Optimizing Capacity-Heterogeneous Unstructured P2P Networks for Random-Walk Traffic

Chandan Reddy, Derek Leonard, and Dmitri Loguinov
Department of Computer Science and Engineering, Texas A&M University
College Station, TX 77843 USA
{chandanr, dleonard, dmitri}@cs.tamu.edu

Abstract

Existing algorithms for utilizing high-capacity nodes in heterogeneous P2P systems (e.g., [4], [13], [20]) often require unrealistically large node degree and high maintenance overhead in P2P networks with highly diverse node capacities and high churn. In this paper, we propose an unstructured P2P system that addresses these issues. We first prove that the overall throughput of search queries in a heterogeneous network is maximized if and only if traffic load through each node is proportional to its capacity. We then propose a system that achieves this traffic distribution by biasing search walks using the Metropolis-Hastings algorithm [5], [12] without requiring any special underlying topology. We finish the paper by comparing our method with Gia [4], where we find in simulation that the former outperforms the latter under all studied conditions, two novel saturation metrics introduced in this paper, and such end-to-end parameters as query success rate, latency, and query-hits.

1 Introduction

Recent measurement studies (e.g., [15]) show that P2P overlays contain nodes of varying *capacity*, which is a general term describing the ability of each user to router traffic, store object replicas, and answer queries. If a P2P system is not capacity-aware, slow nodes may become overwhelmed by messages even when many high-capacity peers in the network are still under-utilized. Therefore, node heterogeneity plays a key role in the design and future success of unstructured P2P systems. In this paper, we propose a set of metrics for measuring the throughput of heterogeneous P2P networks under random-walk traffic, create algorithms for maximizing search and replication ability of peers, and evaluate their performance in simulations. We start with basic definitions and assumptions.

1.1 Assumptions

In this paper, we assume a general unstructured P2P network that utilizes random walks (as opposed to flooding [6], [22] or some hybrid methods [3], [23]) for finding neighbors, searching for content, and replicating existing file pointers. The reason for using random walks is their ability to achieve arbitrary stationary distributions (see below) across existing nodes in the system and clear bounds on overhead. The capacity C_i of user i determines its ability to process incoming messages, where excess traffic is backlogged at each user in some infinite queue. Also assume that joining users, as well as peers seeking new neighbors, perform so-called *build walks* with TTL $k_b \geq 2$ and select peers at the end of these walks as their neighbors. Note that the transition probability of these random walks influences the topology of the resulting overlay.

Once the graph is built, *search walks* are then used for discovering the desired content. Nodes looking for certain files start several random walks with TTL $k_s \geq 2$ and examine the content of each peer through which these walks pass. This query is called *successful* if any of the search walks started by a node passes through at least one user that shares the desired file or knows which peer does. The number of found users holding the file is called *query-hits*, which determines the download's ability to be parallelized and its robustness against failure.

To increase query success rates and achieve redundancy, nodes replicate pointers to their shared files to other peers, as long as the number of replicas stored at each node i does not exceed its capacity C_i. When publishing their content into the system, nodes start a *replication walk* with TTL $k_r \geq 2$ and select one or more nodes along the walk to hold pointers to all of their content. After replication, any query for this content can also be answered by one of its replicas thereby improving the query success rate of the P2P system.

A well-designed P2P system should possess build, search, and replication components that guarantee a high network-wide throughput under load and achieve good performance using such end-to-end parameters as query success rates, query-hits, and latency to find the first result. Next, we discuss the challenges involved in designing these three components and then outline our proposed system to address these issues.

1.2 Challenges

One direction in constructing heterogeneous P2P systems [13], [20] is to ensure that node degree d_i is linearly proportional to capacity C_i. However, in real networks, node capacities are extremely diverse [15], which means that this linear dependence often produces nodes with very

large degree. Such high degree may not be feasible in practice due to the high overhead involved in maintaining neighbor connections, the large number of failed links each time a high-capacity node departs the system, and the difficulty in maintaining $d_i \sim C_i$ under churn. Another direction [4], [20] is to continuously perform *dynamic topology adaptation*, which involves nodes constantly replacing their existing neighbors with better ones so as to satisfy some performance objective. In P2P networks with heavy-tailed lifetimes [21] where the majority of nodes fail quickly, these systems almost constantly remain in suboptimal states, which causes enormous traffic overhead and leads to lower performance since the system never reaches the desired (i.e., stable) state.

For a P2P system to function properly, it is highly desirable that it also not be congested by non-topology-related traffic, which primarily consists of search and replication walks. To properly load high-capacity nodes, existing search methods heavily rely on special overlay properties (e.g., $d_i \sim C_i$ or underlying adaptation algorithms); however, more fundamental questions of whether it is possible to construct a well-behaving system that does not depend on a specific topology, whether it can function regardless of churn, and what search algorithm achieves optimal throughput have not been explored before. Since search algorithms typically visit higher-capacity nodes more often, it makes sense to replicate content to those nodes with a higher probability. However, existing schemes such as 1-hop replication [4], in which replicas are stored on immediate neighbors, again rely on the stationary properties of the overlay topology, which are not easily achievable under churn. Furthermore, as there is no well-known correlation between file popularity and the capacity of neighbors of the user sharing the file, 1-hop replication is biased against peers with low degree and/or low-capacity neighbors.

1.3 Our Contributions

We start by defining metrics for assessing the overhead and maximum throughput of a given topology-construction algorithm. To quantify the amount of traffic necessary to maintain a given overlay, we introduce a metric we call *Build Saturation Point* (BSP) as the maximum node arrival/departure rate μ for which the average queue length $E[Q]$ in the system in some time-interval $[0, t]$ is no larger than c seconds, where c is a constant. Networks with higher BSP experience less overhead due to churn and can thus retain more spare capacity for other types of traffic (e.g., queries and replication). Our second metric, which we call *Search Saturation Point* (SSP), quantifies the overlay's capacity for sustaining random walks and is defined as the maximum rate at which walks of a certain TTL can be completed in the overlay. Higher SSP allows more queries to be answered and more objects replicated per time unit.

