Searching for Rare Objects using Index Replication

Krishna P. N. Puttaswamy, Alessandra Sala and Ben Y. Zhao
Computer Science Department, U. C. Santa Barbara
{*krishnap, ravenben*}@*cs.ucsb.edu sala@dia.unisa.it*

***Abstract*—Searching for objects is a fundamental problem for popular peer-to-peer file-sharing networks that contribute to much of the traffic on today's Internet. While existing protocols can effectively locate highly popular files, studies show that they fail to locate a significant portion of existing files in the network. High recall for these "rare" objects would drastically improve the user experience, and make these networks the ideal distribution infrastructure for user-generated content such as home videos and photo albums. In this paper, we examine simple techniques that can improve search recall for rare objects while minimizing the overhead incurred by participating peers. We propose several strategies for multi-hop index replication, and demonstrate their effectiveness and efficiency through both analysis and simulation. We further evaluate our simple techniques using detailed traces from a real Gnutella network, and show that they improve the performance of these overlays by orders of magnitude in both lookup success and overhead.**

I. Introduction

Searching for objects is a fundamental problem faced by unstructured peer-to-peer file-sharing networks. Various algorithms such as Flooding, Random Walks and their variants have been proposed to address this problem. However, most of these algorithms are only effective for locating popular objects, including algorithms used by popular deployed networks such as Gnutella and Kazaa. Studies have shown that in the widely-used Gnutella network, as much as 18% of all queries return no responses even when results are available [14]. Compared to similar operations in their structured counterparts, finding rare objects (those with few replicas) in unstructured networks is generally ineffective (in terms of search success) and inefficient (in terms of overhead and response time) [24].

Existing work has explored several approaches to improve search recall. One approach is to use higher Time To Live (TTL) for search algorithms. However, higher TTL values significantly increase overall bandwidth consumption and provide diminishing returns, particularly for coarse-granular search algorithms such as flooding [14]. An alternative is to utilize object replication strategies to improve search success [7], [16]. But these techniques require replicating entire objects, incurring significant overheads in both storage and network bandwidth, thereby limiting their applicability.

Because these unstructured file-sharing systems account for a major portion of traffic in today's Internet [4], [25], any technique to improve search recall for rare objects must be light-weight, effective, and easily deployable. Our focus in this paper is to answer the following question: *Can we develop simple techniques to improve the search effectiveness for rare items, and also reduce the bandwidth overhead incurred?*

We believe intelligent index replication can provide the solution to this question. Not only does proactive index replication incur much lower overhead compared with data replication, previous work has shown it to be effective at improving the scalability of unstructured networks [6], [11], [22]. Our work explores the use of multi-hop index replication, which can significantly improve the cover region or effective search space of these overlays, while incurring low overhead compared to alternatives such as data replication. We explore the effectiveness of two-hop index replication, and propose different variants that traverse the overhead-performance tradeoff. To quantify the impact and feasibility of our proposed techniques, we evaluate them in a full-system Gnutella simulation using real measurement traces.

This paper makes four main contributions. First in Section III, we introduce Supernode-Constrained Random Walk (SCRW) and several two-hop index replication techniques that work well together, improving search recall of unstructured overlays by more than two orders of magnitude. Second, we prove asymptotical bounds for their various performance and overhead metrics in Section IV. Third, we experimentally evaluate our proposals and show that they apply to both large (100K) power-law networks as well as state-of-the-art networks proposed in research. Finally, in Section V, we implement these techniques in a large (72K) measurement-driven simulation of a Gnutella network, and show that our techniques can significantly improve the performance of deployed networks. Our proposed techniques are simple, easy to deploy, and incur low overheads in storage and updates of data indices under network churn.

II. Background and Related Work

A. Searching in Unstructured Overlays

Search Algorithms Flooding, the earliest search algorithm for unstructured networks, has limited control over the total number of messages generated in the network, incurs very high search overhead, and produces significant number of duplicate messages. Recent studies [6], [11], [12] have shown random walks to be significantly more efficient than Flooding.

Index Replication One-hop index replication, where every node stores just the metadata of its data on all of its one-hop neighbors, is a valuable technique in scaling unstructured networks [6]. Index replication enables a random walker to cover a larger portion of the network in fewer hops, thus improving the search success. In terms of bandwidth consumed, index replication is not only much cheaper than full-object replication [7], [16], but also more practical. Object replication strategies require additional state maintenance to correctly implement in a decentralized environment.

Since existing index replication algorithms always replicate a node's index to all one-hop neighbors, naively extending this technique to multiple hops would incur heavy overheads. In this paper, we seek a thorough analysis to fully understand the various costs of two-hop replication.

Exploiting Heterogeneity It has been observed that nodes in the Internet are heterogeneous in node capacity, defined as the amount of resources at the node, e.g. bandwidth. Recent protocols have exploited this to improve the scalability of unstructured networks [5], [6], by biasing random walks towards high-capacity nodes. As a result, searching for rare items stored on low-capacity nodes suffer heavy overheads, since they are less likely to be reached by random walks. Even one-hop index replication does not help, since it does not guarantee that the indices of low-capacity nodes reach the high-capacity nodes.

Hybrid Networks Hybrid networks [14] improve the search for rare objects by using a combination of structured and unstructured networks. The key idea is to identify rare objects and insert them into a structured network to improve the chances of lookup while use an unstructured network for finding popular objects. While effective, this requires deploying and maintaining a new overlay network along with rare objects, making this approach very expensive.

B. Power-law networks

Faloutsos et al. [9] showed that the Internet follows power-law both at the router level and inter-domain level. It is also argued that Gnutella-like networks follow power-law [1].

Random graphs with a given degree sequence are well studied [15], [18], [19]. A Random graph on n vertices with power-law degree distribution shows the following properties: the fraction of vertices of degree $0, 1, 2...$ is asymptotically $\lambda_1, \lambda_2, ...$, where the λ's sum to 1. It is shown in [18] that if $Q = \sum_i i(i-2)\lambda_i > 0$ (and the maximum degree is not too large), then such random graphs have a giant component with probability tending to 1 as n goes to infinity. While if $Q < 0$ (and the maximum degree is not too large), then all components are small with probability tending to 1 as $n \to \infty$.

