An Optimal Overlay Topology for Routing Peer-to-Peer Searches

Brian F. Cooper

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

Abstract. Unstructured peer-to-peer networks are frequently used as the overlay in various middleware toolkits for emerging applications, from content discovery to query result caching to distributed collaboration. Often it is assumed that unstructured networks will form a power-law topology; however, a power-law structure is not the best topology for an unstructured network. In this paper, we introduce the *square-root topology*, and show that this topology significantly improves routing performance compared to power-law networks. In the square-root topology, the degree of a peer is proportional to the square root of the popularity of the content at the peer. Our analysis shows that this topology is optimal for random walk searches. We also present simulation results to demonstrate that the square-root topology is better, by up to a factor of two, than a power-law topology for other types of search techniques besides random walks. We then describe a decentralized algorithm for forming a square-root topology, and evaluate its effectiveness in constructing efficient networks using both simulations and experiments with our implemented prototype. Our results show that the square-root topology can provide a significant performance improvement over power-law topologies and other topology types.

Keywords: peer-to-peer search, overlay topology, random walks.

1 Introduction

Peer-to-peer search networks have gone from serving as application-specific overlays to become generally useful components in systems for finding and distributing content. In particular, "unstructured" peer-to-peer networks, such as those in Gnutella and Kazaa, continue to remain popular and widely deployed. Even with the advent of more "structured" networks for content-based routing (such as [1,2,3]), unstructured networks continue to be important, both because of their usefulness for content discovery [4] and because they can be used together with structured networks in so-called hybrid systems [5,6]. Several types of systems have an unstructured topology as a sub-network: superpeer networks [7] use an unstructured topology to connect the superpeers, caching networks [8] use an unstructured topology to connect caches, scientific collaboration networks [9] use an unstructured topology to locate data sets, and so on. Since a variety of middleware tools implement an unstructured peer-to-peer network, it is important to investigate techniques for optimizing unstructured topologies.

G. Alonso (Ed.): Middleware 2005, LNCS 3790, pp. 82–101, 2005.

Unstructured networks tend toward power-law topologies, and several techniques for searching in power-law topologies have been developed. One especially effective technique is to conduct a "random walk," where each peer forwards a search message to a random neighbor until results have been found [10,11,12]. This technique requires far fewer messages than Gnutella's original flooding-based algorithm, and results have shown that random walk searches are a scalable and effective way to find content in a peer-to-peer network.

Although these techniques have been developed to work with power-law topologies, a power-law network is not the best network for a random walk. Implementing a protocol that causes the network to converge to a more efficient topology can significantly improve search performance. In this paper, we introduce the *square-root topology*, where the degree of each peer is proportional to the square root of the popularity of the content at the peer (measured in terms of the number of submitted searches that match the peer's content). We present analysis based on random walks in Markov chains to show that the square-root topology is not only better than power-law networks, it is in fact optimal in the number of hops needed to find content. Intuitively, the probability that a random walk quickly reaches a peer is proportional to the degree of the peer, and if peers with popular content have correspondingly high degrees, then most searches will quickly reach the right peers and find matching content. Simulation results confirm our analysis, showing that a random walk requires up to 45 percent fewer hops in a square-root topology than in a power-law topology.

We also present simulation results to show that several other walk-based techniques perform better in a square-root topology than in a power-law topology. One technique is suggested by Adamic et al [10], who propose biasing random walks toward high degree peers. If peers track their neighbors' content, then high degree peers will have knowledge of the content of many peers, and searches will quickly be evaluated over a large amount of content. Another technique is suggested by Lv et al [11], who argue for starting multiple parallel random walks for the same search. This technique reduces the time before searches complete, though it requires roughly the same total number of messages. A third technique is to bias random walks based on previous results from peers, as suggested by Yang and Garcia-Molina [13]. In each case, the square-root topology performs better than a power-law topology, decreasing the number of messages per search by as much as 50 percent.

Next, we introduce a decentralized algorithm, *square-root-construct*, for building and maintaining the square-root topology as peers join and leave the system. Each peer uses purely local information to estimate the popularity of its content, avoiding the need for tracking the global distribution of popularities among peers. Then, each peer adds or drops connections to other peers to achieve its optimal degree. Simulation results as well as experiments using our implemented peer-to-peer system prototype demonstrate the performance advantages of the square-root topology. For example, in a network of 1,000 peers running on a cluster in our lab, a random topology required more than twice the bandwidth of a topology maintained using *square-root-construct*.

A related result to the square-root topology was obtained by Cohen and Shenker [14], who suggested that content be replicated proactively to improve search efficiency. Their result showed that the optimal replication was the *square-root replication*, where

the number of copies of a content object is proportional to the square root of the object's popularity. Our results are complementary, as we deal with the number of neighbors each peer has rather than the number of copies of each document. In particular, our square-root topology can be used in cases where a square-root replication is not feasible, such as applications where there are high storage and bandwidth costs for replicating content. Moreover, in cases where square-root replication is used, a square-root topology still provides better efficiency than a power-law topology, with an improvement of more than 50 percent.

We are implementing a flexible peer-to-peer content location middleware toolkit, called *Overlay-Dynamic Information Networks (ODIN)*. ODIN can be layered on top of existing data repositories (such as document repositories, local filesystems or scientific databases) to connect these repositories into a large scale searching network for use by different applications. The square-root topology forms the basis of the overlay networks constructed in ODIN. In this paper, we focus on the square-root topology, and show its usefulness for a wide range of different searching techniques that might be employed by peer-to-peer middleware like ODIN. In particular, our contributions include:

- We define the *square-root topology*, and give analysis based on random walks in Markov chains to show that a square-root topology is optimal for random walk searches. (Section 2)
- We present simulation results to show that a square-root topology is better than a power-law topology for a variety of search techniques, and when square-root replication is used. (Section 3)
- We develop a distributed algorithm, *square-root-construct*, for dynamically building the square-root topology based on purely local information available to a peer. (Section 4)
- We present results from simulations and from our prototype that demonstrate the effectiveness of *square-root-construct* for constructing efficient topologies. (Section 5)

We examine related work in Section 6, and present our conclusions in Section 7.