We next focus on defining a network that achieving optimal SSP and then building a decentralized approximation to it. For a given number of users n, set of capacities $\{C_1, \ldots, C_n\}$, fixed average degree d, and random walk length k, we define a network $\mathcal{N}$ with a search algorithm $\mathcal{S}$ to be *throughput-optimal* if pair $(\mathcal{N}, \mathcal{S})$ achieves the maximum possible SSP equal to $\sum_i C_i/k$. We prove that this optimality can be realized if the stationary probability of random walks visiting a node i is simply

$$\pi_i = \frac{C_i}{\sum_{j=1}^{n} C_j}, \tag{1}$$

which means that each node must be loaded proportionally to its capacity. We should note here that when all nodes in the network are congested, further increase in query rate actually decreases the completion rate of random walks and hurts performance, even though the rate at which individual messages are processed remains constant.

We then provide a framework we call *Capacity-Proportional Metropolis-Hastings* (CPMH) for achieving (1). CPMH applies the Metropolis-Hastings algorithm [5], [12], a Markov chain based sampling method, to calculate the transition probability of random walks. CPMH does not impose additional restrictions on the overlay topology to achieve capacity-proportionality. To illustrate this, we apply CPMH on top of various topologies (such as BA [2] and Gnutella [11]) and show the convergence of random walks to the target stationary distribution given by (1).

We then build a heterogeneous P2P system by designing its three components – topology, search, and replication. We first propose an overlay topology we call *Capacity-Scalable Out-Degree* (CSOD), in which the out-degree $d_{out}(i)$ of node i is given by

$$d_{out}(i) = a + \lfloor b \log_{10} C_i \rfloor, \tag{2}$$

where a, b are some constants. Observe that node degree in CSOD is sublinear in C_i and hence the new approach is more scalable as $C_i \to \infty$ than the methods presented in [20]. For searching, we use CPMH random walks and show that among the studied topologies, CSOD achieves the fastest convergence to the target stationary distribution $\pi = (\pi_1, \ldots, \pi_n)$. We call the resulting combination CSOD-CPMH and use it throughout the paper. To address the drawbacks of existing replication schemes such as 1-hop replication, we propose a method we call *CPMH-replication*, which uses CPMH walks for replicating shared content and places greater responsibility on higher-capacity nodes without requiring any special overlay properties.

We finally perform simulation-based evaluation of the proposed CSOD-CPMH system and compare its performance with that of Gia. Simulations show that CSOD-CPMH typically achieves four times higher SSP than Gia and close to 100 times higher BSP. In a static network with CPMH-replication, CSOD-CPMH achieves 15% higher query success rates, ten times smaller query latencies, and twice the number of query-hits when compared to Gia. We also evaluate these networks under churn by simulating Poisson node arrivals with Pareto lifetimes and find that

CSOD-CPMH obtains 20% higher query success rates at half the latency when compared to Gia.

Overall, CPMH achieves close-to-optimal traffic distribution, high SSP/BSP, and good end-to-end performance without requiring any special overlay topology. This property enables incremental deployment of the proposed P2P components on existing networks such as Gnutella.

The rest of the paper is organized as follows. In Section 2, we discuss related work. In Section 3, we establish optimality of capacity-proportional networks and in Section 4, we describe the CPMH framework and build a capacity-proportional system using it. In Section 5, we provide simulation results and in Section 6 conclude the paper.

2 Related Work

Many random walk techniques [1], [8], [9], [14], [18] have been proposed to reduce the overhead of flooding in unstructured P2P systems like Gnutella [11]. Among the approaches that explicitly take into account heterogeneous capacity of end-users, Gia [4] is one prominent example of an unstructured P2P network in which nodes seek neighbors that match their own capacity in a process called *dynamic topology adaptation*, which involves replacing existing neighbors with more suitable options based on their capacity and degree. The rate of adaptation of the nodes is controlled by their *satisfaction level*, which indicates whether their neighbors have enough capacity to handle the arriving traffic.

Vishnumurthy *et al*. [20] discuss several build and search walk strategies for heterogeneous networks. In these methods, nodes maintain their out-degree $d_{out}(i)$ linearly proportional to their capacity C_i and then bias build walks to achieve in-degree $d_{in}(i) = d_{out}(i)$. Kwong *et al*. [13] propose a protocol for constructing heterogeneous P2P networks, where a node joining the network starts random walks to preferably connect to nodes with *high capacity per neighbor*. In this system, the stationary probability of build walks is given by

$$\pi_i = \frac{C_i/d_i}{\sum_{j=1}^{n} C_j/d_j},$$

where n is the number of nodes in the network. The transition probability of these walks is calculated using the Metropolis-Hastings algorithm [12].

Additional prior work [7], [17] on P2P networks employs the Metropolis-Hastings algorithm, but it addresses entirely different problems related to unbiased node/data sampling. Among the existing literature, Gia [4] is the only complete system that provides topology construction, search algorithms, and replication strategies. Therefore, during evaluation, we compare our system only with Gia.

3 Optimal Network

In this section, we define optimality of a P2P network under random-walk traffic and derive the corresponding stationary distribution.

3.1 Basics and Assumptions

The P2P systems considered in this paper use random walks for key overlay operations such as topology construction, search, and content replication. As a result, a major part of the overlay traffic is due to random walks, whose distribution across online users determines the volume of messages that can be handled by a P2P system and thus its throughput. We define a network to be *throughput-optimal* if it achieves the maximum rate of completion of random walks for a given average degree d of the system and node-capacity distribution $\{C_1, \ldots, C_n\}$.

As before, the capacity C_i of a node i is the maximum rate at which it can handle incoming messages, where queuing at each node is assumed with infinite buffers. Observe that the traffic distribution among nodes in the network can be controlled by the stationary probability π of random walks. If T is the total traffic rate in the network, then the average rate of incoming messages at node i

$$T_i = \pi_i T, \tag{3}$$

where π_i is the stationary probability for the walks to visit node i. Intuitively, to achieve maximum completion of random walks in the network, the traffic distribution among nodes should be proportional to their capacity such that all nodes are loaded to their maximum, but none are yet congested. In this section, we provide a simple proof for this.