In these networks, for $2 < \gamma < 3.475$, where γ is the exponent of the power-law distribution, a Largest Connected Component (LCC) is shown to exist [2], and LCC contains the majority of the nodes of the original graph and most of the links. In [10] the authors show that this giant component of a power-law random graph matches several characteristics of real complex networks, and hence is a good candidate for generating synthetic Internet topologies.

III. System Design

Existing unstructured networks have one main problem: they are highly limited in their ability to locate rare items. The goal of our work is to develop techniques that improve the chances of finding rare objects in unstructured networks. To achieve our goal, we need to find rare objects with low overhead, otherwise the total load on the network becomes unbearable, potentially limiting the scalability of the network. We begin the description of our contributions by providing the reasons that carved our design.

A. Design Rationale

Two main techniques lie at the core of our design: Supernode-Constrained Random Walk (SCRW) and two-hop index replication. The main intuition behind them is to use the high-capacity nodes to build concentrated clusters of indices, then constraining the propagation of queries to these nodes. To ensure that queries terminate with high success rate, we need to efficiently place the indices on these clustered nodes. Our topology construction and search algorithms address the former while our two-hop replication achieves the latter.

1) Topology Construction and Search: Previous work showed that node degree in Internet topologies and unstructured overlay network topologies follow power-Law distributions, and hence contain significant heterogeneity. Exploiting this improves network scalability by orders of magnitude in Gia [6]. The idea is to assign work (through in-degree) to nodes proportional to their capacity and organize the network so all low-capacity nodes are closeby to high-capacity nodes. Because of their high in-degree, high-capacity nodes have large amount of information, and hence provide better responses to queries. In addition, these supernodes exhibit much low churn rates compared to other nodes [13], naturally forming a stable core of the network.

Our topology construction algorithm is motivated by these two observations. A careful inspection reveals that these two approaches are similar in that they use high capacity nodes as main paths for search and try to propagate the index from low capacity nodes to these main paths. Our algorithm uses Gia's topology adaptation algorithm while building the network topology and assigning in-degree to nodes based on their capacity. Along with this, it tags the nodes with high capacity as supernodes forming a stable path with large capacity.

2) Index Replication: One-hop replication has been shown to significantly improve scalability for unstructured networks. We extend this one-hop replication to two hops in this work. Note that extending index replication to two hops might lead to an explosion in the amount of index stored on high-capacity nodes. Thus managing the index replication and storage overheads becomes critical. We propose different replication strategies and study their performance in detail.

B. Two-hop Index Replication Strategies

We explore three two-hop index replication strategies.

Full replication. In this strategy, each node sends its index to all of its one-hop neighbors in its routing table, just like in one-hop replication. All of the one-hop neighbors, in turn, forward this index to all of their one-hop neighbors except the source node. This strategy effectively reduces to a two-hop flooding of indices around the nodes. We use SCRW with this replication strategy.

Square-root replication. In this strategy, each node performs one-hop replication. Supernodes then replicate the indices of their one hop neighbors to a random subset of their supernode neighbors. Each supernode's two-hop replica set size is equal to the square-root of the number of its supernodes neighbors. Thus, this is simple one-hop replication augmented with square-root replication only at the supernodes. The intuition is that by replicating on the supernodes, we are favoring SCRW to find objects quickly. At the same time, we are reducing the amount of replication and its cost.

Constant replication. Finally, we use a strategy where each supernodes does two-hop index replication to a constant number of supernode neighbors. After each node does one-hop index replication, supernodes propagate the index to only a constant number of their supernodes. This reduces indexing load on supernodes. We further reduce the load further by using normal random walk instead of SCRW with this strategy. Since each peer has the indices of its one-hop neighborhood, it can forward the query to a different neighborhood without having to go through a superpeer.

We analyze these strategies in the next section with a focus on the following three properties:

- Cover Time. We define cover time as the number of hops taken by a query to walk through all the supernodes in the network.
- Supernode Load. The average amount of load (in terms of the number of queries processed) on the supernodes is called the supernode load.
- Storage Load. The average amount of index stored on supernodes is the Storage Load. This also gives us an estimate of the amount of index transfer overhead incurred in the network.

C. Supernode-Constrained Random Walk

We use supernode-constrained random walk (SCRW) as our search algorithm. The main difference between SCRW and a normal random walk is that in SCRW nodes always forward the query to one of its randomly selected supernode neighbors – not just any random neighbor. If no supernodes are found in its routing table, then the node forwards the query to one of its neighbor selected at random.

Since random walks could lead to duplicate queries at a node, avoiding revisiting nodes is important. To do this, we embed a small history in each query to keep track of recently visited nodes. This history is a moving window is updated when a query reaches a new node. If the query reaches a node that has a degree of just one, then the query jumps back to the least recently visited node and continues the walk. Detailed ananlysis of this new random walk technique is, however, not the focus of this paper.

IV. Theoretical Analysis

In this section, we seek to better understand the performance characteristics of different index replication strategies. We compare several index replication strategies in terms of search performance, per-node query load, and index storage overhead.

Since prior work has shown that power-law properties hold on both the Internet and peer-to-peer topologies [6], [9], we will base our analysis on a power-law network context. In this model, a node's maximum connectivity is roughly proportional to its "capacity," which is modeled in practice using bandwidth capacity. Given the heterogeneity across node capacities, we divide all peers into those with a relatively low degree, called "standard peers," which are connected by a set of highly connected nodes we call superpeers.

Specifically, we will compare one-hop replication against three variants of two-hop replication. In one-hop index replication, each node maintains an index of objects stored by its one-hop neighbors. This significantly improves the average and maximum search size when the random walk is forced to go through superpeers. The cost for this performance improvement is that highly connected nodes incur overhead that scales linearly with the number of queries. We compare one-hop replication against three of our proposed two-hop replication strategies and show how we can navigate the overhead and performance tradeoff.

Formally, we can consider a network as a graph $G = \{V, E\}$, where V is a set of N nodes and E is a set of edges that connect two elements of V. Using the generating function formalism introduced by [21] for graphs with arbitrary degree distribution, we now analytically evaluate the performance of our strategies on power-law networks.