2 Network Topologies

Random walk searches were initially introduced as a way to optimize searches in power-law networks [10], and recent research often takes the power-law topology as a given (see for example [11,7]). While random walk searches are better than Gnutella-style search broadcasts in power-law networks, power-law networks are not the best structure for random-walk searches. In this section, we provide analysis showing that square-root networks provide optimal performance for random walk searches, and thus are better than power-law networks. Our analysis is backed up with simulation results for different scenarios in Section 3.

2.1 Background

A peer-to-peer search network is a partially connected overlay of peers, sitting on top of a fully connected underlying network (such as the Internet.) The main reason to keep the

overlay network partially connected is to reduce the state that each peer must maintain. Since each peer only has to stay connected to a few neighbors, no peer has to know about all of the peers in the system or understand the whole topology. Furthermore, a peer only needs to react to changes concerning its immediate neighbors; changes to remote parts of the topology do not directly affect peers. This limited state and localized impact of changes improves scalability, even when there is a high amount of peer *churn*, with many peers joining and leaving the system.

The topology of the overlay network is built up over time in a decentralized way. Peers that join the system connect to peers that are already in the system, and the choice of neighbors is essentially random in many existing systems. Topologies in these systems tend toward a power-law distribution, where some long-lived peers have many connections while most peers have a few connections. Formally, in a power-law network, the number of neighbors of the i^{th} most connected peer is proportional to $1/i^{\alpha}$, where α is a constant that determines the skew of the distribution. Larger α results in more skew.

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 one or a subset of its neighbors, who each process and forward the query. In this way, the search "walks" around the network, until it terminates according to some stopping criterion. There are several alternatives for terminating the walk [11]: a walk can be given a *time-to-live* which limits the number of hops the walk makes, or the walk can terminate after G results have been found, where G is a user-defined parameter (the "goal"). Several researchers have adapted random walk searches in various ways to make them less random and more efficient. We examine these adaptations in more detail in Section 3.

2.2 The Square-Root Topology

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 $D = \sum_{k=1}^{N} d_k$. Equivalently, the total number of connections in the network is $D/2$.

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 a 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? The expected length of the random walk (called the *hitting time* or *mean first passage time*) depends on the degree of k:

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 [15], and is derived using the properties of Markov chains. We now briefly summarize the reasoning behind the lemma. A Markov chain consists of a

set of states, where the probability of transitioning from state i to state j depends only on i and j, and not on any other history about the process. For our purposes, the states of the Markov chain are the peers in the system, and $1 \leq i, j \leq N$. Associated with a Markov chain is a transition matrix T that describes the probability that a transition occurs from a state i to another state j. In our context, this transition probability is the probability that a search message that is at peer i is next forwarded to peer j. With simple random walks, the transition probability from peer i to peer j is $1/d_i$ if i and j are neighbors, and zero otherwise. The result in [15] depends only on the node degrees, and not on the structure; that is, the expected length of a walk does not depend on which peers are connected to which other peers. This property follows from the fact that the Markov chain converges to the same stationary distribution regardless of which vertices are connected.

This model assumes peers forward search messages to a randomly chosen neighbor, even if that search message has just come from that neighbor or has already visited this neighbor. This assumption simplifies the Markov chain analysis. Previous proposals for random walks [11] have noted that avoiding previously visited peers can improve the efficiency of walks, and we examine this possibility in simulation results in the next section.

Using the transition matrix, we can calculate the probability that a search message is at a given peer at a given point in time. First, we define an N element vector V_0, called the *initial distribution vector*; the k^{th} entry in V represents the probability that a random walk search starts at peer k. The entries of V sum to 1. Given T and V_0, we can calculate V_1, where the k^{th} entry represents the probability of the search being at peer k after one hop, as $V_1 = TV_0$. In general, the vector V_m, representing the probabilities that a search is at a given peer after m hops, is recursively defined as $V_m = TV_{m-1}$.

Under the conditions of the lemma (the network is connected and non-bipartite), V_m converges to a *stationary distribution vector* V_s, representing the probability that a random walk search visits a given peer at a particular point in time. Most importantly for our purposes, it can be shown [15] that the k^{th} entry of V_s is d_k/D. In other words, in the steady state, the probability that a search message is at a given peer k is d_k/D.

What is the expected number of hops before a search reaches its goal? We can treat the search routing as a series of experiments, each choosing a random peer k from the population of N peers with probability d_k/D. A "successful" experiment occurs when a search chooses a peer with matching content. The expected number of experiments before the search message successfully reaches a particular peer k is a geometric random variable with expected value $\frac{1}{d_k/D} = \frac{D}{d_k}$. This is the result given by Lemma 1.

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. 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 taken by 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 documents at that peer. 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. We use the method of Lagrange multipliers to minimize equation (1). Recall the constraint that all degrees d_k sum to D; that is, the constraint for our optimization problem is $f = (\sum_{k=1}^{N} d_k) - D = 0$. We must find a Lagrange multiplier λ that satisfies $\nabla H = \lambda \nabla f$ (where ∇ is the gradient operator). First, treating the g_k values as constants,

$$\nabla H = \sum_{k=1}^{N} -D \cdot g_k \cdot d_k^{-2} \cdot \hat{\mathbf{u}_\mathbf{k}} \tag{3}$$

where $\hat{\mathbf{u}_\mathbf{k}}$ is a unit vector. Next,

$$\lambda \nabla f = \lambda \sum_{k=1}^{N} \hat{\mathbf{u}_\mathbf{k}} = \sum_{k=1}^{N} \lambda \hat{\mathbf{u}_\mathbf{k}} \tag{4}$$