Consider a connected, undirected graph G with n nodes. Walks are started from a node i at rate λ_i and continue for k hops. Let $\Lambda = \sum \lambda_i$ be the total input rate in the network and let M be the completion rate of these walks. At a node i, if $T_i > C_i$ then messages are added to the node's input queue, which is processed by the node at rate C_i. By its definition, a throughput-optimal network achieves $M = \Lambda = \sum_{i=1}^{n} C_i/k$, which is the absolute maximum possible for a given n and capacity distribution $\{C_i\}$ (i.e., all nodes are fully utilized and every $1/k$-th message is completing some random walk). Our first result links traffic volume at each node with stationary distributions of random walks.

Lemma 1 *Assume that random walks are started at each node with rate λ_i and proceed according to a positive irreducible Markov chain with transition matrix P. If k is larger than the mixing time of P, the following holds*

$$T_i = \pi_i k \Lambda, \tag{4}$$

where π_i is the unique solution to $\pi = \pi P$.

Recalling that random walks on an undirected graph form a positive irreducible Markov chain, the next result follows.

Theorem 1 *Assuming k is sufficiently large, the optimal stationary distribution of random walks is*

$$\pi_i = \frac{C_i}{\sum_{j=1}^{n} C_j}. \tag{5}$$

We next use this result to quantify the capacity of P2P systems in supporting search walks.

3.2 Search Saturation Point

To evaluate an overlay network in terms of its ability to route traffic, one requires a metric that is independent of the specific file popularity and replication strategy. For this purpose, we propose a new metric, which we call *Search Saturation Point* (SSP), to express the capacity of an overlay topology in supporting random walks.

Consider an overlay graph G with n nodes of capacity $\{C_1, \ldots, C_n\}$ and average node degree d. In this network, random walks of length k are started from randomly selected nodes. As earlier in this section, let Λ be the rate at which these walks are started and $M(\Lambda)$ be the completion rate of the walks. In the beginning, as Λ increases, $M(\Lambda)$ also increases until a certain point where the network is saturated. Beyond this point, any increase in Λ results in additional message backlog at nodes and thus decreases $M(\Lambda)$. The curve $M(\Lambda)$ has a unique global maximum, which we define as the SSP of graph G

$$SSP = \max_{\Lambda} M(\Lambda). \tag{6}$$

If a P2P system uses search walks, then the SSP signifies the query completion rate that can be achieved in the network. Unlike graph properties such as expansion (i.e., second eigenvalue λ_2), which characterize G's topological properties, the SSP is a more direct measure of the overlay's effect on queries which are run over it. During evaluation, we perform simulations to obtain the SSP of various P2P networks.

3.3 Centralized Construction

While comparing different overlays using SSP, one needs a standardized upper-bound on the achieved performance. For this purpose, we next create an optimal network (OPT), which has the maximum SSP for a given capacity distribution $\{C_i\}$, and use it in all comparisons later in the paper.

To obtain the optimal stationary distribution (5), we first construct a network in which $d_i = C_i$ and then run unbiased random walks on this network, which gives

$$\pi_i = \frac{d_i}{\sum_{j=1}^{n} d_j} = \frac{C_i}{\sum_{j=1}^{n} C_j}. \tag{7}$$

Consider Algorithm 1, which first generates node capacities $\mathbf{C}$ from a given distribution $\{C_i\}$. For each node i, the goal is to attempt selecting $d_i = C_i$ neighbors from the available neighbor slots $\mathbf{AS}$. While choosing neighbors of i, duplicate edges and self-loops are prevented by not considering i (Line 8) or its existing neighbors (Lines 6, 13) as available for connectivity. Using this $\mathbf{AS}$ vector, a random neighbor r is selected in Line 11 with probability

$$p(r) = \frac{AS[r]}{\sum_{j=1}^{n} AS[j]},$$

Algorithm 1 Create Optimal topology

```
1:  C ← capacities of n nodes.
2:  for each node i do
3:      d_i ← C[i]                          ▷ Required degree
4:      AS ← C                              ▷ Available neighbor slots
5:      for each neighbor j of node i do
6:          AS[j] ← 0                       ▷ Avoid duplicate edges
7:      end for
8:      AS[i] ← 0                           ▷ Avoid self loops
9:      tas = Σ_{j=1}^{n} AS[j]             ▷ Total available slots
10:     while (d_i > 0 && tas > 0) do
11:         r ← getRandomNode(AS)           ▷ Biased by AS
12:         addEdge(i, r)
13:         AS[r] ← 0                       ▷ Update available slots
14:         tas ← tas − C[r]
15:         C[r] ← C[r] − 1                 ▷ Update unsatisfied degree
16:         C[i] ← C[i] − 1
17:         d_i ← d_i − 1
18:     end while
19: end for
```

Capacity	Fraction
1	0.65
10	0.3
100	0.049
1000	0.001

Table 1. Capacity distribution of nodes.

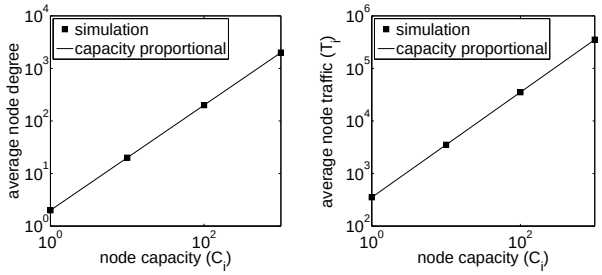
which ensures that the generated graph is random and the nodes requiring high-degree get sufficient neighbors. This step is repeated until no more available slots are left or the required degree d_i is met.

Graphs constructed using Algorithm 1 exhibit $d_i \approx C_i$. For generating graphs with exact degree distributions $d_i = C_i$ one can use randomized variations of the Havel-Hakimi algorithm [10], [19]. But unlike these methods, Algorithm 1 is faster and simpler due to its relaxed constraints. This algorithm can also produce graphs with a degree distribution close to the required distribution even when the exact target degree-set $\{C_1, \ldots, C_n\}$ is infeasible.