Let $G_0(x)$ be a generating function for the distribution of nodes degree k such that

$$G_0(x) = \sum_{k=0}^{k=\infty} p_k x^k$$

where p_k is the probability that a random chosen node has degree k. In our model we use a power-law distribution with exponent γ, so that, the above generating function is given by $G_0(x) = \sum_{k=0}^{k=k_{max}} ck^{-\gamma}x^k$, where c is a *normalization constant* that satisfies the following equation: $G_0(1) = \sum_{k=0}^{k=k_{max}} ck^{-\gamma} = 1$. Note that the maximum value of k has to be $k_{max} = N^{1/\gamma}$ and the exponent of our power-law distribution on the nodes degree is: $2 < \gamma < 3.475$, as shown in the experiments done in [2].

We characterize our network by a two-level hierarchical structure, where we have "superpeers" (*i.e.* nodes with high connectivity and high computational capability) on the top level, and all remaining "standard peers" on the bottom level. For generality, we cannot fix a single threshold value to define the minimum degree of a superpeer. We solve this problem with the following definition:

Definition 1. *Superpeers have a degree k such that $N^{1/\gamma-\delta} \leq k \leq N^{1/\gamma}$ with $\delta \in]0, 1/\gamma[$[1] and let S be the set of superpeers that we will call the network core.*

Applying one- and two-hop index replication changes our view of the network. Most nodes in a power-law network will be clearly grouped into clusters, where nodes in each

[1] Note that the notation $]0, 1/\gamma[$ means that the extremes are not included.

cluster share knowledge of stored objects in the cluster through replicated indices. We define a cluster as follows.

Definition 2. *Let C be a cluster of nodes such that:* $\forall x \mid x \in S$ *then* C_x $=\{v \mid v$ *has an edge that connects v and x* $\}$*. The number of clusters in the network is proportional to number of superpeers.*

For each superpeer $x \in S$ we define the following sets:

Definition 3. *Let* $V_1(x)$ *be a set of all distinct superpeers in one-hop distance from x, and let* $V_2(x)$ *be a set of all distinct superpeers in two-hop distance from x.*

Definition 4. *Let* $\alpha = (1/\gamma - \delta)(\gamma - 2)$ *and* $\alpha \in]0, 0.425[$[2]. *Note that* α *is used only as a way to simplify notation here.*

As described earlier, we use the following three performance metrics to compare different index replication strategies: *cover time* on the network core, *query load* on superpeers and the *size of index caches* at nodes.

A. One-hop Index Replication

Our theoretical analysis aims to bound the time to find objects in the network. We will focus on the case of an uncommon object: an object with a constant number of replicas independent of the network size. We will show that a query will locate an object with high probability if it covers the network core.

Theorem 1. *Let* δ*,* γ *and* α *be three parameters that characterize our system, and in particular* $\delta \in]0, 1/\gamma[$*,* $\gamma \in]2, 3.475[$ *and* $\alpha \in]0, 0.425[$*, then the number of superpeers in our system is* $\Omega(N^{\frac{\gamma-1}{\gamma}-\alpha+\delta})$.

Proof: Let d be the degree of a generic node in the system, then the probability that this node is a superpeer is conditioned on the fact that d has to be more then $N^{1/\gamma-\delta}$. So, the probability that a node x with degree d is a superpeer is $P[d > N^{1/\gamma-\delta}]$. Thus, $P[d > N^{1/\gamma-\delta}] = \sum_{k=N^{1/\gamma-\delta}}^{N^{1/\gamma}} k^{-\gamma}$ For all decreasing function like $k^{-\gamma}$, it is possible to bound it as following:

$$\sum_{k=N^{1/\gamma-\delta}}^{N^{1/\gamma}} k^{-\gamma} > \int_{N^{1/\gamma-\delta}}^{N^{1/\gamma}} k^{-\gamma} d(k)$$

$$= \frac{N^{\frac{-\gamma+1}{\gamma}}}{-\gamma+1} - \frac{N^{(1/\gamma-\delta)(-\gamma+1)}}{-\gamma+1} > \frac{N^{(\gamma-1)(\delta-1/\gamma)}}{2(\gamma-1)}$$

so the probability that x is a superpeer is, $\Omega(N^{-(1/\gamma-\delta)(\gamma-1)})$.

Let $X_1, X_2, ..., X_N$ be N random variables such that:
$X_i = \begin{cases} 1, & \text{if i is a superpeer;} \\ 0, & \text{otherwise.} \end{cases}$
Let p_i be the probability that $X_i = 1$ and let $\mu = E[X] = \sum_{i=1}^{N} p_i = \sum_{i=1}^{N} N^{-(1/\gamma-\delta)(\gamma-1)}$. By the Chernoff bound [20] we have $P(X < (1-\tau)\mu) < e^{\frac{-\mu(\tau)^2}{2}}$. Substituting μ with the values of our system and let τ be $\frac{1}{2}$, we obtain the following:

$$P[X < \frac{1}{2} N^{1-(\gamma-1)(1/\gamma-\delta)}] < e^{-\frac{N^{1-(\gamma-1)(1/\gamma-\delta)}}{8}} < 1/N.$$

For each choice of γ in its definition interval there is a set of δ values that satisfy the inequality. The proof shows that the number of superpeers in the network is $\Omega(N^{1-(\gamma-1)(1/\gamma-\delta)}) = \Omega(N^{\frac{\gamma-1}{\gamma}-\alpha+\delta})$. ■

To cover the core of the network we first have to reach one superpeer and then go through each node in the core. Indeed, we want to reach a superpeer using a minimal number of routing hops. We use a random walk to perform routing so the problem is: starting from a randomly chosen node, how many random walk steps must we take to reach a superpeer?

Theorem 2. *The number of routing steps to reach a superpeer using random walk is* $O(N^{\alpha})$.

We omit the details of the proof for brevity. The full version of the proofs can be found in our technical report [23].

We need to estimate how many superpeers are in V_1, that will tell us the number of superpeers who are one hop away from each other. If we are able to reach a superpeer, then by going through each superpeer in the core we will be able to scan a large fraction of peers. This gives us a high probability of finding the desired object.