Because $\nabla H = \lambda \nabla f$, we can set each term in the summation of equation (3) equal to the corresponding term of the summation of equation (4), so that $-D \cdot g_k \cdot d_k^{-2} \cdot \hat{\mathbf{u}_\mathbf{k}} = \lambda \hat{\mathbf{u}_\mathbf{k}}$. Solving for d_k gives

$$d_k = \frac{\sqrt{D \cdot g_k}}{\sqrt{-\lambda}} \tag{5}$$

Now we will eliminate λ, the Lagrange multiplier. Substituting equation (5) into f gives

$$\sum_{k=1}^{N} \left(\frac{\sqrt{D \cdot g_k}}{\sqrt{-\lambda}}\right) = D \tag{6}$$

and solving gives

$$\frac{1}{\sqrt{-\lambda}} = \frac{D}{\sqrt{D}\sum_{k=1}^{N}\sqrt{g_k}} \tag{7}$$

If we change the dummy variable of the summation in equation (7) from k to i, and substitute back into equation (5), we get equation (2). □

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. Note also that our result holds regardless of which peers are connected to which other peers, because of the properties of the stationary distribution of Markov chains.

Finally, peer degrees must be integer values; it is impossible to have a third of a connection for example. Therefore, the optimal peer degrees must be calculated by rounding the value calculated in equation (2).

3 Experimental Results for the Square-Root Topology

Our analysis of the square-root topology is based on an idealized model of searches and content. Real peer-to-peer systems are less idealized; for example, searches may match content at multiple peers. In this section we present simulation results to illustrate the performance of a square-root topology for realistic scenarios. 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. Searches terminate when "enough" results were found, where "enough" is defined as a user specified goal number of results G. In summary, our results show:

- Random walks perform best on the square-root topology, requiring up to 45 percent fewer messages than in a power-law topology. The square-root topology also results in up to 50 percent less search latency than power-law networks, even when multiple random walks are started in parallel.
- The square-root topology is the best topology when proactive replication is used, and the combination of square-root topology and square-root replication provides higher efficiency than either technique alone.
- Other search techniques based on random walks, such as biased high-degree [10], biased towards most results or fewest result hop neighbors [13], and random walks with statekeeping [11] performed best on the square-root topology, decreasing the number of messages sent by as much as 52 percent compared to a power-law topology.
- The square-root topology performed better than other topology structures as well, including a constant degree network, and a topology with peer degrees directly proportional to peer popularity. In super-peer networks [7] the square-root was the best topology for connecting the supernodes.

In this section, we first describe our experimental setup, and then present our results.

3.1 Experimental Setup

Our experimental results were obtained using a discrete-event peer-to-peer simulator that we have developed. Our simulator models individual peers, documents and queries, as well as the topology of the peer-to-peer overlay. Searches are submitted to individual peers, and then walk around the network according to the specified routing algorithm. Our simulations used networks with 20,000 peers. Simulation parameters are listed in Table 1.

Because the square-root topology is based on the popularity of documents stored at different peers, it is important to accurately model the number of queries that match each document, and the peers at which each document is stored. It is difficult to gather accurate and complete query, document and location data for tens of thousands of real peers. Therefore, we use the content model described in [16], which is based on a trace of real queries and documents, and more accurately describes real systems than simple uniform or Zipfian distributions. In particular, we downloaded text web pages from

Table 1. Experimental parameters

Parameter	*Value*
Number of peers	20,000
Documents	631,320
Queries submitted	100,000
Goal number of results	10
Average links per peer	4
Minimum links per peer	1

1,000 real web sites, and evaluated keyword queries against the web pages. We then generated 20,000 synthetic queries matching 631,320 synthetic documents, stored at 20,000 peers, such that the statistical properties of our synthetic content model matched those of the real trace. The resulting content model allowed us to simulate a network of 20,000 peers. In our simulation, we repeatedly submitted random queries chosen from the set of 20,000 to produce a total of 100,000 query submissions. In [16] we describe the details of this method of generating synthetic documents and queries, and provide experimental evidence that the content model, though synthetic, results in highly accurate simulation results. Most importantly, the synthetic model retains an accurate distribution of the popularity of peer content, which is critical for the construction of the square-root topology.

3.2 Random Walks

First, we conducted an experiment to examine the performance of random walk searches in different topologies. In this experiment, queries matched documents stored at different peers, and had a goal $G = 10$ results. We compared three different topologies:

- A *square-root topology*, generated by assigning a degree to each peer based on equation (2), and then creating links between randomly chosen pairs of peers based on the assigned degrees.
- A *low-skew power-law topology*, generated using the PLOD algorithm [17]. In this network, $\alpha = 0.58$.
- A *high-skew power-law topology*, generated using the PLOD algorithm, with $\alpha = 0.74$.

The results of our experiment are shown in Figure 1. As the figure shows, random walks in the square-root topology require 8,940 messages per search, 26 percent less than random walks in the low-skew power-law topology (12,100 messages per search) and 45 percent less than random walks in the high-skew power-law topology (16,340 messages per search). In the power-law topologies, searches tend toward high degree peers, even if the walk is truly random and not explicitly directed to high degree peers (as in [10]). Unless these high degree peers also have the most popular content, the result is that searches have a low probability of walking to the peer with matching content, and the number of hops and thus messages increases. If the power-law distribution is more skewed, then the probability that searches will congregate at the wrong peers is higher and the total number of messages necessary to get to the right peers increases.