We next apply Algorithm 1 to construct an OPT P2P network with $n = 10,000$ nodes with the capacity distribution shown in Table 1, which was obtained by prior measurements [15] of Gnutella. The target degree of nodes in this setup is twice their capacity, i.e., $d_i = 2C_i$. Observe in Figure 1(a) that the expected degree $E[d_i]$ of each node indeed equals $2C_i$ and in Figure 1(b) that traffic volume achieved by unbiased random walks is, as expected, capacity-proportional.

4 Distributed Capacity Proportionality

The previous section shows that (5) can be achieved using Algorithm 1; however, it assumes centralized construction and static node conditions (i.e., no churn). This sec-



(a) degree (b) traffic

Figure 1. Capacity-proportionality in OPT.

tion provides a framework for implementing throughput-optimality using distributed operation and under dynamic node arrival/departure.

4.1 CPMH Framework

The stationary distribution π of random walks depends on their transition probability matrix P and the overlay graph G on which they are performed. Therefore, to achieve a particular π, one can construct overlays with special properties, e.g., $d_i = C_i$, and then run *unbiased* random walks on these networks. However, in P2P systems with churn, maintaining these special overlays is not just challenging, but sometimes simply impossible. As a result, it is desirable to design transition matrix P so as to achieve the target π *without* imposing any restrictions on topology G. In this direction, we use the Metropolis-Hastings algorithm, a Markov-chain sampling method [5], [12], to find P that achieves the optimal π. We call this technique *Capacity-Proportional Metropolis-Hastings* (CPMH) and describe it next in more detail.

In CPMH, the next transition of a random walk from a node i is found in two steps. Define $N(i)$ to be the set of i's neighbors. First, we choose one neighbor among the users in $N(i)$ to which the walk should transition. The probability $q(i,j)$ to select $j \in N(i)$ is given by

$$q(i,j) = \begin{cases} C_j / \sum_{x \in N(i)} C_x & j \in N(i) \\ 0 & j \notin N(i) \end{cases}. \quad (8)$$

Then, the random walk transitions to the above selected node j with probability $\alpha(i,j)$ or stays at node i with probability $1 - \alpha(i,j)$, where the acceptance probability α is given by

$$\alpha(i,j) = \begin{cases} \min\left(1, \frac{\sum_{x \in N(i)} C_x}{\sum_{x \in N(j)} C_x}\right) & j \in N(i) \\ 0 & j \notin N(i) \end{cases}. \quad (9)$$

Note that in the Metropolis-Hastings algorithm, functions $q(.)$ and $\alpha(.)$ are given in some general form, while the above two equations are customized to our particular needs. While several choices of $q(.)$ are possible, we found that (8) performed better in terms of achieving the target distribution π.

4.1.1 Self Transitions

Observe that in CPMH, self transitions at node i occur when a selected jump $i \rightarrow j$ is rejected, which happens with probability $1 - \alpha(i,j)$. From (9), notice that the acceptance probability $\alpha(i,j)$ is low for transitions from a high-capacity node i to a low-capacity node j. Therefore, when a random walk hits a high-capacity node in its path, it is likely to undergo multiple self-transitions and become "stuck" there. In real networks, however, these self-loops are just virtual hops and do not count toward network traffic. Three intuitive observations follow from this: 1) the TTL of CPMH walks is implicitly adapted based on the capacity of nodes visited and is generally shorter when passing through high-capacity nodes; 2) if the replication scheme ensures that high-capacity nodes in the network share more resources, then walks passing through these users may need to visit fewer peers to get the required number of query hits; and 3) random walks are much more likely to terminate at high-capacity nodes, which ensures that search, replication, and neighbor selection favors users with sufficient resources and desired content.

4.2 Topology Construction

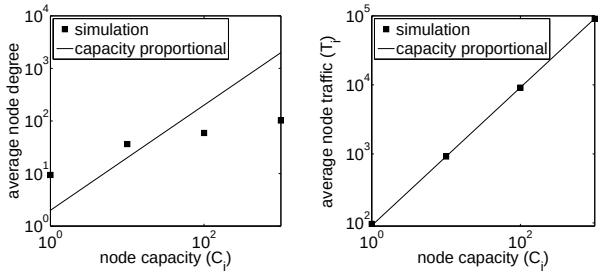
In this section, we describe the overlay topology of our proposed system. In this network, a new node i joining the system will start $d_{out}(i)$ build walks for selecting its out-neighbors among existing peers. The desired out-degree $d_{out}(i)$ of node i is given by

$$d_{out}(i) = a + \lfloor b \log_{10} C_i \rfloor, \quad (10)$$

where a and b are constants. During simulations, we use $a = 4$ and $b = 15$ to achieve an average degree equal to that of other networks evaluated in this paper. In (10), observe that the out-degree of a node is not linearly proportional to its capacity (as in prior approaches) and thus scales well in networks with an extremely wide range of node capacities. We call this approach *Capacity Scalable Out-Degree* (CSOD).

It should be noted that CSOD uses *unbiased* build walks; however, CPMH walks can also be used for the construction of overlays if the system can tolerate in-degree linearly proportional to C_i. In certain cases, however, such high degree may not be feasible in practice due to the overhead involved in maintaining a large set of neighbors. Since we generally assume that the volume of build traffic is insignificant compared to that of search traffic, the use of unbiased build walks has little effect on capacity-proportionality of combined traffic at each peer.

Figure 2(a) shows the average node degree in a $10,000$-node CSOD network with the capacity distribution given by Table 1. Observe in the figure that node degree scales sublinearly (i.e., logarithmically) as capacity increases. At the same time, Figure 2(b) shows that the average volume of CPMH traffic (TTL = 1024 hops) at each node is in fact capacity-proportional.



(a) degree (b) traffic

Figure 2. CPMH walks on the CSOD topology.

4.2.1 CPMH Walk Convergence

In general, CPMH does not require any special underlying topology (e.g., CSOD) for achieving capacity-proportionality of search traffic. To illustrate this, we show the convergence of CPMH walks to the optimal stationary distribution π using two additional topologies – BA [2] and Gnutella [11] – that do not consider node capacity during construction of the overlay.