Theorem 3. *Let x be a superpeer, then* $|V_1(x)|$ *is* $O(N^{\frac{\gamma-1}{\gamma}-\alpha})$ *with probability* $\Omega(N^{-\alpha})$ *and* $|V_1(x)|$ *is* $\Omega(N^{\frac{\gamma-1}{\gamma}-2\alpha})$ *with high probability.*

Proof: Sketch of the proof[3]. The probability that a superpeer is connected to "s" different other superpeers, is strongly bound to the number of links that end in superpeers. That is: $\sum_{i=1}^{s} \frac{N^{\frac{\gamma-1}{\gamma}-\alpha+\delta} N^{1/\gamma-\delta} - iN^{1/\gamma}}{N} \geq \frac{N^{-\alpha}}{2}$ for each $s \leq \frac{N^{\frac{\gamma-1}{\gamma}-\alpha}}{2}$. Using the Chernoff bound [20] we show that the lower bound on the number of different superpeers connected to another superpeer is $\Omega(N^{\frac{\gamma-1}{\gamma}-2\alpha})$, with high probability. ■

Theorem 3 shows the degree of connectivity of nodes in the network core. We use this to prove the next result.

Theorem 4. *The time to cover the superpeers in the network is* $O(N^{\frac{\gamma-1}{\gamma}-\alpha+\delta} \log N)$.

Proof: We consider the core nodes (*i.e.* the superpeers in the network) as a subgraph. On this subgraph we have to bound the number of steps to cover all nodes. We use the degree of connection between nodes from theorem 3 to define the network's core as a d-random regular graph. Indeed, by construction each superpeer chooses only d neighbor superpeers that considers as real superpeer (i.e. SCRW is routed using one random choice among these d neighbor superpeer). In our case, the value of d is $N^{\frac{\gamma-1}{\gamma}-2\alpha}$ and the number of nodes to cover are $N^{\frac{\gamma-1}{\gamma}-\alpha+\delta}$. Let n be $N^{\frac{\gamma-1}{\gamma}-\alpha+\delta}$. Using the argument in [8], we find that the cover time in our system is asymptotic to $\frac{d-1}{d-2} n \log n$, with high probability. Therefore,

[2]Note that the α interval is derived from γ's interval and definition 1.

[3]A full version of all proofs is included in our technical report [23].

$\frac{d-1}{d-2} n \log n < \frac{d}{\frac{d}{2}} n \log n = 2n \log n$. Substituting n with our value we have:

$$N^{\frac{\gamma-1}{\gamma}-\alpha+\delta} \log N^{\frac{\gamma-1}{\gamma}-\alpha+\delta} < cN^{\frac{\gamma-1}{\gamma}-\alpha+\delta} \log N$$

where c is a constant. We have to add the time to reach a superpeer and so the total time is $cN^{\frac{\gamma-1}{\gamma}-\alpha+\delta} \log N + N^{\alpha}$ that is still $O(N^{\frac{\gamma-1}{\gamma}-\alpha+\delta} \log N)$, which is sub-linear. ■

It is easy to show that covering the core superpeers is enough to find an uncommon item, because indices at superpeers cover the large majority of the network via one-hop index replication. The main problem is that in the worst case, a query for an uncommon item must cross all superpeers in the network and lead to high query load on superpeers.

Fact 1. *The load is proportional to the number of queries for uncommon item, because for each of these queries the query must traverse through superpeers until the item is found.*

Fact 2. *The cache size of a superpeer is proportional to the superpeer's degree, which is $O(N^{1/\gamma})$.*

B. Two-hop Index Replication

We analyze three variants of two-hop index replication: *Full two-hop*, *Constant two-hop* and *Square root two-hop*. We compare these index replication strategies on search performance, per-node query load, and index storage overhead.

Full Two-hop Index Replication

The first approach is the full two-hop index replication. In this strategy each node maintains an index of objects stored by all neighbors within two hop. We use Theorem 5 to support our result on the cover time for this strategy.

Theorem 5. *Let x be a superpeer, then $|V_2(x)|$ is $O(N^{\frac{\gamma-1}{\gamma}-2\alpha})$ with probability $\Omega(N^{-2\alpha})$ and $|V_2(x)|$ is $\Omega(N^{\frac{\gamma-1}{\gamma}-4\alpha})$ with high probability.*

The proof is omitted for brevity since it follows a similar formulation done in theorem 3. Additional details are in [23].

Full two-hop index replication guarantees that each superpeer does not need to visit its neighbor superpeers because it already knows all their neighbors. We are interested to prove how many hops are needed to cover the network's core leveraging the assumption that each superpeer know objects stored by its two-hop neighbors.

Theorem 6. *The cover time on the network's core, by full index replication, is $O(N^{\delta+\alpha} \log N)$, with high probability.*

Proof: As we argued in theorem 3, in each cluster C_x the number of superpeers is $d = \Omega(N^{\frac{\gamma-1}{\gamma}-2\alpha})$. Each superpeer selects, by construction, exactly d superpeers in order to build the network core as a d-random regular graph and keep the random walks, of the routing algorithm, inside the network core. Moreover, each superpeer x receives indices from all clusters rooted in each of the superpeer in C_x. Since, $\Omega(N^{\frac{\gamma-1}{\gamma}-2\alpha})$ superpeers are into each cluster then with :

$$\frac{N^{\frac{\gamma-1}{\gamma}-\alpha+\delta}}{N^{\frac{\gamma-1}{\gamma}-2\alpha}} = N^{\delta+\alpha}$$

superpeers we could know all superpeers in the network. Using theorem 5, we can consider the network's core as a set of different $N^{\delta+\alpha}$ sets with degree $\Omega(N^{\frac{\gamma-1}{\gamma}-4\alpha})$ among them. If $\bigcap_{i=1}^{N^{\delta+\alpha}} C_i = \phi$ then the cover time is $O(N^{\delta+\alpha} \log N)$, using the assumption in [8]. The point is that we cannot guarantee that the previous intersection is empty. So we must prove that going into $O(N^{\delta+\alpha} \log N)$ clusters, we will check all superpeers at least once with high probability. We can formulate our problem as a random selection with replacement problem: let $N^{\frac{\gamma-1}{\gamma}-\alpha+\delta}$ be the number of balls, for each draw we select $N^{\frac{\gamma-1}{\gamma}-2\alpha}$ balls. We need to prove that after $N^{\delta+\alpha} \log N$ draws each ball has been selected at least one time. For each draw the probability that one ball is not chosen is $(1 - \frac{1}{N^{\frac{\gamma-1}{\gamma}-\alpha+\delta}})^{N^{\frac{\gamma-1}{\gamma}-2\alpha}}$. After $N^{\delta+\alpha} \log N$ draws the probability that comes out is:

$$(1 - \frac{1}{N^{\frac{\gamma-1}{\gamma}-\alpha+\delta}})^{N^{\frac{\gamma-1}{\gamma}-\alpha+\delta} \log N} = \frac{1}{e^{\log N}} < \frac{1}{e^{\log_e N}} = \frac{1}{N}$$

Therefore the total time is gives from $N^{\delta+\alpha} \log N$ steps to cover the network's core plus N^{α} steps to reach the first superpeer that is $O(N^{\delta+\alpha} \log N)$. ■

We now analyze the query load on the superpeers. In our system we know that for an uncommon item, the way to find it is to visit all superpeers in the network.

