Paper ID: manku-13750.ps

Structured Coupon Collection over Cliques for P2P Load Balancing

Krishnaram Kenthapadi
Stanford University
kngk@cs.stanford.edu

Gurmeet Singh Manku
Stanford University
manku@cs.stanford.edu

Abstract: *We study Structured Coupon Collection, a problem introduced by Adler et al [AHKV03] in STOC 2003, over n/b disjoint cliques with b nodes per clique. Nodes are initially uncovered. At each step, we choose d nodes independently and uniformly at random. If all the nodes in the corresponding cliques are covered, we do nothing. Otherwise, from among the chosen cliques with at least one uncovered node, we select one at random and cover an uncovered node within that clique. We show that as long as $bd \geq c \log n$, $O(n)$ steps are sufficient to cover all nodes w.h.p. and each of the first $\Omega(n)$ steps succeed in covering a node w.h.p.. These results are then utilized to analyze a stochastic process for growing binary trees that are highly balanced – the leaves of the tree belong to at most four different levels w.h.p. The stochastic process constitutes the core idea underlying a practical P2P load balancing scheme that beats earlier proposals for the same, in terms of message complexity.*

1 Introduction

In the standard coupon collector process, there are n types of coupons and in each trial, a coupon is chosen independently and uniformly at random. It is well known that the number of trials needed to collect at least one copy of each type is *sharply concentrated* around $n \log n$ (see [MR95], for example). One generalization is to have multiple choices: in each trial, pick d coupons at random and if any of them is not collected, collect a random uncollected coupon. Another generalization is to introduce a graph structure: coupon collection is carried out on a graph whose nodes correspond to coupons and are initially uncovered. In each trial, pick a node at random and if any of its neighbors is uncovered, cover a random uncovered neighbor. Adler *et al* [AHKV03] call this process *Structured Coupon Collection* over graphs. They establish that with high probability[1], $O(n)$ steps suffice to cover all nodes of hypercubes on n nodes, Δ-regular graphs with $\Delta = \Omega(\log n \log\log n)$ and random Δ-regular graphs with $\Delta = \Omega(\log n)$.

In Section 2, we analyze Structured Coupon Collection over n/b disjoint cliques, each of size b. In each trial, we choose $d \geq 1$ nodes independently and uniformly at random. If all the nodes in the corresponding cliques are covered, we do nothing. Otherwise, from among the chosen cliques containing an uncovered node, we select one at random and cover an uncovered node in it. We show that w.h.p., all the nodes are covered in $O(n)$ trials and each of the first $\Omega(n)$ trials covers an uncovered node, for any choice of b and d satisfying $bd \geq c \log_2 n$ for a suitably large constant c.

In Section 3, we use the results proved in Section 2 to analyze a stochastic process for growing binary trees, thereby extending the suite of results known in this space. Adler *et al* [AHKV03] showed that if we repeatedly perform a random walk down the tree and split the shallowest of the "hypercubic neighbors" of the leaf node encountered, the resulting tree has leaves in $\Theta(1)$ levels. Abraham *et al* [AAA$^+$03] and Naor and Wieder [NW03] showed that the same property holds for trees resulting from the following process: at each step, we perform $c \log n$ random walks down the tree and split the shallowest leaf node

[1]By "with high probability" (w.h.p.), we mean "with probability at least $1 - O(n^{-\lambda})$ for an arbitrary constant $\lambda > 1$.

encountered. Manku [M04] showed that if we perform one random walk, and split the shallowest leaf in the "vicinity" of the leaf node, where the vicinity has size $c \log n$, the resulting tree has leaves in at most three different levels, w.h.p. The process we analyze is a generalization of these two extremes (and includes both as special cases): we perform r random walks, inspect vicinities of size v for each of the r leaf nodes and then identify the shallowest leaf. We show that as long as $rv \geq c \log n$, the tree has leaves in at most four different levels, w.h.p. Analysis of the generalized process requires a new proof technique, for which we borrow ideas from [AHKV03] – the proof is different from the approaches taken by authors who analyze the extreme cases [AAA+03, NW03, M04].

In Section 4, we show how balanced binary trees are useful for addressing the load balancing problem in Distributed Hash Tables (DHTs). The tradeoff between r, the number of random walks and v, the vicinity-size, is exploited to arrive at the optimal number of random walks required. We show that with $r = \sqrt{\log \log n}$ random walks, $O(\log n / \sqrt{\log \log n})$ messages are sufficient w.h.p., when a new member joins the DHT. In comparison, existing schemes require $O(\log n)$ messages [AHKV03, M04] or $O(\log^2 n / \log \log n)$ messages [AAA+03, NW03, KR04].

2 Structured Coupon Collection over Cliques

Consider the problem of collecting b copies each of n/b coupons. In each trial, exactly one of the n/b coupons is chosen independently and uniformly at random. If b is a constant, the number of trials needed to collect all copies is *sharply concentrated* around $\frac{n}{b}(\ln \frac{n}{b} + (b-1) \ln \ln \frac{n}{b})$ [MR95]. In this section, we study the following variant: we have to collect b copies each of n/b coupons. In each trial, d coupons are chosen independently and uniformly at random but at most one of them can be retained to augment our collection: if we have already collected b copies of each of these d coupons, we do nothing; otherwise, from among the chosen coupons having less than b copies, we randomly select one to include in our collection. This process is equivalent to the process on cliques defined in Section 1. In this Section, our main results are that if $bd \geq c \log n$ for some suitably large constant c, then with high probability, (a) $O(n)$ trials suffice to collect b copies of all the coupons, and (b) each of the first $\Omega(n)$ trials increases the size of our collection.