To measure convergence of random-walk distributions, we use the *difference metric* from [24]. Assuming $T(C_i)$ is the average amount of traffic passing through all nodes of capacity C_i, then the convergence error of CPMH walks is given by

$$\phi = \frac{1}{2} \sum_i \left| \frac{T(C_i)}{\sum_i T(C_i)} - \frac{C_i}{\sum_i C_i} \right|. \qquad (11)$$

On the evaluated topologies, CPMH walks are started at random nodes and at a constant rate of 50 new walks/s. Convergence is evaluated by computing ϕ after 1000 seconds of simulation time. The TTL of these walks is increased until ϕ becomes less than or equal to 0.01, which allows us to compute the smallest TTL that allows the empirical distribution π to approximate the desired one with sufficient accuracy.

Table 2 shows the minimum TTL required to achieve convergence of CPMH walks on different topologies. Observe that due to its capacity-aware nature, CSOD achieves dramatically faster convergence than the other two graphs. While CPMH can indeed be applied incrementally over any underlying overlay structure, convergence time of random walks in such cases may be longer than in CSOD.

4.3 Search Methodology

In the proposed system, CPMH walks are used for propagating search requests. Specifically, a node i looking for a file, starts a search walk of TTL t_s. When this walk passes through a node containing the required file, a query-hit message is sent back to i. Search walks continue until their TTL reduces to 0 or the maximum number of query-hits, specified by the query-initiator i, is reached. For a query to succeed, it should result in at least one query-hit message.

Topology	TTL
CSOD	50
BA	640
Gnutella	620

Table 2. Convergence TTL for CPMH walks.

As CPMH walks are used for query propagation, the search load in the proposed system is capacity-proportional. To indicate the topology and the type of search walks used, we call the proposed system CSOD-CPMH.

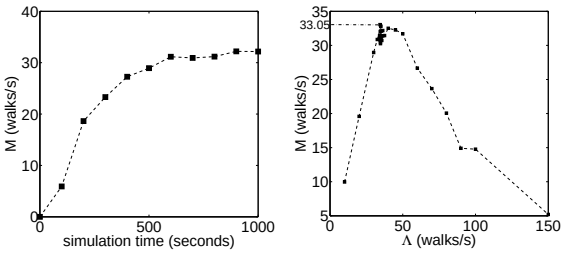
4.4 Replication Strategy

In a P2P network, file replication involves storing replicas of shared files in other nodes. Replication is used to improve the query success rate and reduce the query latency by making pointers to shared files available in the paths of query walks. One-hop [4] and random-walk replication [8], [16] are the two most common schemes used in unstructured P2P networks. To ensure a capacity-proportional distribution of replication load in the network, we propose to use the random-walk replication scheme with CPMH walks and call this strategy *CPMH replication*. In this scheme, to achieve a replication factor r, every node starts a CPMH walk with TTL k_r. This walk first transitions for h_f hops and then starts replicating at every unique node visited until the required replication factor r is achieved or TTL reduces to 0. The value of h_f depends on the mixing time of random walks on the network.

Unlike 1-hop replication, which requires capacity-proportional degree distribution to achieve the same effect as our approach, CPMH-replication works with any underlying topology and maintains the desired replication distribution in dynamic networks under churn, while requiring almost no additional overhead during construction of the graph. In addition, CPMH-replication does not exhibit bias against files shared by low-capacity nodes, which is a major limitation of 1-hop replication as there is no well-known correlation between file popularity and capacity of the nodes sharing them. CPMH-replication overcomes this problem by ensuring probabilistically equal visibility of files shared by all nodes in the network, irrespective of their capacity.

5 Evaluation

In P2P systems that use random walks for searching for content in the network, user-perceived metrics (e.g., query success rate) can be improved in two stages – 1) building an overlay topology that supports high random-walk completion rates and 2) designing a suitable replication scheme that ensures that shared files are present with high probability in the paths of these random walks. Along these lines, we follow a two-step evaluation process, where we first evaluate how well each P2P topology can support search requests and then analyze end-to-end metrics to measure the



(a) stabilization of M (b) SSP = 33.05 walks/s

Figure 3. SSP of OPT-unbiased.

performance of these networks for different file replication schemes.

Throughout this section, we perform simulation-based evaluation of the proposed CSOD-CPMH network and compare it with Gia-biased, OPT-unbiased, and CSOD-biased systems, where the naming convention consists of {topology}-{search walk}. For example, the Gia-biased system uses Gia overlay rules (i.e., dynamic topology adaptation [4]) and capacity-biased search walks whose transition probability is given by

$$p(i,j) = \begin{cases} C_j / \sum_{x \in N(i)} C_x & j \in N(i) \\ 0 & \text{o.w.} \end{cases}. \qquad (12)$$

The OPT-unbiased system uses the optimal topology constructed using Algorithm 1, and unbiased walks for searching. As this system achieves the optimal stationary distribution π, it is used as an upper bound on the SSP while comparing different P2P systems. Finally, CSOD-biased is introduced to evaluate the need for CPMH search walks in the proposed CSOD-CPMH system.

It should be noted that message flow-control is an additional mechanism for preventing backlog in the system. In this paper, however, we are only focused on the pure properties of topology, search, and replication mechanisms, while flow control is an orthogonal approach that can be used to enhance all studied methods and will be considered in future work.

5.1 Topology Evaluation

We start by calculating the Search Saturation Point (SSP), as defined in Section 3.2, of different overlay topologies. In these simulations, we use search walks with $k_s = 1024$ on networks with $10,000$ nodes and capacity distribution in Table 1. To find the SSP of a network, we perform a binary search by varying the query input rate Λ and measuring the corresponding completion rate $M(\Lambda)$. For each Λ, the network is allowed to stabilize before calculating the corresponding $M(\Lambda)$. Here, a network is said to be stable if for 3 successive 100-second intervals, the rate of completion M is within 5% of the previous completion rate. Figure 3(a) shows this stabilization process for a single Λ in the OPT-unbiased system and Figure 3(b) plots the corresponding curve $M(\Lambda)$ whose globally unique maximum is 33.05 walks/s, which is the SSP of this system.