Theorem 7. *The average load on the superpeers is proportional to $\frac{\#\ query \log N}{N^{\frac{\gamma-1}{\gamma}-2\alpha}}$.*

Proof: As shown in theorem 6, each query crosses $N^{\delta+\alpha} \log N$ superpeers in the network such that the overhead in the system is proportional to the crossed nodes . Therefore, the load on each superpeer is sub-linear on the number of query for uncommon items, and is $\frac{\#\ query \log N}{N^{\frac{\gamma-1}{\gamma}-2\alpha}}$. ■

The size of indices cached at each superpeer is proportional to the number of superpeers in each cluster.

Theorem 8. *The cache size of superpeers is $O(N^{1-\alpha})$.*

Proof: Let x be a superpeer, the maximum number of the peers in C_x is $N^{1/\gamma}$, and at most $N^{\frac{\gamma-1}{\gamma}-\alpha}$ are superpeers as showed in theorem 3,hence the superpeer x has to store:

- for each of its neighbor superpeers all their neighbors that are at most $N^{1/\gamma} N^{\frac{\gamma-1}{\gamma}-\alpha} = N^{1-\alpha}$;
- for all standard peers in its cluster C_x a constant number of their neighbors. Its neighbor standard peers are less then $N^{1/\gamma}$ and so the amount of information is $< k' N^{1/\gamma}$.

Adding this together, the amount of information that a superpeer x has to store is: $k' N^{1/\gamma} + N^{1-\alpha} = O(N^{1-\alpha})$ ■

Constant Two-hop Index Replication

The main problem of full two-hop index replication is that caches at superpeers could become too big. Here the idea is to reduce the index replication at superpeer. While we guarantee the cover time is similar to one-hop replication, we reduce cache size compared to two-hop strategy and query load on superpeers compared to both one and two-hop index

	Cover Time		Load $\propto$		Cache size for Superpeers	
		$N's\ Exponent$		$N's\ Exponent$		$N's\ Exponent$
One-hop index repl.	$O(N^{\frac{\gamma-1}{\gamma}-\alpha+\delta}\log N)$	< 1	$\#queries$		$O(N^{1/\gamma})$	< 0.5
Two-hop full	$O(N^{\delta+\alpha}\log N)$	< 0.5	$\frac{\#\ query\log N}{N^{\frac{\gamma-1}{\gamma}-2\alpha}}$	< 0.7	$O(N^{1-\alpha})$	< 1
Two-hop constant	$O(N^{\frac{\gamma-1}{\gamma}-\alpha+\delta}\log N)$	< 1	$\frac{\#\ query}{N}$	≤ 1	$O(N^{1/\gamma})$	< 0.5
Two-hop square root	$O(N^{\frac{\gamma-1}{2\gamma}+\delta}\log N)$	< 0.75	$\frac{\#\ queries\log N}{N^{\frac{\gamma-1}{2\gamma}-\alpha}}$	< 0.35	$O(N^{\frac{2\gamma-1}{2\gamma}-\alpha/2})$	< 0.75

TABLE I
COMPARING THE KEY PERFORMANCE AND OVERHEAD METRICS FOR ALL FOUR INDEX REPLICATION STRATEGIES.

replication. The idea is to implement the routing going through each cluster using standard peers.

Lemma 1. *Each normal peer that lives in a cluster can reach another cluster in constant hops, with high (constant) probability.*

This lemma proves that each standard peer that lives in a cluster has the possibility to reach another. This is done to show the capability that queries can go through clusters using standard peer, and yet cover the same amount of information as that of walking through superpeers with one hop replication. Therefore, constant replication obtains an optimal load balancing. It is possible because the standard peers that live in a cluster have the complete knowledge of all other peer's index in their cluster.

Theorem 9. *The cover time, in constant two-hop replication, is* $O(N^{\frac{\gamma-1}{\gamma}-\alpha+\delta}\log N)$.

The proof is similar to that of Theorem 4.

In the constant two-hop strategy, the routing hops required is greater than full two-hop. Compared to one-hop replication, it significantly reduces query load at the superpeers, and cache size at superpeers is asymptotically the same.

Theorem 10. *The average load on each superpeer x is less than* $\frac{\#\ query}{N}$.

The proof is omitted for brevity. The intuition is that instead of biased random walks on superpeers, we use random walks through standard peers to cover at least one peer per cluster.

The cache size of superpeers is drastically reduced because only a constant number of neighbor superpeers are considered during the index storage phase, and all others are treated as standard peers.

Theorem 11. *The cache size of superpeers, for the constant two-hop strategy, is* $O(N^{1/\gamma})$.

The proof follows the same technique as Theorem 8.

Square Root Index Replication

Our analysis shows that compared to one-hop index replication, full two-hop index replication improves both cover time on the network core and query load on each superpeer. But the cache size for each superpeer is significantly larger. Constant two-hop index replication shows a smaller cache size per superpeer than full two-hop, but similar cover time to one hop replication. Now we want to prove the properties of Square root two-hop index replication.

Theorem 12. *The cover time using Square root index replication, is* $O(N^{\frac{\gamma-1}{2\gamma}+\delta}\log N)$*, with high probability.*

We omit the proof for brevity, since it follows the same outline shown in theorem 6. With the same argument as in theorem 7, we can assert that the average load on the superpeers, in Square root index replication, is proportional to $\frac{\#\ queries\log N}{N^{\frac{\gamma-1}{2\gamma}-\alpha}}$.

