2003.

[4] E. Cohen and S. Shenker. Replication strategies in unstructured peer-to-peer networks. In *Proc. SIGCOMM*, 2002.

[5] B. F. Cooper. A content model for evaluating peer-to-peer searching techniques. In *Proc. ACM/IFIP/USENIX Middleware Conference*, 2004.

[6] B. F. Cooper. Random walks revisited: optimizing topologies for peer-to-peer searches. Technical report, available at http://www.cc.gatech.edu/˜cooperb/-pubs/squareroot.pdf, October 2004.

[7] B.F. Cooper and H. Garcia-Molina. Studying search networks with SIL. In *Proc. IPTPS*, 2003.

[8] A. Crespo and H. Garcia-Molina. Routing indices for peer-to-peer systems. In *Proc. ICDCS*, 2002.

[9] Z. Ge, D.R. Figueiredo, S. Jaiswal, J. Kurose, and D. Towsley. Modeling peer-peer file sharing systems. In *Proc. INFOCOM*, 2003.

[10] K.P. Gummadi, R.J. Dunn, S. Saroiu, S.D. Gribble, H.M. Levy, and J. Zahorjan. Measurement, modeling and analysis of a peer-to-peer file-sharing workload. In *Proc. SOSP*, 2003.

[11] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *Proc. Int'l Conf. on Supercomputing (ICS)*, 2002.

[12] Q. Lv, S. Ratnasamy, and S. Shenker. Can heterogeneity make Gnutella scalable? In *Proc. IPTPS*, 2002.

[13] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, New York, NY, 1995.

[14] W. Nejdl, M. Wolpers, W. Siberski, C. Schmitz, M. Schlosser, I. Brunkhorst, and A. Loser. Super-peer-based routing and clustering strategies for RDF-based peer-to-peer networks. In *Proc. WWW*, 2003.

[15] C. Palmer and J. Steffan. Generating network topologies that obey power laws. In *Proc. GLOBECOM*, 2000.

[16] G. Pandurangan, P. Raghavan, and E. Upfal. Building low-diameter P2P networks. In *Proc. IEEE FOCS*, 2001.

[17] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proc. SIGCOMM*, Aug. 2001.

[18] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In *Proc. IFIP/ACM International Conference on Distributed Systems Platforms*, 2001.

[19] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proc. SIGCOMM*, Aug. 2001.

[20] B. Yang and H. Garcia-Molina. Efficient search in peer-to-peer networks. In *Proc. ICDCS*, 2002.

[21] B. Yang and H. Garcia-Molina. Designing a super-peer network. In *Proc. ICDE*, 2003.