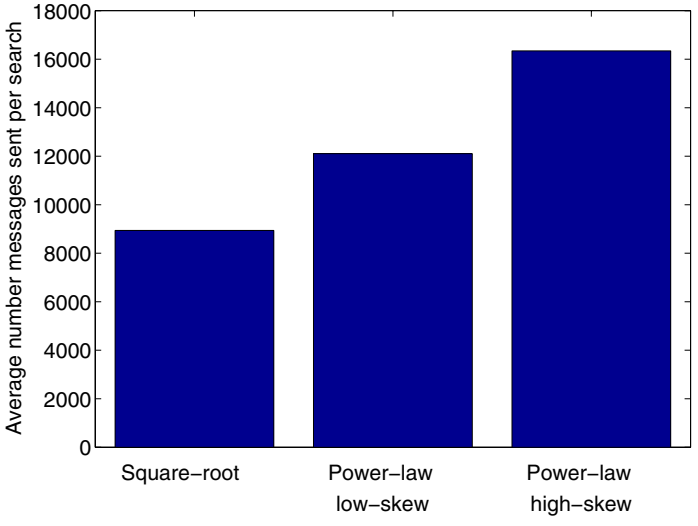

Fig. 1. Random walk searches on different topologies

Table 2. Parallel random walks: search latency (ticks)

Walks	*Square-root*	*Power-law* low-skew	*Power-law* high-skew
1	8930	12090	16350
2	4500	6210	8970
5	1800	2490	3740
10	904	1250	1880
20	454	630	947
100	96	130	194

Even though random walks perform best in the square-root topology, a large number of messages need to be sent (8,940 messages in a network of 20,000 peers in the above results). However, this result is a significant improvement over traditional Gnutella-style search: flooding in a high-skew power-law network, with a TTL of five in order to find at least ten results on average, requires 17,700 messages per search. Moreover, the above results are for simple, unoptimized random walks. Adding optimizations such as proactive replication or neighbor indexing significantly reduces the cost of a random walk search, and results for these techniques (presented in the next sections) show that the square-root topology is still best.

Another issue with random walks is that the search latency is high, as queries may have to walk many hops before finding content. To deal with this, Lv et al [11] propose creating multiple, parallel random walks for each search. Since the network processes these walks in parallel, the result is significantly reduced search latency (even though the total number of messages is not reduced). We ran experiments where we created 2, 5, 10, 20, and 100 parallel random walks for each search, and measured search latency as

the number of simulation time ticks required to find the goal content (one tick represents the time to process a search and forward it one hop.) These results are shown in Table 2.

As the table shows, the square-root topology provided the lowest search latency, regardless of the number of parallel walks that were generated. The improvement for the square-root topology was consistently 27 percent compared to the low-skew power-law topology, and 50 percent compared to the high-skew power-law topology. Even when searches are walking in parallel, the square root topology helps those search walks quickly arrive at the peers with the right content.

3.3 Proactive Replication

The square-root topology is complementary to the square-root replication described in [14]. In situations where it is feasible to proactively replicate content, the square-root replication specifies that the number of copies made of content should be proportional to the square root of the popularity of the content. The square-root topology can be used whether or not proactive replication is used, but the combination of the two techniques can provide significant performance benefits.

We conducted an experiment where we proactively replicated content according to the square-root replication. Each peer was assigned capacity equal to twice the content they were already storing, and this extra capacity was used to store proactively replicated copies. We then connected peers in the square-root, high-skew power-law, and low-skew power-law topologies, and measured the performance of random walk searches. Again, $G = 10$.

The results are shown in Figure 2. As expected, proactive replication provided better performance than no replication (e.g., Figure 1). Proactive replication performs best with the square-root topology, requiring only 2,830 messages per search, 42 percent

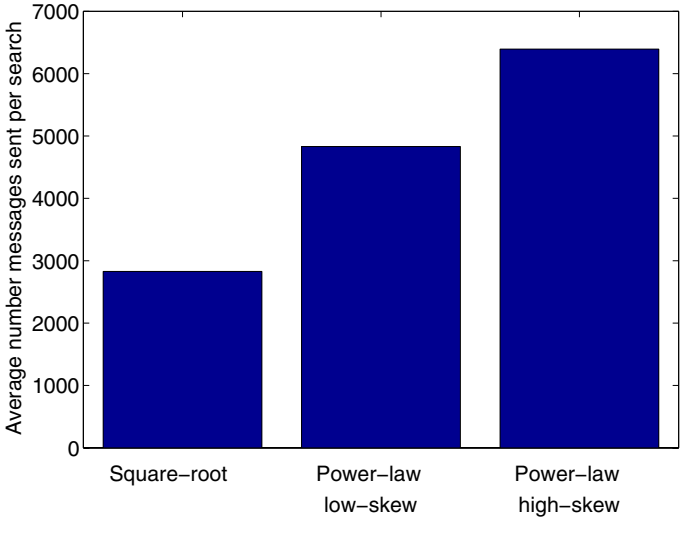

Fig. 2. Random walk searches with proactive replication

less than in the low-skew power-law network (4,830 messages) and 56 percent less than in the high-skew power-law network (6,390 messages). Proactive replication makes more copies of the documents that a search will match, while the square-root topology makes it easier for the search to get to the peers where the documents are stored. The combination of the two techniques provides more efficiency than either technique alone. For example, in our experiment, the square root topology with proactive replication required 68 percent fewer messages than the square root topology without replication.

3.4 Other Search Walk Techniques

Next, we examined the performance of other walk-based techniques on different topologies. We compared three other techniques based on random walks:

- *Biased high degree*: messages are preferentially forwarded to neighbors that have the highest degree [10].
- *Most results*: messages are forwarded preferentially to neighbors that have returned the most results for the past 10 queries [13].
- *Fewest result hops*: messages are forwarded preferentially to neighbors that returned results for the past 10 queries who have travelled the fewest average hops [13].

In each case, ties are broken randomly. For the biased high degree technique, we examined both neighbor-indexing (peers track their neighbors' content) and no neighbor-indexing. Although [13] describes several ways to route searches in addition to most results and fewest result hops, these two techniques represent the "best" that the authors studied: fewest result hops requires the least bandwidth, while most results has the best chance of finding the requested number of matching documents.