2.1 Analysis

In terms of bins and balls, we have n/b bins, each with *capacity* b. In each trial, we choose d bins independently and uniformly at random. If all the d bins are full, we do nothing (the trial *fails*). Else, we select one of the non-full bins (from among the d choices) at random and place a ball into it. The first lemma below contains two useful forms of inequalities by Chernoff [C52] and Hoeffding [H63]. The second lemma helps us derive tail bounds for dependent binary random variables under certain conditions.

Lemma 2.1 *Let Z denote a random variable with a binomial distribution $Z \sim B(n,p)$.*

$$\text{For every } \lambda > 1, \quad Pr[Z > \lambda np] < (e^{\lambda - 1} \lambda^{-\lambda})^{np}.$$
$$\text{For every } a > 0, \quad Pr[Z < np - a] < e^{-a^2/(2np)}.$$

Lemma 2.2 *Let $\omega_1, \omega_2, \ldots, \omega_n$ be a sequence of random variables. Let $Z_1, Z_2, \ldots, Z_n$ be a sequence of binary random variables, with the property that $Z_i = Z_i(\omega_1, \ldots, \omega_{i-1})$. Let $Z = \sum_{i=1}^{n} Z_i$. Then*

$$Pr[Z_i = 1 \,|\, \omega_1, \ldots, \omega_{i-1}] \leq p \quad \Rightarrow \quad Pr[Z \geq k] \leq Pr[B(n,p) \geq k].$$
$$Pr[Z_i = 1 \,|\, \omega_1, \ldots, \omega_{i-1}] \geq p \quad \Rightarrow \quad Pr[Z \leq k] \leq Pr[B(n,p) \leq k].$$

Theorem 2.1 *There exists a constant α such that, with high probability, all bins are full in αn trials, for any choice of b and d satisfying $bd \geq c \log_2 n$ for a sufficiently large constant c.*

Proof: We will use the fact that $(1-\frac{1}{x})^x < e^{-1} < (1-\frac{1}{x})^{x-1}$ for all $x > 1$.

Let f denote the fraction of non-full bins at any time. Fraction f is non-increasing over time, and we divide the process into two phases: In Phase I, $f \geq 1/d$. In Phase II, $0 < f < 1/d$. The intuition underlying our analysis is as follows. In Phase I, many bins are non-full. Hence we make rapid progress in populating the bins, terminating the phase in $O(n)$ steps. In Phase II, progress is slow. However, from the perspective of an individual non-full bin, progress is fast enough to fill it in $O(n)$ steps.

Claim: Phase I terminates within $t_1 = (\frac{e}{e-1} + \epsilon_1)n$ trials, w.h.p., where ϵ_1 is a small constant.

Proof: At any time-step, the success probability, i.e., the probability that the ball gets placed into some non-full bin is at least $1-(1-f)^d > 1-1/e$. Let n_s denote the number of balls lying in various bins when Phase I terminates. Clearly $n_s \leq n$. Let T be the total number of trials in this phase and Y_t be the number of successes in the first t trials. $Y_t = \sum_{i=1}^{t} Z_i$ where Z_i is the indicator random variable corresponding to success in the i^{th} trial. Let ω_i denote the random choices available to the i^{th} ball. Then, $Pr[Z_i = 1 | \omega_1, \ldots, \omega_{i-1}] \geq 1-1/e$. Using Lemma 2.2 , we can conclude that

$$Pr[T > \tfrac{n(1+\delta)}{1-1/e}] = Pr[Y_{\frac{n(1+\delta)}{1-1/e}} < n_s] \leq Pr[B(\tfrac{n(1+\delta)}{1-1/e}, \tfrac{e-1}{e}) < n_s] \leq Pr[B(\tfrac{n(1+\delta)}{1-1/e}, \tfrac{e-1}{e}) < n]$$

Using Lemma 2.1, the probability is less than $e^{-n\delta^2/2(1+\delta)}$, which is $o(1/n^2)$ when $\delta = \epsilon_1(1-1/e)$. Thus Phase I terminates within t_1 steps w.h.p.

Claim: Phase II terminates within $t_2 = (2e + \epsilon_2)n$ trials, w.h.p., where ϵ_2 is a small constant.

Proof: Let C denote a bin that is non-full at the end of Phase I. The probability that C is one of the d bins selected is $1-(1-b/n)^d > db/2n$. Given that one of the bins is C, the probability that each of the other $d-1$ bins is full is $(1-f)^{d-1} > 1/e$. Overall, the probability that C gets the ball in any time-step in Phase II is at least $db/2en$. As before, it follows from Lemma 2.2 that the number of balls in C stochastically dominates[2] the random variable $B(t_2, db/2en)$. Using $bd \geq c \log_2 n$, Lemma 2.1 yields that, in t_2 trials, C becomes full with probability $1-o(1/n^2)$. By taking the union bound over all the n bins, Phase II terminates within t_2 steps w.h.p.

Choosing $\alpha = (\frac{e}{e-1} + 2e + \epsilon_1 + \epsilon_2)$, we find that αn trials are sufficient to fill all bins w.h.p., where ϵ_1 can be made arbitrarily small, and ϵ_2 can be made small by choosing a large c. □

Note that Theorem 2.1 holds even if at any step, we choose a non-full bin (from among the d choices) *arbitrarily* (for example, in an adversarial fashion).

Theorem 2.2 *With high probability, each of the first βn trials succeeds in placing a ball, for any $\beta < \frac{1}{2}$ and any choice of b and d satisfying $bd \geq c \log_2 n$, for a sufficiently large constant c.*

Proof: The proof follows from a series of four claims:

Claim: In any of the first βn trials, for any β, the probability that a specific bin receives a new ball is at most $\frac{b}{n(1-\beta)}$.