Name	SSP
OPT-unbiased	33.05
CSOD-CPMH	27.75
Gia-biased	6.60
CSOD-biased	5.94

Table 3. SSP of several overlay networks.

Table 3 compares the SSP of the four evaluated overlays. Observe that OPT-unbiased has the highest SSP because it achieves capacity-proportional traffic with the highest accuracy. The proposed approach CSOD-CPMH has a slightly lower SSP due to the finite TTL and distributed construction; however, its throughput is much higher than that of the other two methods. Gia's topology adaptation ensures that high-capacity nodes have high degree, but it does not aim to achieve perfect capacity-proportionality, which results in reduced overall throughput in the system. CSOD-biased also performs poorly because it uses capacity-biased search walks, which again do not result in capacity-proportional traffic. This shows that CSOD by itself is insufficient for approximating the ideal system OPT and that it must be paired with CPMH random walks.

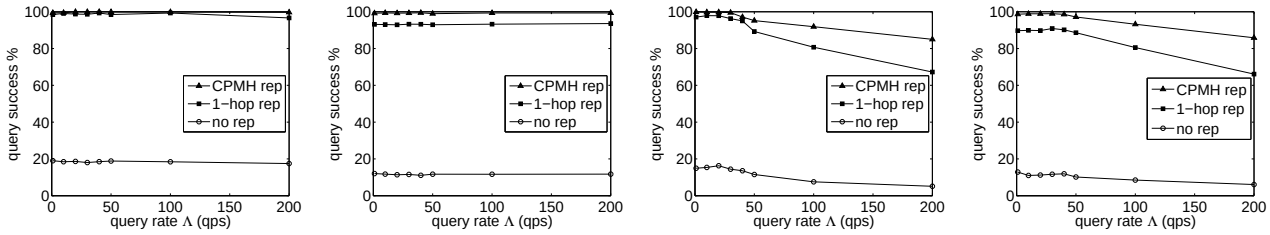
5.2 Replication Schemes

To evaluate CPMH-replication, we compared it with 1-hop and no replication. The replication factor for 1-hop replication is $r = E[d_i] = 20$ (i.e., the average degree). To achieve this replication factor during CPMH replication, a node starts a replication walk with initial forwarding length $h_f = 50$ and TTL $k_r = 200$. To capture the replication ability of a node, we limit the number of replicas at each node i to its capacity C_i.

We next perform simulations to study how each of the four combinations of topology/search mechanisms handle replication. Each shared file j has popularity P_j, which is the fraction of users that hold it. Variables $\{P_j\}$ are drawn from a Pareto distribution with shape parameter $\alpha = 3$ and expected popularity $E[P_j] = 0.001$. Queries for file j are started with frequency proportional to P_j and continue for 200 hops. Figure 4 shows the effect of replication on query success rates in various P2P networks. Observe in the figure that success rates improve from 20% to over 90% from replication and that CPMH-replication is more effective than 1-hop in all cases except OPT-unbiased where the two are identical due to the topological properties of the underlying graph.

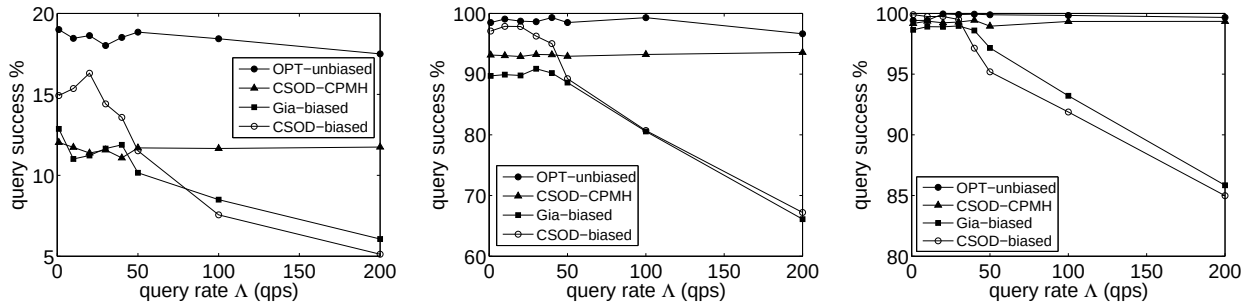
5.3 End-to-End Metrics: Static Network

We now apply these replication schemes to various static networks (i.e., no churn) and compare them using three end-to-end metrics – success rates, latency, and query hits.



(a) OPT-unbiased (b) CSOD-CPMH (c) CSOD-biased (d) Gia-biased

Figure 4. Effect of file replication on query success rate.



(a) no replication (b) 1-hop replication (c) CPMH replication

Figure 5. Query success rate.

5.3.1 Query Success Rate

Recall that a query is said to be successful if it results in at least one query-hit. Figure 5 shows success rates of different networks under three types of replication schemes. Observe that OPT-unbiased, which has a centralized construction, achieves the highest query success rate in all three replication scenarios and acts as an upper bound for comparison. CSOD-CPMH achieves close-to-optimal query success rates (i.e., over 99%) under CPMH-replication, while performing slightly worse under 1-hop and no replication. Due to capacity-proportionality achieved by CSOD-CPMH, it balances the load in the network and maintains a constant success rate even at higher query rates, which cannot be said about Gia-biased or CSOD-biased whose rates plummet as Λ increases. In fact, CSOD-CPMH is $15-25\%$ more successful than Gia-biased for $\Lambda = 200$ queries/sec.

5.3.2 Query Latency

Query latency is the delay to obtain the first result of a query. It shows the responsiveness of the system to the user, with lower latency indicating better performance. In these simulations, we measure the average latency of all successful queries at a given input rate Λ. Figure 6 compares the same four networks and three replication schemes using query latency in seconds. Observe that CSOD-CPMH not only has the lowest latency, but also that its delay to the first result is over 10 times smaller than in Gia.

Interestingly, the OPT-unbiased network has a higher latency than CSOD-CPMH under these conditions. This can be explained by the fact that OPT-unbiased uses unbiased query walks that visit low-capacity nodes earlier in their path compared to CPMH walks. While OPT is designed for maximizing query completion rates (i.e., SSP), it is *not* guaranteed to have the minimum query latency.