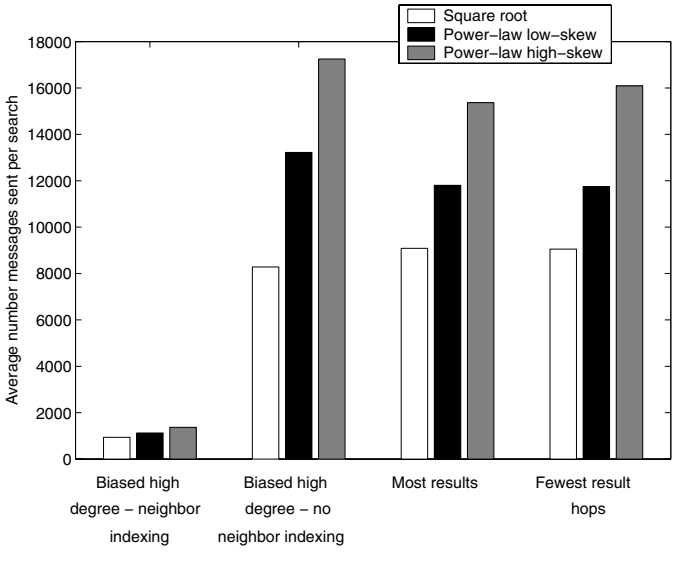

Fig. 3. Other walk-based search techniques

The results are shown in Figure 3. As the figure shows, in each case the square-root topology is best. The most improvement is seen with the biased high degree technique, where the improvement on going from the high-skew power-law topology (17,250 messages on average) to the square-root topology (8,280 messages on average) is 52 percent. Large improvements are achieved with the fewest result hops technique (44 percent improvement versus the high-skew power-law topology) and most results (41 percent improvement versus the high-skew power-law topology). The smallest improvement observed was for the biased high degree technique with neighbor indexing; the square-root topology offers a 16 percent decrease in messages compared to the low-skew power-law topology. Overall, the square-root topology provides the best performance, even with the extremely efficient biased high degree/neighbor indexing combination. Moreover, the square-root topology can be used even when neighbor indexing is not feasible.

The combination of square-root topology, square-root replication and biased high degree walking with neighbor indexing provides even better performance. Our results (not shown) indicate that this approach is extremely efficient, requiring only 248 messages per search on average. Again, the square-root topology is better than the power-law topology when square-root replication and neighbor indexing are used. Using all three techniques together results in a searching mechanism that contacts less than 2 percent of the system's peers on average while still finding sufficient results.

Finally, the results so far assume *state-keeping* [11], where peers keep state about where the search has been. Then, peers can avoid forwarding searches to neighbors that the search has already visited. We also ran experiments for no statekeeping. The results (not shown) demonstrate that the square-root topology is better than power-law topologies, whether or not statekeeping is used.

3.5 Other Topologies

We also tested the square-root topology in comparison to several other network structures. First, we compared against two simple structures:

- *Constant-degree topology*: every peer has the same number of neighbors. In our simulations, each peer had five neighbors.
- *Proportional topology*: every peer had a degree proportional to their popularity g_k (rather than proportional to $\sqrt{g_k}$ as in the square-root topology).

Our results show that the square-root topology is best, requiring 10 percent fewer messages than the constant degree network, and 7 percent fewer messages than the proportional topology. Although the improvement is smaller than when comparing the square-root topology to power-law topologies, these results again demonstrate that the square-root topology is best. Moreover, the cost of maintaining the square-root topology is low, as we discuss in Section 4, requiring easily obtainable local information. Thus, it clearly makes sense to use the square-root topology instead of constant degree or proportional topologies.

A widely used topology in many systems is the super-peer topology [7,18]. In this topology, a fraction of the peers serve as super-peers, aggregating content information from several "leaf" pears. Then, searches only need to be sent to super-peers. The super-peers are connected using a normal unstructured topology (which, like other topologies,

tends to form into a power-law structure). We ran simulations using a standard super-peer topology, in which searches are flooded to super-peers. We compared this standard topology to a super-peer topology that used the square-root topology and random walks between super-peers. The results indicate a significant improvement using our techniques: the square-root super-peer network required 54 percent fewer messages than a standard super-peer network.

4 Constructing Square-Root Networks

In order for the square-root topology to be useful in peer-to-peer systems, there must be a lightweight, distributed algorithm for constructing the topology. We cannot expect a centralized planner to organize peers into the square root topology, nor can we expect individual peers to keep a large amount of state about the rest of the network. In particular, it is too costly in a large network to expect each peer to track all of the queries in the network or the popularity of content at all the other peers in order to compute equation (2). In this section, we describe an algorithm, called *square-root-construct*, that allows peers to construct the square-root topology in a distributed manner, using only local information.

In our algorithm, when peers join the network, they make random connections to some number of other peers. The number of initial connections that peer k makes is denoted d_k^0. The actual value of d_k^0 is not as important as the fact that peers make enough connections to keep the network connected. Then, as peer k is processing queries, it gathers information about the popularity of its content. From this information, peer k calculates its first estimate of its ideal degree, d_k^1. If the ideal degree d_k^1 is more than d_k^0, peer k adds $d_k^1 - d_k^0$ connections, and if the ideal degree is less than d_k^0, peer k drops $d_k^0 - d_k^1$ connections. Over time, peer k continues to track the popularity of its content, and recomputes its ideal degree ($d_k^2, d_k^3 ...$). Whenever its ideal degree estimate is different from its actual degree, peer k adds or drops connections. As in other peer-to-peer systems, peers can find new neighbors using a hostcatcher at a well known address, or by caching peer addresses from network messages.

Peers use purely local information to estimate the popularity of their content. In particular, each peer k maintains two counters: Q_{total}^k, the total number of queries seen by k, and Q_{match}^k, the number of queries that match k's content. Then, peers can estimate g_k in equation (2) as $Q_{match}^k / Q_{total}^k$. As peer k sees more and more queries, it can continue to recompute its estimate of g_k in order to calculate successive estimates of its ideal degree.