Proof: At any time-step, for a specific bin C,

$$\Pr[C \text{ is chosen}] = 1-(1-b/n)^d < db/n$$

Let f denote the fraction of non-full bins at any time-step. Then Pr[Ball is placed in $C \mid C$ is chosen] $= \sum_{i=1}^{d} (\frac{1}{i})\binom{d-1}{i-1} f^{i-1}(1-f)^{d-i} = (\frac{1}{df}) \sum_{i=1}^{d} \binom{d}{i} f^i (1-f)^{d-i} = \frac{1-(1-f)^d}{df} < \frac{1}{df}$. At the end of the first βn trials, the fraction of full-bins is at most β. Therefore, at any earlier time-step, $f > 1-\beta$. By conditioning on the number of non-full bins found in the d bins, we get

$$Pr[\text{Ball is placed in C} \mid \text{C is chosen}] < \tfrac{1}{d(1-\beta)}$$

Therefore, the probability that C receives a new ball is at most $\frac{db}{n} \cdot \frac{1}{d(1-\beta)} = \frac{b}{n(1-\beta)}$.

[2] A random variable X *stochastically dominates* random variable Y iff $Pr[X \geq r] \geq Pr[Y \geq r] \forall r \in \Re$.

Claim: For any $\beta < \frac{1}{2}$, there exists a constant $\mu > 1$ such that the probability that a specific bin becomes full at the end of βn trials, is at most $1/\mu^{\beta}$.

Proof: From Lemma 2.2 and the previous claim, the random variable $B(\beta n, \frac{b}{n(1-\beta)})$ stochastically dominates the number of balls received by a specific bin C in βn trials. Using the Chernoff/Hoeffding inequalities in Lemma 2.1, the probability that C becomes full is at most $1/\mu^b$ where $\mu = (\frac{\beta}{1-\beta})e^{(\frac{1-2\beta}{1-\beta})}$, and $\mu > 1$ iff $\beta < \frac{1}{2}$.

Claim: With high probability, the fraction of full bins at the end of βn trials is at most $1/\nu^b$, for some constant $\nu > 1$.

Proof: Let $\nu = \sqrt{\mu} > 1$. There are two cases:

a) $b < \log_\nu n - \log_\nu \log_\nu n$: Let I_i for $i = 1, \ldots, n/b$, denote a set of indicator variables, one per bin. The variable is 1 if the bin becomes full within βn trials. The set of variables are dependent but *negatively correlated* [DR98]. Therefore, for tail bounds on their sum, it suffices to replace them by a set of independent variables. The sum is dominated by the random variable $B(n/b, 1/\mu^b)$. Using the Chernoff/Hoeffding inequalities in Lemma 2.1, the number of full bins is at most $\frac{n}{b\nu^b}$ (where $\nu = \sqrt{\mu} > 1$) w.h.p., provided $b < \log_\nu n - \log_\nu \log_\nu n$.

b) $b \geq \log_\nu n - \log_\nu \log_\nu n$: Any process with $b \geq \log_\nu n - \log_\nu \log_\nu n$ dominates the corresponding process with $d = 1$. A simple application of Chernoff/Hoeffding inequalities in Lemma 2.1 shows that the first βn trials succeed w.h.p., for sufficiently large c.

Claim: Each of the first βn trials succeeds in placing a ball w.h.p., where $\beta < \frac{1}{2}$.

Proof: The fraction of full bins at the beginning of i^{th} trial, for any $i \leq \beta n$, is also at most $1/\nu^b$. Therefore, the i^{th} trial fails with probability at most $(1/\nu^b)^d = o(1/n^2)$, if c is sufficiently large. By taking the union bound over the first βn trials, we obtain that w.h.p., all of them succeed. □

The constant α in Theorem 2.1 can be improved (see Theorem 3.2 in Section 3). We suspect that further improvement is possible – a sharp threshold result should hold. Further, we speculate that Theorem 2.2 should hold for any $\beta < 1$, not just $\beta < \frac{1}{2}$. In Section 3, we use Theorems 2.1 and 2.2 to prove that binary trees resulting from a certain stochastic process are highly balanced.

2.2 Related Work

The classic balls-and-bins problem involves bins with infinite capacity and $d = 1$ (see Johnson and Kotz [JK77] or the book by Kolchin *et al* [KSC78]). Recently, there has been interest in the computer science community in analyzing the height of the fullest bin. With n bins and n balls, the height of the fullest bin is $\Theta(\log n / \log \log n)$ (see Gonnet [G81], Mitzenmacher [M96] and Raab and Steger [RS98]). For the case $d \geq 2$, a breakthrough was achieved by Azar *et al* [ABKU99] who showed that the height of the fullest bin is $\log \log n / \log d + \Theta(1)$, if the least-loaded bin among the d bins is chosen at each trial. For further results and a survey of proof techniques for $d \geq 2$, see Mitzenmacher *et al* [MRS01]. Our focus is on cliques, which are equivalent to bins with *finite* capacity. Moreover, we wish to bound the height of the bin with the *fewest* balls.