Theorem 13. *The cache size for each superpeers is* $O(N^{\frac{2\gamma-1}{2\gamma}-\frac{\alpha}{2}})$.

The proof is similar to theorem 8 and is omitted for brevity.

Finally, we summarize all of our analytical results in Table I. We want to emphasize the substantial difference between our three two-hop replication strategies and one-hop replication. Two-hop constant strategy maintains the same cover time compared to one-hop replication, since only the non-superpeers nodes do two-hop replication. While two-hop constant reduces load for superpeers, it does not improve the cover time. We prove that the best cover time can be reached using two-hop full replication. Since all nodes do two-hop flooding, the concentration of indices at superpeers makes the SCRW very efficient, despite the larger cache loads for superpeers. Finally, square-root two-hop replication obtains the best tradeoff between cover time and cache size.

V. EXPERIMENTAL EVALUATION

A. Experimental Setup

We present our experimental results on OverSim [3], an event-based overlay network simulator. For our simulations, we modified an implementation of Gia [6] included with OverSim to use the same parameters as the Gia paper. In addition to Gia, we implemented a simple unstructured overlay protocol to experiment with different search and replication algorithms. We also added support in OverSim to import different network topologies to build overlay networks.

We used the bandwidth measurement traces from the Gnutella study [26] to derive the node capacities for Gia, allowed the network to stabilized, and generate a 100K node network topology. We used this Gia topology for all our experiments, and refer to it as Gia in our graphs. In addition to this topology, we also evaluated our algorithms using power-law network topologies generated by BRITE [17].

We simulated our 100K large overlay network on a quad-core Dell server with 16GB memory and four 2.3GHz CPUs. The large simulation size was necessary to provide realistic and representative results. In our Gia tests, we initialize the overlay on the pre-built topology, store a constant number of randomly selected objects on each node, and then issue queries from each node for random objects in the network. Our object assignment algorithm ensures that each object in the network has 3 copies distributed among random nodes in the network. In addition, we fix the number of supernodes in the network at approximately around 3%, following the measurement results from [26]. We use just one copy of the random walk for all search experiments (*i.e.* 1-SCRW) and terminate the walk when a copy is found or when the TTL expires. Each experiment submitted 100K queries into the network and evaluates different aspects of search performance.

B. Performance Metrics

We use the following metrics to study the performance of our proposed algorithms.

- Lookup Success. This metric describes the effectiveness of a search strategy in locating objects. This is expressed as a percentage of total search queries that return successfully.
- Lookup Overhead. The number of hops taken by a search query before it terminates is defined as the lookup overhead of that query. The overhead is always less than or equal to the maximum random walk depth. We use the average of all lookup query overheads in a test to quantify the search cost.
- Query Load. We measure the number of queries processed by each node in the network during a test run to understand query load per node. In addition to the load on all nodes, while using SCRW, we also quantify the load on each supernodes.
- Index Storage Cost. We use this to measure the per-node storage overhead from two-hop replication. This also quantifies the amount of index data transferred in the network during network churn.

C. Simulation Results

We present the simulation results in three parts. We start with an evaluation of search effectiveness and efficiency, then evaluate our system under a full-system simulation of the Gnutella network, and finally present the index replication overhead incurred because of churn. We ran all tests on both Gia and Brite overlays, and present only the Gia results when results from both topologies are similar.

1) Search Effectiveness and Efficiency: Figures 1 and 2 compare the lookup success and lookup overhead of two-hop replication strategies using SCRW with that of one-hop replication using standard random walk. We see that full two-hop replication with SCRW provides the highest lookup and the lowest overhead, while one-hop replication with RW provides the lowest lookup with highest overhead. Sqrt index replication's performance is very close to that of full two-hop, while the constant replication is better than one-hop replication but much worse than sqrt and full two-hop replication. Recall that these results are for all objects with only 3 replicas. We can expect higher search success in practice by increasing the replication factor. While full two-hop replication is the clear winner thus far, we need to fully understand its other effects before choosing a variant for practical deployment.

Figures 3 and 4 show the query load experienced by nodes in the network. Figure 3 shows the load distribution on all nodes for different combination of search and replication algorithms. We see that with their use of SCRW, Sqrt and Full have uneven load distribution, with most search queries processed by the supernodes. Both the normal RW one-hop (inherently biased towards high-degree nodes) and two-hop constant strategies spread the load more evenly in the network. Explicitly biasing the RW towards high degree nodes, like in Gia, takes the distribution away from the normal RW, towards SCRW. Figure 4 shows the average query load just on the supernodes. Sqrt experiences slightly higher load than Full strategy. Two-hop constant replication, however, incurs load comparable to that of just one-hop replication. Studying just the query load indicates that two-hop constant replication is desirable for distributing load evenly in the network, while full and sqrt replication strategies are good for two-tier architectures where supernodes have significant resources.

Figure 5 presents the total index storage (one-hop + two-hop) overhead on all the nodes in the network for different replication strategies. We see that the storage overhead (and hence the index transfer overhead) in Full and Constant replication is nearly ten times more than that of one-hop replication. Sqrt replication incurs the same overhead as that of one-hop on all nodes except supernodes which incur almost the same load as that of Constant replication (this explains the spike in the Sqrt overhead). Figure 6 presents the same results for a BRITE topology and we see that Sqrt overhead is the lowest. The main reason for the variations in these graphs is that in the Gia topology, superpeer degrees deviate from the power-law distribution, and is less connected than their counterparts in BRITE.

We see from these graphs that Full replication provides high lookup success, but also incurs high query load and index overhead. Constant replication is desirable for spreading the index transfer cost across the network or for even balancing of query load, but has poor lookup success and high lookup overhead. Sqrt replication provides a mix of the good properties of Full and Constant strategies. Its high lookup success with low lookup overhead while the total index overhead is only slightly higher than that of one-hop replication. We argue that Sqrt is the best variant for practical deployment because the supernodes chosen in deployed networks have high bandwidth and processing capacity, and index transfers can be amortized across time. Additionally, the significant savings in the bandwidth spent on processing queries should more than offset the cost of index replication. Also note that, as shown in Table I, Sqrt replication's cover time is sub-linear on the network size, indicating as network size grows, the

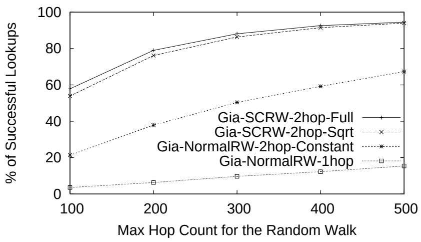

Fig. 1. The lookup success achieved by different replication strategies in a 100K Gia network.