It is much more difficult to estimate the denominator of equation (2), which is the sum of the square roots of the popularity of all of the peers. Luckily, we can avoid this problem, since we have another degree of freedom: D, the sum of the d_k values for all peers. Recall from our analysis in Section 2.2 that D does not impact the overall performance of the system, as long as the system remains connected. Therefore, we can choose $D \propto \sum_{i=1}^{N} \sqrt{g_i}$, and substituting such a D into equation (2) eliminates $\sum_{i=1}^{N} \sqrt{g_i}$. More formally, we choose a maximum degree d_{max}, representing the degree we want for a peer whose popularity $g_k = 1$. Of course, it is unlikely that any peer will have content matching all queries, so the actual largest degree will almost certainly be less than d_{max}. Then, we define D as:

$$D = d_{max} \cdot \sum_{i=1}^{N} \sqrt{g_i} \tag{8}$$

Substituting equation (8) into equation (2) gives the ideal degree of a peer as:

$$d_k = d_{max} \cdot \sqrt{g_k} \approx d_{max} \cdot \sqrt{Q^k_{match}/Q^k_{total}} \tag{9}$$

If the popularity of a peer's content is very low, then d_k will be very small. If peer degrees are too small, the network can become partitioned, which will prevent content at some peers from being found at all. In the worst case, because d_k must be an integer, we must round equation (9), so the ideal degree might be zero. Therefore, we define a value d_{min}, which is the minimum degree a peer will have. The degree a peer will aim for is:

$$d_k = \begin{cases} \text{round}(d_{max} \cdot \sqrt{Q^k_{match}/Q^k_{total}}) & \text{if greater than } d_{min} \\ d_{min} & \text{otherwise} \end{cases} \tag{10}$$

Our algorithm *square-root-construct* can be summarized as follows:

- We choose a maximum degree d_{max} and minimum degree d_{min}, and fix them as part of the peer-to-peer protocol.
- Peer k joins, and makes some number d^0_k of initial connections; $d_{min} \leq d^0_k \leq d_{max}$.
- Peer k tracks Q^k_{match} and Q^k_{total}, and continually computes d_k according to equation (10).
- When the computed d_k differs from peer k's actual degree, k adds or drops connections.

Eventually, this method will cause the network to converge to the square root topology; as peers see more queries their estimates of their popularity will become increasingly accurate. Simulation results in the next section show that the network converges fairly quickly to an efficient structure.

Our algorithm also deals with situations where peer popularities change. Then peers will see more or fewer matching queries for their content, and will adjust their g_k estimates and degrees accordingly. In this situation, we may decide to use a decay factor μ to decrease the importance of older information in the estimate of g_k ($0 \leq \mu \leq 1$). Periodically, peer k would multiply both Q^k_{match} and Q^k_{total} by μ. Then, newer samples would have greater weight, and the network would converge more quickly according to the new distribution of popularities.

5 Experimental Results for the *Square-Root-Construct* Algorithm

We conducted two experiments to evaluate the effectiveness of square-root-construct. First, we ran simulations with 20,000 peers. Then, we validated our simulation results by running an experiment with our implemented peer-to-peer prototype in a network with 1,000 peers. Both experiments show that the *square-root-construct* algorithm effectively produces an efficient square-root topology.

5.1 Simulation Results

We ran simulations to measure the performance of searches over time as the topology adapted under the *square-root-construct* algorithm, and compared the performance to searches in square-root and power-law topologies constructed *a priori* using complete knowledge about peers and queries. We used the same experimental setup as described in Section 3. The parameters for the *square-root-construct* algorithm are shown in Table 3. We experimented with several parameter settings, and found that these settings worked well in practice. In particular, they produced connected networks with approximately the same total degree as the networks from experiments in Section 3.

Figure 4 shows the number of messages per search, calculated as a running average every 1,000 queries. As the figure shows, initially the performance of the network being adaptively constructed with the *square-root-construct* algorithm is not quite as good as the *a priori* square-root topology. However, the performance quickly improves, and after about 8,000 queries the performance of the adaptive square-root topology is consistently as good as the topology constructed *a priori*. (Other experiments show that the time for convergence to the performance of the *a priori* structure varies linearly with the number of peers in the network.) The *square-root-construct* network already performs

Table 3. Parameters for *square-root-construct*

Parameter	*Value*
d_{max}	160
d_{min}	3
d_k^0	4

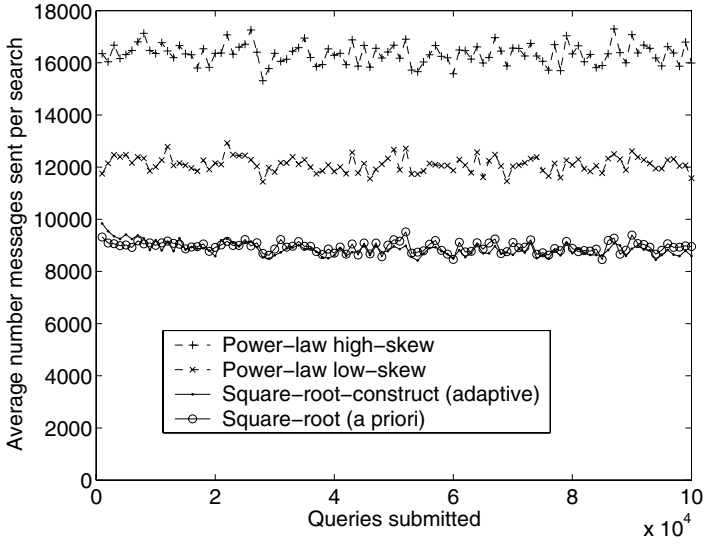

Fig. 4. Square-root-construct versus topologies constructed *a priori*

better than the power-law networks after 1,000 queries (the first data point). Although 1,000 queries are only enough to provide rough estimates of peer popularity, even rough estimates are able to produce a more efficient topology than a power law network.

5.2 Prototype Measurements

We have implemented a prototype peer-to-peer middleware toolkit, called *Overlay-Dynamic Information Networks (ODIN)*, and we used it to test the square-root topology and *square-root-construct* algorithm with queries over real data. ODIN is implemented in C++, and communicates using XML messages over HTTP connections. Each peer connects to randomly chosen peers, whose addresses are gathered from a "host-catcher" at a well known address or from the headers of messages observed in the network. Our peers used the *square-root-construct* algorithm (with parameters from Table 3) to adapt the network topology as they processed searches. We compared this network to one constructed using a traditional (i.e. Gnutella) unstructured topology policy. In this policy, peers connected to random remote peers, always trying to keep at least five connections alive but without aiming for a particular topology.