Structured Coupon Collection over graphs was defined by Adler *et al* [AHKV03] who proved that $O(n)$ steps suffice for covering all nodes of hypercubes, Δ-regular graphs with $\Delta = \Omega(\log n \log \log n)$ and random Δ-regular graphs with $\Delta = \Omega(\log n)$. Alon [A04] has shown that at least $n - \frac{n}{\Delta} + \frac{n}{\Delta} \log_e \frac{n}{\Delta}$ steps are necessary to cover all nodes for any Δ-regular graph. The processes analyzed in [A04, AHKV03] choose exactly one random node per trial. Our focus is on cliques but with multiple nodes chosen in each trial. In fact, we allow a tradeoff between the number of random choices and clique sizes by allowing any $\langle b, d \rangle$ satisfying $bd \geq c \log n$ for an n-node graph. In Section 4, this tradeoff is exploited to derive the optimal number of random messages to be sent for load balancing in Distributed Hash Tables.

3 Balanced Binary Trees

In this Section, we study a variety of stochastic processes over binary trees which result in highly balanced trees. Balance is measured in terms of the number of different levels to which leaf nodes belong. We begin with some definitions:

Level and vicinity: In a binary tree, the *level* of a node is the length of the path from the root to that node. The root has level 0. There are at most 2^ℓ nodes at level ℓ. The *vicinity* of a node at level ℓ is the set of $v(\ell)$ nodes at level ℓ that have a common ancestor at level $\ell - \log_2 v(\ell)$.

Functions $\lceil x \rceil$, r and v: Let $\lceil x \rceil \equiv 2^k$ where integer k satisfies $2^{k-1} < x \leq 2^k$. Let $r : \mathbb{N} \to \mathbb{N}$ be a monotonically non-decreasing function, i.e., $r(\ell + 1) \geq r(\ell)$. Let $v : \mathbb{N} \to \mathbb{N}$ be a function satisfying $v(0) = 1$ and $v(\ell) \leq 2^\ell$. Moreover, either $v(\ell + 1) = v(\ell)$ or $v(\ell + 1) = 2v(\ell)$. Thus $v(\ell)$ always equals some power of two.

To motivate the stochastic process that we analyze, we summarize related results established in earlier papers. Each paper studies a different stochastic process for growing the tree:

1. At each step, we carry out one random walk and split the leaf node encountered. Adler *et al* [AHKV03] and Naor and Wieder [NW03] show that leaf nodes belong to $\Theta(\log \log n)$ different levels w.h.p.
2. At each step, we carry out $c \log n$ random walks down the tree and split the shallowest leaf node encountered. Abraham *et al* [AAA+03] showed that after n steps, the tree has leaves in $\Theta(1)$ different levels. Naor and Wieder [NW03] analyze a similar stochastic process in which we first estimate $\log n$ and then carry out $c \log n$ random walks.
3. Let $v(\ell) = \lceil c\ell \rceil$, where c is a suitably-large constant. First, we carry out a random walk to reach a leaf node $\mathbf{r}$. If all nodes in the vicinity of $\mathbf{r}$'s parent are split, we split $\mathbf{r}$ itself. Otherwise, we split one of the leaf nodes in the vicinity of $\mathbf{r}$'s parent. Manku [M04] showed that the leaf nodes in a tree with n leaves belong to at most three different levels.
4. First, we carry out a random walk to reach a leaf node $\mathbf{r}$. We then identify the *shallowest hypercubic neighbor*[3] of $\mathbf{r}$, splitting it into two. Adler *et al* [AHKV03] established that the resulting tree has leaf nodes in $\Theta(1)$ different levels.

3.1 A Stochastic Process for Growing Binary Trees

In this paper, we study a generalization of processes 2 and 3 above. We grow the binary tree in a randomized fashion by splitting some leaf node at each step, as follows:

> *We first carry out a random walk down the tree. Let ℓ denote the level of the leaf node encountered. We then carry out $r(\ell) - 1$ additional random walks, to obtain a set of leaf nodes X. For leaf node $\mathbf{x} \in X$, if all nodes in the vicinity of its parent are already split, we retain $\mathbf{x}$ in the set. Otherwise, we replace $\mathbf{x}$ by its parent. Let X' denote the new set thus obtained. Let ℓ' denote the level of the shallowest node in X'. We shrink X' to arrive at set $X'' \subseteq X'$, limited to nodes at level ℓ'. We then choose some $\mathbf{x}'' \in X''$ uniformly at random, and split an un-split node belonging to the vicinity of $\mathbf{x}''$.*

Different combinations of functions r and v result in different processes: Process 1 corresponds to $r(\ell) = v(\ell) = 1$. Process 2 is equivalent to $r(\ell) = c \log n$ and $v(\ell) = 1$. A variation of this process is $r(\ell) = c\ell$ and $v(\ell) = 1$. Process 3 amounts to $r(\ell) = 1$ and $v(\ell) = \lceil c\ell \rceil$. Process 4 amounts to $r(\ell) = 1$

[3] Label the left and right branches of the tree with 0's and 1's respectively. Let the sequence of bits along the path from the root to $\mathbf{r}$ denote the ID of $\mathbf{r}$. A hypercubic neighbor of a leaf node is obtained by flipping a bit in the ID string, and identifying the leaf node with the longest matching prefix of the new string. Please see [AHKV03] for a pictorial definition and more details.

and $v(\ell) = \ell$ where the vicinity is defined to be the hypercubic neighbors of a node (see [AHKV03] for more details). Our interest in this paper lies in the following general condition: $\forall \ell : r(\ell)v(\ell) \geq c\ell$. This includes Processes 2 and 3 as special cases. Our main result is the following theorem:

Theorem 3.1 *For any combination of r and v satisfying $\forall \ell : r(\ell)v(\ell) \geq c\ell$, where c is a suitably large constant, the tree is highly balanced – leaf nodes belong to at most four different levels.*

The motivation for devising and analyzing the generalized stochastic process are three-folds:

1. We are able to demonstrate that the binary tree is balanced for all combinations of $\langle r(\ell), v(\ell)\rangle$ ranging from $\langle 1, c\ell\rangle$ to $\langle c\ell, 1\rangle$. In other words, randomness can be traded off for vicinity-size *smoothly*.
2. The generalized process requires a new proof technique, for which we borrow ideas from [AHKV03]. The proof involves the analysis of an interesting balls-and-bins problem as a sub-problem (Section 2).
3. In Section 4, we design a simple load-balancing scheme for Distributed Hash Tables in peer-to-peer systems. The scheme is based upon the generalized stochastic process. The smooth tradeoff between r and v allows us to identify the optimal number of messages necessary for load balance. We show that $O(\sqrt{\log\log n})$ random walks are sufficient, resulting in a total of $O(\log n/\sqrt{\log\log n})$ messages. The message complexity improves upon all previous schemes [AAA$^+$03, AHKV03, NW03, M04, KR04] for comparable load balance. See Section 4 for more details.

3.2 Proof of Theorem 3.1

Throughout this section, we will assume that $\forall \ell : r(\ell)v(\ell) \geq c\ell$ for a suitably large constant c.

Lemma 3.1 *Assume that the following three claims hold for some constants μ_1 and μ_2:*
1. When $n > \mu_1 2^L$, no leaf is at level L or less, w.h.p., where $2^a < \mu_1 < 2^{a+1}$ for some $a \geq 1$.
2. When $n < \mu_2 2^L$, no leaf is at level L or more, w.h.p., where $2^b < 1/\mu_2 < 2^{b+1}$ for some $b \geq 1$.
3. $\mu_1/\mu_2 < 2^{a+b+1}$.
Then with high probability, the leaves of the tree belong to most $a + b + 1$ different levels.

Proof: Let $2^k \leq n < 2^{k+1}$ for some integer k. There are three cases to consider:
a) $2^k \leq n \leq \mu_1 2^{k+1}$: Leaves belong to levels $[k - a, k + b]$ w.h.p.
b) $\mu_1 2^{k+1} < n < \mu_2^{-1} 2^{k+1}$: Leaves belong to levels $[k - a + 1, k + b]$ w.h.p.
c) $\mu_2^{-1} 2^{k+1} \leq n < 2^{k+1}$: Leaves belong to levels $[k - a + 1, k + b + 1]$ w.h.p.
In any case, the leaves are in at most $a + b + 1$ different levels. □

Consider the structured coupon collection process over a graph with $2^i/v(i)$ cliques, each of size $v(i)$. At each step, $r(i)$ random choices are made. Let $\mathcal{A}_i$ denote the process that terminates when all nodes in the graph have been covered. Let $\mathcal{B}_i$ denote the process that terminates when the first failure occurs, i.e., no new node could be covered. Let $\mathcal{A}^{(\ell)}$ and $\mathcal{B}^{(\ell)}$ denote series of processes $\langle \mathcal{A}_0, \mathcal{A}_1, \ldots, \mathcal{A}_\ell\rangle$ and $\langle \mathcal{B}_0, \mathcal{B}_1, \ldots, \mathcal{B}_\ell\rangle$, respectively.

In the remainder of the section, we will use four constants: α, β, γ and δ. The first two are defined as follows: Let $\alpha 2^i$ denote an upper bound on the number of steps taken by $\mathcal{A}_i$ to terminate, with probability at least $1 - 1/poly(2^i)$ (see Theorem 2.1). Let $\beta 2^i$ denote a lower bound on the number of steps taken by $\mathcal{B}_i$ to terminate, with probability at least $1 - 1/poly(2^i)$. From Theorem 2.2, β can be set to any constant less than half. Constants γ and δ emerge in Lemmas 3.2 and 3.3 respectively. The interplay of all four constants will appear towards the end of this section, when we prove Theorem 3.1.

Lemma 3.2 *$\mathcal{A}^{(k)}$ terminates in at most $\alpha(2+\gamma)2^k$ steps, w.h.p., where γ is an arbitrarily small constant.*

Proof: Let $j = \lceil \log_2 1/\gamma \rceil$, a constant depending upon γ. For process $\mathcal{A}_i$ where $0 \leq i < k - \log_2 k - j$, we allocate $\alpha k 2^i$ steps. The probability that $\mathcal{A}_i$ does not terminate in $\alpha 2^i$ steps is at most $1/poly(2^i)$. Therefore, the probability that $\mathcal{A}_i$ does not terminates in $\alpha k 2^i$ steps is at most $(1/poly(2^i))^k = O(1/poly(2^k))$. The total number of steps we have allocated so far is $\sum_{i=0}^{i=k-\log_2 k-j-1} \alpha k 2^i < \alpha 2^{-j} 2^k \leq \alpha\gamma 2^k$.

We now allocate $\alpha 2^i$ time-steps to each process $\mathcal{A}_i$ where $k - \log_2 k - j \leq i \leq k$. With probability at least $1 - 1/poly(2^i) = 1 - O(1/poly(2^k))$, process $\mathcal{A}_i$ terminates within $\alpha 2^i$ steps. The total number of steps is $\sum_{i=k-\log_2 k-j}^{i=k} \alpha 2^i < \sum_{i=0}^{i=k} \alpha 2^i < \alpha 2^{k+1}$.