5.3.3 Query Hits

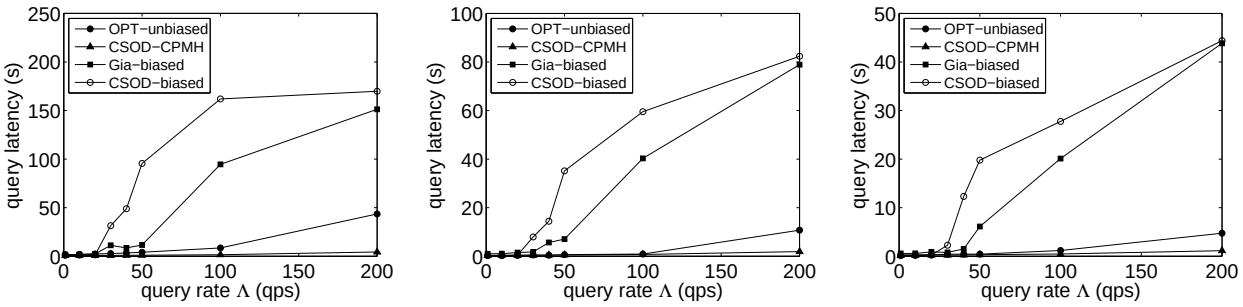
Total number of query-hits returned for all queries started in a network can also be used as an end-to-end metric to compare different P2P systems. In these simulations, there is no limit on the number of hits per query. Figure 7 shows that OPT-unbiased achieves the highest number of query-hits followed by CSOD-CPMH, which gets up to twice the number of hits compared to Gia-biased. This shows that more files can be discovered in a P2P system if the distribution of replicated files and the query traffic is capacity-proportional, especially at high search loads.

5.4 End-to-End Metrics: Dynamic Network

Node churn is a common characteristic of existing P2P networks in which peers depart and rejoin at a later time. In this section, we first describe our churn model and then define a metric that quantifies the overhead incurred by P2P overlays due to churn. We finally compare the various networks under churn, both in terms of their overhead and end-to-end metrics used in the previous subsection.

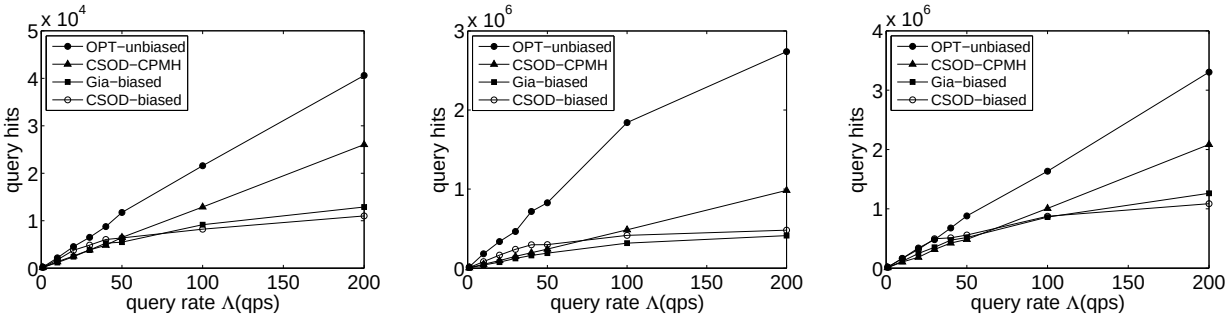
5.4.1 Churn Model

Many measurement studies (e.g., [21]) have shown that P2P users commonly exhibit heavy-tailed lifetimes. In our simulations, we initially construct an overlay with $10,000$ nodes and then start churn in this network. During the churn



(a) no replication (b) 1-hop replication (c) CPMH replication

Figure 6. Query latency.



(a) no replication (b) 1-hop replication (c) CPMH replication

Figure 7. Query hits.

stage, new nodes with Pareto lifetimes with $\alpha = 3$ arrive as a Poisson process. To keep the expected node count in the network constant, we match node arrival and departure rates. Specifically, if $E[L]$ is the mean user lifetime, then the departure rate of a node is $1/E[L]$ and that of a system with n nodes is $\mu = n/E[L]$. By varying $E[L]$, one can achieve different arrival and departure rates in the network.

5.4.2 Build Saturation Point

Under churn, incoming nodes seek new neighbors, while existing peers in the network constantly seek to replace their departed neighbors. Therefore, increase in the churn rate μ increases network traffic, which eventually results in backlog and increased queue size at each node. Let $E[Q]$ be the average queue length across all live nodes in the network during interval $[0, t]$, where t is some constant. To quantify the capacity of a P2P topology in handling churn, we define a metric we call *Build Saturation Point* (BSP) as the maximum μ for which $E[Q] \leq c$ for some constant c.

In simulations below, we use $t = 1000$ seconds and backlog threshold $c = 1$ second. In Figure 9, observe that the BSP of Gia and CSOD topologies differ by a factor of 100, where higher BSP of CSOD indicates dramatically lower build traffic and much higher rates of churn allowed by the proposed system. This result can be explained by the fact that Gia uses complex rules for satisfying neighbor requests and constant topology adaptation, which under churn often results in "unsatisfied" nodes starting many