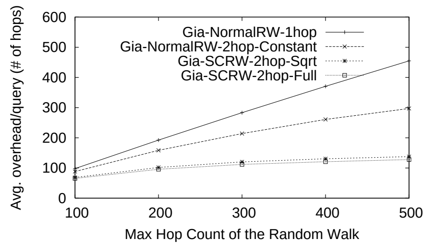

Fig. 2. Average number of hops to find an object, in a 100K Gia network, for different replication strategies.

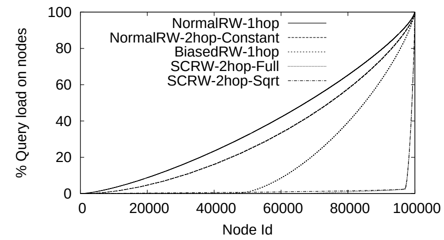

Fig. 3. CDF of query load for different replication strategies and random walks in a 100K Gia network. Plots ordered as legend.

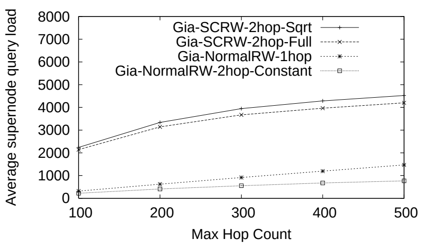

Fig. 4. Average number of queries processed by Supernodes, in a 100K Gia network, for different replication strategies.

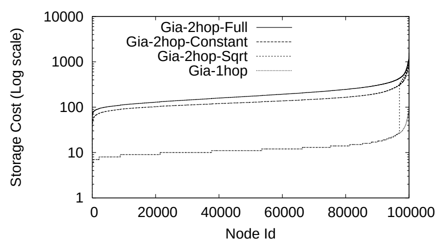

Fig. 5. Comparison of index storage overhead (sorted values) incurred by nodes, in a 100K Gia network, for different replication strategies. Plots ordered in the order of legends.

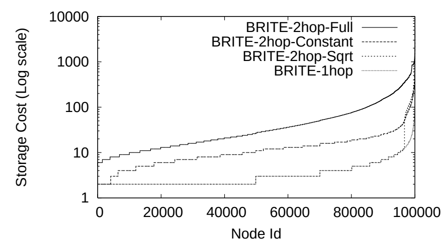

Fig. 6. Comparison of index storage overhead (sorted values) incurred by nodes for different replication strategies in a 100K BRITE network. Plots ordered in the order of legends.

lookup overhead continues to improve relative to Constant and One-hop replication strategies. From here on, we will use Sqrt two-hop as our chosen index replication technique.

D. Gnutella Full-System Simulation

To understand the performance of various search and replication algorithms in a real unstructured network, we simulated a complete Gnutella-like network. To make our simulation realistic, we obtained the network topology, the files stored in the nodes, the number of files stored, and the file distribution from Gnutella measurement study traces [27]. We extracted a Gnutella network topology with approximately 72K superpeers and 760K leafpeers from one of these topology traces. Since the leafpeers do not participate in query forwarding in Gnutella, we considered only the superpeers in our network topology. Then we extracted the list of files stored on 72K random nodes in the Gnutella trace and assigned them to the nodes in our topology. Researchers have empirically shown that this randomized placement of files on nodes approximately represents the real network, since there is very little correlation between file locations and the network topology [27]. There were approximately 27 million files, in total, with 7 million unique files assigned to nodes in our network. Since we are using the real traces to build our network, the object popularity in the network follows the same popularity we see in a real network.

To evaluate the search recall for rare objects, we preprocessed the files on the nodes in our network and queried for only the files with exactly 3 replicas in the network (approx. 300K of them). Furthermore, to make a fair comparison across the different search techniques and to understand the overall performance of the network, we explicitly limited the bandwidth that can be consumed by our tests. We limited the number of messages that can be forwarded in the entire network. Once this overhead cap is reached, all search messages are dropped wherever they are, and no additional messages or search queries are processed.

We examine the performance of three different search algorithms in this setup, SCRW with Sqrt two-hop index replication, Gia's biased randomwalk, and a simple flooding algorithm. Unlike early versions of Gnutella, our flooding only occurs between superpeer nodes. We experimented with different flooding depths and found that a flooding depth of 3 provides the best performance in our experiment setup. We compare this against a random walk depth of 500 for the Gia and SCRW. Note that random walk with depth of 500 incurs significantly less overhead than flooding with TTL of 3, which on average reaches 42K nodes.

Figures 7 and 8 show the performance of the three algorithms for same overhead. Figure 7 shows the absolute lookup success rates while the Figure 8 shows the relative improvement in search success. We see clearly that SCRW with Sqrt replication is more than 600 times better than simple flooding, and nearly 200 times better than normal RW with just one-hop replication. The improvement decreases at higher TTL values mainly because of the diminishing returns in lookup success experienced by SCRW.

E. Churn Measurements

Finally, we want to quantify the bandwidth costs of pushing index updates across the network following changes in network membership. Using our full-system Gnutella simulation,

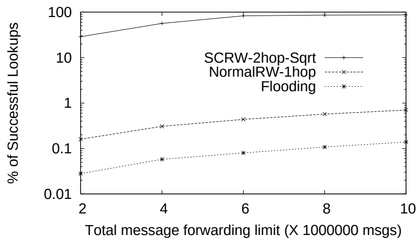

Fig. 7. Lookup success of under a bandwidth cap. Note the log scale on the y-axis.

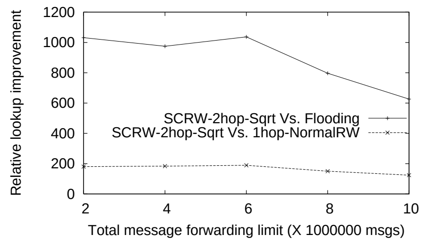

Fig. 8. Ratio of lookup success of different strategies.