The total number of steps is at most $\alpha(2+\gamma)2^k$. □

Lemma 3.3 *$\mathcal{B}^{(k)}$ takes at least $\beta(2-\delta)2^k$ steps to terminate, w.h.p., where δ is an arbitrarily small constant.*

Proof: Let $j = \lceil \log_2 1/\delta \rceil$, a constant depending upon δ. For $k - j - 1 \leq i \leq k$, the probability that process $\mathcal{B}_i$ runs for less than $\beta 2^i$ steps is at most $O(1/poly(2^i)) = O(1/poly(2^k))$. Therefore, the series of processes $\langle \mathcal{B}_{k-j-1}, \mathcal{B}_{k-j}, \ldots, \mathcal{B}_k \rangle$ runs for at least $\sum_{i=k-j-1}^{i=k} \beta 2^i \geq \beta(2-\delta)2^k$ steps w.h.p. Thus $\mathcal{B}^{(k)}$ takes at least $\beta(2-\delta)2^k$ steps to terminate, w.h.p. □

Lemma 3.4 *When $n > \alpha(2+\gamma)2^L$, no leaf is at level L or less, with high probability, where γ is an arbitrarily small constant.*

Proof: We divide the growth of the tree into phases. Phase i is over (and phase $i+1$ starts) when no node at level i is a leaf node. Let T_i denote the time-step at which phase i terminates. To prove that no leaf is at level L or less, we will show that T_L is stochastically dominated by the time taken for $\mathcal{A}^{(L)}$ to terminate. The claim then follows from Lemma 3.2.

Let ℓ denote the level of the leaf node encountered in the first random walk down the tree. In phase i, all leaves are in level i or more. Therefore, $\ell \geq i$. Since function r is monotonically non-decreasing, $r(\ell) \geq r(i)$. Moreover, each vicinity that permits splitting of a leaf at level i, has size exactly $v(i)$, corresponding to a clique in process $\mathcal{A}_i$. Thus it follows that T_L is dominated by the time taken for $\mathcal{A}^{(L)}$ to terminate. □

Lemma 3.5 *When $n < \frac{1}{4}\beta(2-\delta)2^L$, no leaf is at level L or more, with high probability, where δ is an arbitrarily small constant.*

Proof: Consider the following variant of our algorithm: As soon as the *first* leaf at some level ℓ is created, we instantly create all leaf nodes at level $\ell - 1$ as well. This variant grows the tree faster than our original algorithm. Clearly, the variant is dominated by the original algorithm, in terms of the number of steps taken before creating the first leaf at level L. The variant is equivalent to process $\beta^{(L-2)}$, which runs for at least $\frac{1}{4}\beta(2-\delta)2^L$ steps (from Lemma 3.3). □

We are now ready to prove a claim slightly weaker than Theorem 3.1: we will establish that leaf nodes of the tree belong to at most five different levels. Let $\mu_1 = \alpha(2+\gamma)$ and $\mu_2 = \frac{1}{4}\beta(2-\delta)$. From Theorem 2.1, $\alpha = \frac{e}{e-1} + 2e + \epsilon_1 + \epsilon_2$, where ϵ_1 and ϵ_2 are arbitrarily small constants. It is possible to fix these constants so that μ_1 satisfies $2^2 < \mu_1 < 2^3$. Moreover, with a suitable choice of $\beta < 1/2$ (allowed by Theorem 2.2), and sufficiently small δ, we can arrive at a value for μ_2 that satisfies $2^2 < 1/\mu_2 < 2^3$, and $\mu_1/\mu_2 < 2^5$. From Lemma 3.1, it then follows that leaf nodes belong to at most five different levels.

To prove that leaf nodes belong to at most four different levels, as claimed in Theorem 3.1, we need a tighter version of Theorem 2.1, which we now prove.

Theorem 3.2 *With high probability, all bins are full in $\frac{9}{5}n$ trials, for any choice of d and b satisfying $db \geq c \log_2 n$ for a sufficiently large constant c.*

Proof: We treat $d \leq d_0$ (where d_0 is a constant to be defined later) as a special case. For a fixed value of b, the process with $d > 1$ choices dominates the process with a single choice ($d = 1$). A simple application of Chernoff/Hoeffding inequalities in Lemma 2.1 shows that if $\frac{9}{5}n$ balls were placed into $n/b \leq d_0 n/(c \log_2 n)$ bins (of unlimited capacity), each ball choosing a bin uniformly at random (i.e., $d = 1$), then every bin would get at least b balls w.h.p. for a suitably large value of c.

For the rest of the proof, we assume $d > d_0$. We divide the process into two phases as in the proof of Theorem 2.1. The analysis for Phase I is the same as before.

Claim: Phase II terminates within $t_2 = (\frac{1}{5} + \epsilon_2)n$ trials, w.h.p., where ϵ_2 is a small constant.

Proof: As before, for a specific bin C that is non-full at the end of Phase I, the number of balls in C stochastically dominates the random variable $B(t_2, db/2en)$. Choosing $d_0 = 28$ and using $db \geq c \log_2 n$, application of Chernoff/Hoeffding inequalities in Lemma 2.1 yields that, in t_2 trials, C becomes full with probability $1 - o(1/n^2)$. Taking the union bound over all the n bins yields the claim.

We need $(\frac{e}{e-1} + \frac{1}{5} + \epsilon_1 + \epsilon_2)n$ trials w.h.p., where ϵ_1 can be made arbitrarily small, and ϵ_2 can be made small by choosing a large c. The total is less than $\frac{9}{5}n$. □

4 Load Balance in Peer-to-Peer (P2P) Networks

In this Section, we bring out the connection between the stochastic process for growing binary trees (Section 3) and the load-balancing problem in Distributed Hash Tables (DHTs) in P2P networks.