For our experiment, we downloaded 169,902 HTML pages (4.04 GB total) from 1,000 web sites. We then started 1,000 peers on cluster machines in our lab, and each peer stored the content from one web site. Peers processed queries over the full text of web pages using standard techniques (the cosine distance and TF/IDF weights [19]). We generated 20,000 keyword queries from the downloaded data with query terms matching the distribution observed in several real user query sets [20]. Each query was submitted to a randomly chosen peer.

Figure 5 shows a running average (every 1,000 queries) of the total network bandwidth required per search. As the figure shows, the network using the square-root

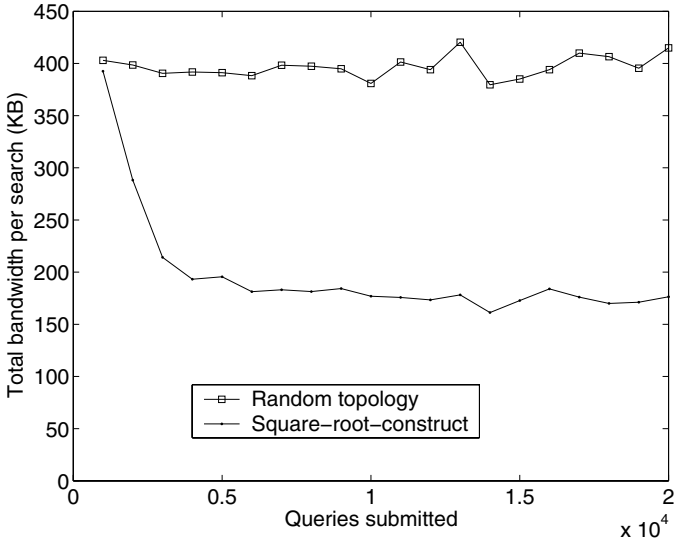

Fig. 5. Bandwidth required for search messages

construct algorithm initially performs poorly but then improves significantly, eventually requiring half the bandwidth on average of the network constructed randomly. Once each peer builds an accurate estimate of the popularity of its content, and adjusts its degree accordingly, the total bandwidth used drops below 180 KB per search, compared to 415 KB per search for the random topology.

In return for this higher efficiency, the *square-root-construct* network must send more control traffic (connect and disconnect messages) between peers. In fact, the square-root network requires 5.4 times as much bandwidth for control messages than in the random network. However, this cost is far outweighed by the savings in search bandwidth; an extra 4.6 KB per search on average for control messages results in a savings of 238 KB in search bandwidth per search on average. We can conclude that the extra control traffic is insignificant compared to the benefits of the square-root-topology.

6 Related Work

Random walk searches in peer-to-peer networks were proposed by Adamic et al [10] in order to cope with the unique characteristics of power-law networks. Follow-on work by others showed how to enhance performance by using replication [11,14], parallel random walks [11] and biased random walks of various types [13]. Most of this work assumes an existing topology, either power-law, random, or some other organization. In our results sections we examined each of these techniques. Other techniques have been proposed, such as "intelligent search" [21], routing indices [22], result caching [23] and so on. We have not yet tested the square-root topology against an exhaustive list of techniques, although we are continuing to gather data about its effectiveness for various techniques. Gkantsidis, Mihail and Saberi [24] discuss how to use random walks and flooding together to achieve high efficiency. Our square-root topology can be used together with their techniques to achieve even higher performance.

Some investigators have looked at building efficient topologies for peer-to-peer searches. Pandurangan et al [25] discuss building low diameter networks, although their focus is on Gnutella-style flooding for which low diameter is important. Lv et al [12] presented a dynamic algorithm for load balancing in peer-to-peer networks. Their goal is to shift load onto high capacity nodes. To achieve this load balancing, overloaded nodes must find nearby nodes to take over some of their connections. Our approach, while similarly using adaptivity, has a different goal of shifting load onto the most popular nodes. Moreover, our algorithm allows a peer to simply drop a connection without having to find a peer to take it over. While our approach can reduce overall load in the system, it does not achieve the load balancing that Lv et al's approach does. It may be possible to extend our techniques to take both popularity and capacity into account. Gia [4] is a system that combines several techniques, including topology adaptation and biasing random walks toward high-capacity nodes. Their goal is load balancing to improve efficiency. It may be possible to combine our techniques with theirs.

Several investigators have examined peer-to-peer systems analytically; examples include models for peer behavior [26], download traffic [27], data semantics [28], and

so on. Gkantsidis, Mihail and Saberi [29] demonstrate analytically that random walks are useful to locate popular content in two cases: a) when the topology forms a super-peer network, and b) when the same search is issued repeatedly. We expand on their work in several ways. First, our analysis holds for both popular and rare items; in fact, the square root topology is specifically optimized to provide efficient searching over a wide range of item popularities. Second, while their analysis and simulation is limited to pure random walks, we demonstrate that the square-root topology is efficient for a wide range of search techniques, such as biased random walks, random walks with proactive replication, and so on. Third, we show that the square-root topology is useful both in the case of super-peer networks and in flat networks.

Several investigators have proposed more structured peer-to-peer networks, sometimes known as distributed hash tables (DHTs). Examples include CHORD [1], CAN [2], Pastry [3], and others. In these systems, the topology is structured according to protocol rules in order to ensure high efficiency. Despite the advent of DHTs, research in and deployment of unstructured systems continues. One reason is the continuing popularity of unstructured systems such as Gnutella and Kazaa, and another reason is the difficulty experienced, at least until recently [5,30], with using DHTs for keyword search. Chawathe et al [4] discuss several reasons why both unstructured networks and DHTs are worthy of study. Loo et al [5,6] discuss a hybrid structured/unstructured architecture for information discovery, and our work could impact the design of the unstructured part of such a hybrid system.