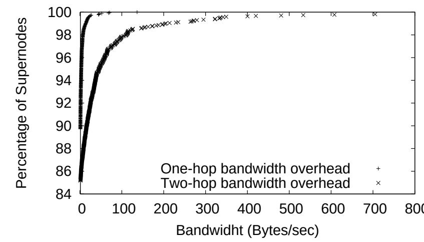

Fig. 9. CDF of index transfer overhead per supernode per second under Skype trace.

we evaluated the effect of churn on Sqrt replication index transfer overhead. Since Sqrt index replication mainly involves supernodes, we study only the effect of supernode churn. In order to do that in a practical manner, we use the measurement results from a recent Skype study which measured the churn characteristics of Skype supernodes. We assign lifetime to supernodes from this measurement data [13], stabilize the network, then perform two-hop Sqrt replication to reach a stable state. We then run the Skype trace, and measure the bandwidth required by nodes in index replication follow each supernode join or leave event. We assume that each data object entry including all metadata fields is 100 bytes.

Figure 9 presents the CDF of the index replication overhead incurred by the supernodes in the Gnutella topology. In our 72K topology we had approximately 2700 highly-connected supernodes. In a 5 day run of our churn experiments, nearly 26% of the nodes died. The overhead CDF shows that the Sqrt two-hop replication overhead is very low, only a few nodes have overhead above 500 Bytes/sec and the average overhead is approximately 10 Bytes/sec per supernode. This shows that deploying Sqrt index replication will not incur a significant bandwidth cost for index replication updates.

VI. Conclusions

While unstructured file-sharing networks have been successful at delivering popular content to its users, they are limited by their low search recall of rare objects. We explore the effectiveness of multi-hop index replication, which is easily-deployable and lightweight in overhead. We derive analytical results that quantify the search time and overheads for several variant protocols, and choose the best performance and overhead tradeoff. We evaluate our approach on both extremely large simulation networks (100,000 peers) and moderately-sized (72,000 peers) topologies from Gnutella measurements. Using the same bandwidth as flooding, our technique improves lookup of rare objects from less than 0.1% to more than 80%.

Acknowledgements

We thank Prof. R. Rejaie's group for their Gnutella traces. This work is supported by the DARPA Control Plane program (BAA04-11) and NSF CAREER Award #0546216.

References

[1] Adamic, L., Lukose, R., Puniyani, A., , and Huberman, B. Search in power law networks. *Phy Rev E, 64* (2001).

[2] Aiello, W., Chung, F., and Lu, L. A random graph model for massive graphs. In *Proc. of STOC* (2000), pp. 171–180.

[3] Baumgart, I., Heep, B., and Krause, S. Oversim: A flexible overlay network simulation framework. In *Proc. of IEEE Global Internet* (May 2007).

[4] CacheLogic research: Peer-to-peer in 2005. http://www.cachelogic.com/home/pages/research/p2p2005.php.

[5] Castro, M., Costa, M., and Rowstron, A. Debunking some myths about structured and unstructured overlays. In *Proc. of NSDI* (2005).

[6] Chawathe, Y., et al. Making gnutella-like p2p systems scalable. In *Proc. of SIGCOMM* (August 2003).

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

[8] Cooper, C., and Frieze, A. The cover time of random regular graphs. *SIAM J. Discrete Mathematics* (2004).

[9] Faloutsos, M., Faloutsos, P., and Faloutsos, C. On power-law relationships of the internet topology. In *Proc. of SIGCOMM* (1999).

[10] Gkantsidis, C., Mihail, M., and Saberi, A. Conductance and congestion in power law graphs. In *Proc. of Sigmetrics* (2003), ACM.

[11] Gkantsidis, C., Mihail, M., and Saberi, A. Random walks in peer-to-peer networks. In *Proc. of INFOCOM* (2004).

[12] Gkantsidis, C., Mihail, M., and Saberi, A. Hybrid search schemes for unstructured peer-to-peer networks. In *Proc. of INFOCOM* (Miami, FL, March 2005).

[13] Guha, S., Daswani, N., and Jain, R. An experimental study of the skype peer-to-peer voip system. In *Proc. of IPTPS* (February 2004).

[14] Loo, B. T., et al. Enhancing p2p file-sharing with an internet-scale query processor. In *Proc. of VLDB* (2004).

[15] Luczak, T. *Sparse random graphs with a given degree sequence*, vol. 2. Wiley, New York, 1992.

[16] Lv, Q., et al. Search and replication in unstructured peer-to-peer networks. In *Proc. of Supercomputing* (June 2002).

[17] Medina, A., et al. Brite: An approach to universal topology generation. In *Proc. of MASCOTS* (August 2001).

[18] Molloy, M., and Reed, B. *A critical point for random graphs with a given degree sequence*, vol. 6. 1995.

[19] Molloy, M., and Reed, B. *The size of the giant component of a random graph with a given degree sequence*, vol. 7. 1998.

[20] Motwani, R., and Raghavan, P. *Randomized Algorithms*. Cambridge International Series on Parallel Computation, 1995.

[21] Newman, M. E. J., Strogatz, S. H., and Watts, D. J. Random graphs with arbitrary degree distributions and their applications. *Phys. Rev. E 64* (2001).

[22] Puttaswamy, K., and Zhao, B. A case for unstructured distributed hash table. In *Proc. of IEEE Global Internet* (May 2007).

[23] Puttaswamy, K. P. N., Sala, A., and Zhao, B. Y. Searching for rare objects using index replication. Tech. Rep. 2007-12, UC Santa Barbara, January 2008.

[24] Qiao, Y., and Bustamante, F. E. Structured and unstructured overlays under the microscope. In *USENIX* (2006).

[25] Saroiu, S., et al. An analysis of internet content delivery systems. In *Proc. of OSDI* (December 2002).

[26] Saroiu, S., Gummadi, P. K., and Gribble, S. A measurement study of peer-to-peer file sharing systems. In *Proc. of MMCN* (January 2002).

[27] Zhao, S., Stutzbach, D., and Rejaie, R. Characterizing files in the modern gnutella network: A measurement study. In *Proc. of MMCN* (January 2006).