4.1 Distributed Hash Tables: A Brief Summary

DHTs in peer-to-peer networks have witnessed a flurry of research activity since 2001. A DHT is maintained cooperatively, but in a decentralized fashion, by a large number of distributed hosts as follows. Imagine $[0, 1)$ as a circle with unit perimeter. Hosts are allowed to dynamically join the system (the set of participants is not fixed *a priori*). Upon joining, a host is assigned an ID in $[0, 1)$. At any instant, the current set of IDs *partitions* $[0, 1)$ into disjoint sub-intervals – each host is the *manager* of one such sub-interval. A host is connected with its successor and its predecessor along the circle with TCP connections, thereby forming a "ring" of hosts. The ring connections constitute the *short-distance* connections. Each host also makes *long-distance* TCP links with a few other hosts, as a function of its own ID. Taken together, the short- and long-distance TCP connections form the *overlay routing network* that is responsible for routing messages between hosts.

The problem of designing a good DHT routing network has received much attention recently. For our purposes, it suffices to treat the routing network as a black-box with the following property: Using the routing network, it is possible to send a message to the "manager" of a randomly-chosen point in $[0, 1)$ by paying a cost of R messages, with high probability. The earliest DHT routing networks were based on the hypercube and its variants. These can route in $R = \Theta(\log n)$ messages, with only $\Theta(\log n)$ connections per node. Examples of these networks are Chord [SMK+01, GM04], Pastry [RD01] and Tapestry [ZHS+04]. Later papers have shown that it is possible to achieve $R = \Theta(\log n / \log \log n)$ with the same number of connections. Examples of these networks are high-degree de Bruijn networks, as has been observed by several groups [AAA+03, FG03, KK03, LKRG03, NW03], high-degree butterflies [KMXY03], Kleinberg-style randomized butterflies [M03], and several other randomized networks that were analyzed in a recent paper [MNW04] (for example, randomized-Chord [ZGG03, GGG+03], randomized-hypercubes [GGG+03], Symphony [MBR03], skip-graphs [AS03] and SkipNet [HJS+03]).

4.2 Balanced Binary Trees for Decentralized Load Balancing in DHTs

Upon arrival, a new host has to select an ID for itself[4]. DHTs are *decentralized* — there is no global knowledge of the current set of IDs. At any instant, any member of the ring can ascertain the IDs of k adjacent hosts in the ring by paying a cost of $2k$ messages, or it can identify the ID of a host that manages a random number in [0, 1), by paying a cost of R messages. A good ID selection algorithm should enjoy three properties: (a) simplicity and decentralization, (b) low-cost in terms of number of messages, and (c) small variation in partition sizes for load balance among managers. We will quantify variation in partition sizes by defining σ, the *partition balance ratio*, as the ratio between the largest and smallest partition sizes.

The relationship between binary trees (described in Section 3) and host IDs is as follows. Only leaf nodes of the tree correspond to host IDs. The internal nodes of the tree are conceptual. The sequence of 0s and 1s along the path from the root to a leaf node, treated as the binary expansion of a fraction in $[0, 1)$, constitutes the ID of that leaf.

A "random walk down the tree" is equivalent to identifying the manager of a point chosen uniformly at random from the interval $\mathcal{I} = [0, 1)$. We need R messages per random walk. Inspecting the "vicinity" of a leaf node or its parent (see Section 3 for a formal definition of vicinity) is equivalent to identifying whether the corresponding set of IDs along the circle exists or not. To make the inspection low-cost, we stipulate that each host maintain knowledge of its vicinity at all times. Thus when a new host is added to the ring (some leaf node splits in the corresponding tree), all other nodes in its vicinity are informed of the arrival. By choosing $r(\ell) = \sqrt{\log \ell}$ and $v(\ell) = \lceil\!\lceil c\ell/\sqrt{\log \ell}\,\rceil\!\rceil$, we obtain the following theorem:

Theorem 4.1 *A new host needs $\Theta(\log n/\sqrt{\log\log n})$ messages w.h.p. to obtain an ID, where n denotes the current number of hosts and $R = \Theta(\log n/\log\log n)$. The partition balance ratio is $\sigma = \Theta(1)$ w.h.p.*

Proof: From Theorem 3.1, the tree has leaves in at most four different levels w.h.p. With n leaf nodes, the level of any leaf is $\Theta(\log n)$ w.h.p. With $r(\ell) = \sqrt{\log \ell}$, the number of random walks down the tree is $\Theta(\sqrt{\log\log n})$. With $R = \Theta(\log n/\log\log n)$ messages per random walk, the total number of messages is $\Theta(\log n/\sqrt{\log\log n})$.

Whenever a new host is inserted, all other members of the vicinity it belongs to, are informed of its existence. Informing all members of a vicinity of size $v(\ell)$ requires at most $v(\ell)$ messages (by using only the short-distance "ring"-connections). With n leaf nodes, the level of a leaf node is $\Theta(\log n)$ w.h.p. With $v(\ell) = \lceil\!\lceil c\ell/\sqrt{\log \ell}\,\rceil\!\rceil$, a vicinity has $\Theta(\log n/\sqrt{\log\log n})$ nodes, requiring as many messages.

Since leaf nodes are in $\Theta(1)$ different levels, $\sigma = \Theta(1)$ w.h.p. □

4.3 Lower Bound on σ with Balanced Binary Trees

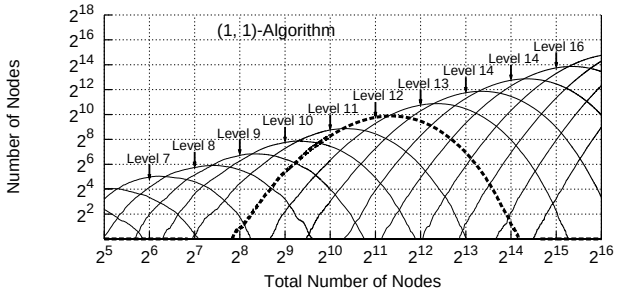
How small a value of σ can possibly be realized in a decentralized setting? Ideally, we would like to have a distributed algorithm which ensures that the number of bits in any ID is either $\lfloor \log_2 n \rfloor$ or $\lceil \log_2 n \rceil$, when the current number of hosts is n. This goal seems unattainable for a decentralized algorithm because of the following intuition. When $n = 2^k - 1$, all leaf nodes in the corresponding tree should ideally be in levels $\{k-1, k\}$. However, with $n = 2^k + 1$, all leaf nodes should be in levels $\{k, k+1\}$. Therefore, a decentralized algorithm is likely to have leaves in at least three different levels, especially when n is close to a power of two[5].