In a previous workshop paper [31], we have examined a narrow application of the square root topology in situations where it is not feasible to replicate data or indexes. Here, we examine the usefulness of the square root topology for a wide range of searching techniques (including proactive replication, supernode networks, and other approaches to using replication).

7 Conclusions

We have presented the square-root topology, and shown that implementing a protocol that causes the network to converge to the square root topology, rather than a power-law topology, can provide significant performance improvements for peer-to-peer searches. In the square-root topology, the degree of each peer is proportional to the square root of the popularity of the content at the peer. Our analysis shows that the square-root topology is optimal in the number of hops required for simple random walk searches. We also present simulation results which demonstrate that the square-root topology is better than power-law topologies for other peer-to-peer search techniques. Next, we presented an algorithm for constructing the square-root topology using purely local information. Each peer estimates its ideal degree by tracking how many queries match its content, and then adds or drops connections to achieve its estimated ideal degree. Results from simulations and our prototype show that this locally adaptive algorithm quickly converges to a globally efficient square-root topology. Our results show that the combination of an optimized topology and efficient search mechanisms provides high performance in unstructured peer-to-peer networks.

References

1. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. In: Proc. SIGCOMM. (2001)
2. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A scalable content-addressable network. In: Proc. SIGCOMM. (2001)
3. Rowstron, A., Druschel, P.: 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)
4. Chawathe, Y., Ratnasamy, S., Breslau, L., Lanham, N., Shenker, S.: Making Gnutella-like P2P systems scalable. In: Proc. SIGCOMM. (2003)
5. Loo, B., Hellerstein, J., Huebsch, R., Shenker, S., Stoica, I.: Enhancing P2P file-sharing with an Internet-scale query processor. In: Proc. Conference on Very Large Data Bases. (2004)
6. Loo, B., Huebsch, R., Stoica, I., Hellerstein, J.: Enhancing P2P file-sharing with an Internet-scale query processor. In: Proc. International Workshop on Peer-to-Peer Systems. (2004)
7. Yang, B., Garcia-Molina, H.: Designing a super-peer network. In: Proc. ICDE. (2003)
8. Kalnis, P., Ng, W., Ooi, B., Papadias, D., Tan, K.: An adaptive peer-to-peer network for distributed caching of OLAP results. In: Proc. SIGMOD. (2002)
9. Agarwal, D., Berket, K.: Supporting dynamic ad hoc collaboration capabilities. In: Proceedings of the 2003 Conference for Computing in High-Energy and Nuclear Physics (CHEP 03). (2003)
10. Adamic, L., Lukose, R., Puniyani, A., Huberman, B.: Search in power-law networks. Phys. Rev. E **64** (2001) 46135–46143
11. Lv, Q., Cao, P., Cohen, E., Li, K., Shenker, S.: Search and replication in unstructured peer-to-peer networks. In: Proc. of ACM Int'l Conf. on Supercomputing (ICS'02). (2002)
12. Lv, Q., Ratnasamy, S., Shenker, S.: Can heterogeneity make Gnutella scalable? In: Proc. of the 1st Int'l Workshop on Peer to Peer Systems (IPTPS). (2002)
13. Yang, B., Garcia-Molina, H.: Improving search in peer-to-peer networks. In: Proc. ICDCS. (2002)
14. Cohen, E., Shenker, S.: Replication strategies in unstructured peer-to-peer networks. In: Proc. SIGCOMM. (2002)
15. Motwani, R., Raghavan, P.: Randomized Algorithms. Cambridge University Press, New York, NY (1995)
16. Cooper, B.F.: A content model for evaluating peer-to-peer searching techniques. In: Proc. ACM/IFIP/USENIX Middleware Conference. (2004)
17. Palmer, C., Steffan, J.: Generating network topologies that obey power laws. In: Proc. GLOBECOM. (2000)
18. Nejdl, W., Wolpers, M., Siberski, W., Schmitz, C., Schlosser, M., Brunkhorst, I., Loser, A.: Super-peer-based routing and clustering strategies for RDF-based peer-to-peer networks. In: Proc. WWW. (2003)
19. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. ACM Press, New York, N.Y. (1999)
20. Cahoon, B., McKinley, K.S., Lu, Z.: Evaluating the performance of distributed architectures for information retrieval using a variety of workloads. ACM Transactions on Information Systems **18** (2000) 1–43
21. Kalogeraki, V., Gunopulos, D., Zeinalipour-Yazti, D.: A local search mechanism for peer-to-peer networks. In: Proc. CIKM. (2002)
22. Crespo, A., Garcia-Molina, H.: Routing indices for peer-to-peer systems. In: Proc. ICDCS. (2002)

23. Bhattacharjee, B.: Efficient peer-to-peer searches using result-caching. In: Proc. IPTPS. (2003)
24. Gkantsidis, C., Mihail, M., Saberi, A.: Hybrid search schemes for unstructured peer-to-peer networks. In: Proc. INFOCOM. (2005)
25. Pandurangan, G., Raghavan, P., Upfal, E.: Building low-diameter P2P networks. In: Proc. IEEE Symp. on Foundations of Computer Science. (2001)
26. Ge, Z., Figueiredo, D., Jaiswal, S., Kurose, J., Towsley, D.: Modeling peer-peer file sharing systems. In: Proc. INFOCOM. (2003)
27. Gummadi, K., Dunn, R., Saroiu, S., Gribble, S., Levy, H., Zahorjan, J.: Measurement, modeling and analysis of a peer-to-peer file-sharing workload. In: Proc. SOSP. (2003)
28. Bernstein, P., et al: Data management for peer-to-peer computing: A vision. In: Proc. WebDB. (2002)
29. Gkantsidis, C., Mihail, M., Saberi, A.: Random walks in peer-to-peer networks. In: Proc. INFOCOM. (2004)
30. Reynolds, P., Vahdat, A.: Efficient peer-to-peer keyword searching. In: Proc. ACM/IFIP/ USENIX International Middleware Conference. (2003)
31. Cooper, B.F.: Quickly routing searches without having to move content. In: Proc. IPTPS. (2005)