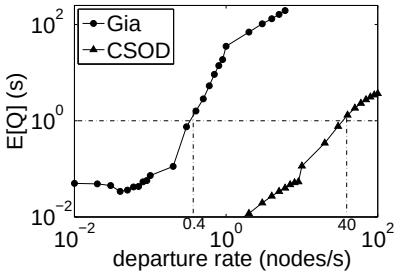
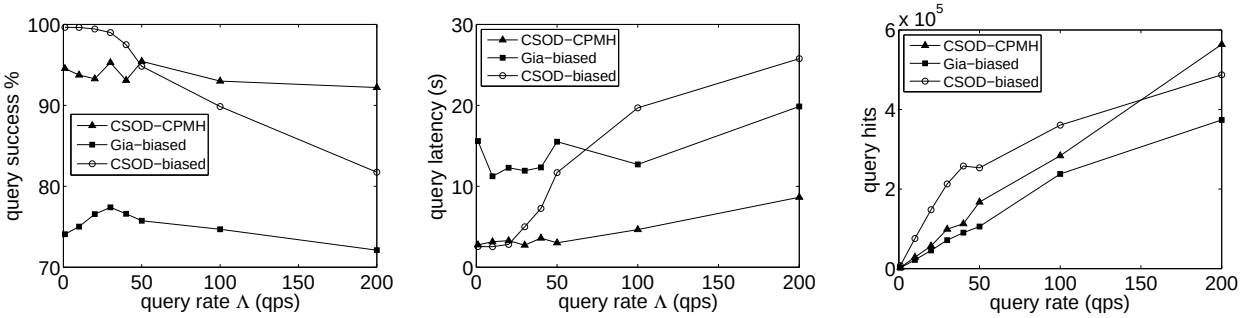


Figure 9. BSP of Gia and CSOD.

build walks to get a sufficient number of neighbors to maintain their connectivity. This in turn snow-balls into a large volume of build traffic and prevents Gia from reaching its ideal stable state.

5.4.3 End-to-end metrics

In our next simulation, we populate the system with $10,000$ nodes and then start churn. Waiting for $1,000$ seconds for the system to reach its steady-state, we start issuing queries at rate Λ and run the simulation for 500 additional seconds. Figure 8 compares the same four P2P networks under churn using our end-to-end query metrics. Observe that CSOD-CPMH beats Gia-biased in all three metrics, achieving 20% higher query success rates at half the latency. CSOD-CPMH also produces significantly more



(a) query success (CPMH replication) (b) query latency (CPMH replication) (c) query hits (CPMH replication)

Figure 8. Effect of Churn.

query-hits than Gia-biased, which shows CSOD-CPMH's exceptional ability to handle churn without performing continuous topology adaptation as in Gia.

6 Conclusion

We studied the question of designing P2P systems to handle high loads of random-walk traffic. We proved optimality of capacity-proportional distributions of random walks and provided a framework for achieving them without requiring construction of a special overlay topology. Using this framework, we built a candidate system CSOD-CPMH and showed in simulations that it had higher SSP/BSP saturation points and end-to-end performance than Gia.

Future work involves implementing CPMH and its deployment in real networks.

References

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

[2] A.-L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, Oct. 1999.

[3] J. M. H. Boon Thau Loo, Ryan Huebsch. The case for a hybrid p2p search infrastructure. In *IPTPS*, Feb. 2004.

[4] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker. Making gnutella-like p2p systems scalable. In *ACM SIGCOMM*, pages 407–418, Aug. 2003.

[5] S. Chib and E. Greenberg. Understanding the metropolis-hastings algorithm. *The American Statistician*, 49(4):327–335, Nov. 1995.

[6] V. Cholvi, P. Felber, and E. Biersack. Efficient search in unstructured peer-to-peer networks. *Euro. Trans. on Telecommun.*, 15(6):535–548, Nov. 2004.

[7] S. Datta and H. Kargupta. Uniform data sampling from a peer-to-peer network. In *IEEE ICDCS*, page 50, Jun. 2007.

[8] R. A. Ferreira, M. K. Ramanathan, A. Awan, A. Grama, and S. Jagannathan. Search with probabilistic guarantees in unstructured peer-to-peer networks. In *IEEE P2P*, Aug. 2005.

[9] C. Gkantsidis, M. Mihail, and A. Saberi. Random walks in peer-to-peer networks. In *IEEE INFOCOM*, pages 120–130, Mar. 2004.

[10] C. Gkantsidis, M. Mihail, and E. Zegura. The markov chain simulation method for generating connected power law random graphs. In *ALENEX*, pages 16–25, Jan. 2003.

[11] Gnutella. http://www.gnutella.com/.

[12] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, Apr. 1970.

[13] K.-W. Kwong and H. K. Tsang. Building heterogeneous peer-to-peer networks: protocol and analysis. *IEEE/ACM Trans. Networking*, 16(2):281–292, Apr. 2008.

[14] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *International Conference on Supercomputing*, pages 84–95, Jun. 2002.

[15] S. Saroiu, P. K. Gummadi, and S. D. Gribble. Analyzing the characteristics of napster and gnutella hosts. *Multimedia Systems*, 9:170–184, 2003.

[16] N. Sarshar, P. O. Boykin, and V. P. Roychowdhury. Percolation search in power law networks: Making unstructured peer-to-peer networks scalable. In *IEEE P2P*, Aug. 2004.

[17] D. Stutzbach, R. Rejaie, N. Duffield, S. Sen, and W. Willinger. On unbiased sampling for unstructured peer-to-peer networks. In *ACM IMC*, pages 27–40, Apr. 2006.

[18] W. W. Terpstra, J. Kangasharju, C. Leng, and A. P. Buchmann. Bubblestorm: resilient, probabilistic, and exhaustive peer-to-peer search. In *ACM SIGCOMM*, pages 49–60, Aug. 2007.

[19] F. Viger and M. Latapy. Efficient and simple generation of random simple connected graphs with prescribed degree sequence. In *COCOON*, pages 440–449, Aug. 2005.

[20] V. Vishnumurthy and P. Francis. On heterogeneous overlay construction and random node selection in unstructured p2p networks. In *IEEE INFOCOM*, Apr. 2006.

[21] X. Wang, Z. Yao, and D. Loguinov. Residual-based measurement of peer and link lifetimes in gnutella networks. In *IEEE INFOCOM*, pages 391–399, May 2007.

[22] B. Yang and H. Garcia-Molina. Improving search in peer-to-peer networks. In *IEEE ICDCS*, pages 5–14, Nov. 2002.

[23] M. Zaharia and S. Keshav. Gossip-based search selection in hybrid peer-to-peer networks. In *IPTPS*, Feb. 2006.

[24] M. Zhong, K. Shen, and J. Seiferas. The convergence-guaranteed random walk and its applications in peer-to-peer networks. *IEEE Transactions on Computers*, 57(5):619–633, May 2008.