[4]It is customarily assumed in DHT design that the new host "knows" one existing member of the ring at the outset.

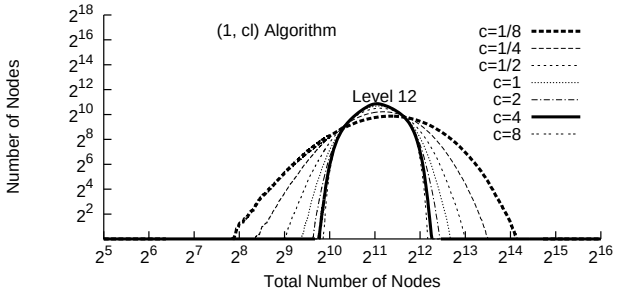
[5]A formal proof requires a precise *definition* of the notion of decentralization.

4.4 Experimental Results

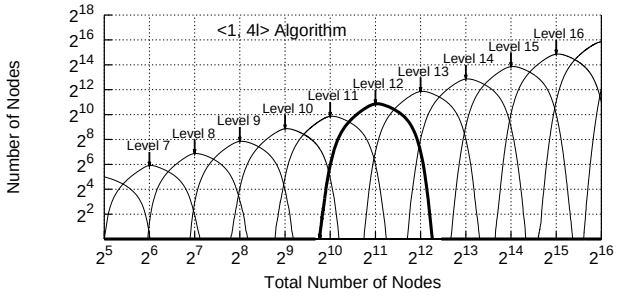
Figure 1 studies various $\langle r, v\rangle$ schemes for growing binary trees. With $\langle r, v\rangle = \langle 1, 1\rangle$, host IDs belong to as many as six different levels when $n = 2^{11}$. With $\langle r, v\rangle = \langle 1, 4\ell\rangle$, host IDs are in only three different levels. Thus 4 appears to be a reasonable value for constant c in the constraint: $\forall \ell : r(\ell)v(\ell) \geq c\ell$. Finally, five random walks are sufficient to obtain 3-level trees, when the number of hosts is $n = 2^{16}$. In terms of messages, this is superior to either of the two extremes: $\langle 1, c\ell\rangle$ and $\langle c\ell, 1\rangle$.



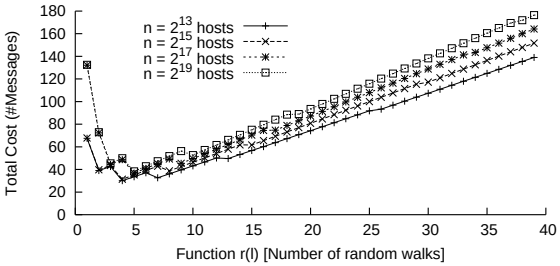
(a) Distribution of host IDs using $\langle 1, 1\rangle$ scheme.



(b) The effect of increasing c in $\langle 1, c\ell\rangle$ schemes.



(c) Distribution of host IDs with $\langle 1, 4\ell\rangle$ scheme.



(d) Total cost for varying number of random walks.

Figure 1: *In (a), the dotted curve shows the number of hosts with 12-bit IDs, as the total number of hosts increases over time, with the $\langle 1, 1\rangle$ scheme. The number of curves intersecting a vertical line equals the number of different levels at which leaf nodes exist, for that n. In (c), using $c = 4$, the $\langle 1, c\ell\rangle$ scheme results in 3-level trees. In (d), the four curves correspond to $n = 2^\ell$ hosts for $\ell = 13, 15, 17$ and 19 respectively. The total cost is $Rx + y$, where x is the number of random walks, $y = \lceil\!\lceil 4\ell/x \rceil\!\rceil$, and $R = \ell/\log_2 \ell$ hops on average.*

4.5 Previous DHT Load Balancing Proposals

Early DHT designs allowed each host to independently choose a number in $[0, 1)$ uniformly at random [SMK+01, ZHS+04, RD01, RFHK01, FG03, KK03, MBR03, MNR02, HJS+03]. Such an algorithm is decentralized and requires zero messages to select an ID. However, $\sigma = \Omega(\log^2 n)$ [NW03] w.h.p., where σ denotes the *partition balance ratio*, defined as the ratio between the largest and the smallest partition sizes. King and Saia [KS04] recently established that $\sigma = \Theta(n \log n)$ w.h.p.

If each host chooses a random number in $[0, 1)$ and *splits* the partition the number falls into, σ diminishes to $\Theta(\log n)$ [AHKV03, NW03]. Further improvement is possible. If each host creates $\Omega(\log n)$ *virtual IDs* [DKK+01], σ reduces to $O(1)$. However, the number of overlay connections per host gets amplified by a factor of $\Omega(\log n)$ – this is costly because higher degree overlay networks require more resources for maintenance.

Two different approaches for ID management have recently been proposed, each of which guarantees $\sigma = \Theta(1)$ with only one ID per host. The first approach [NW03, AAA+03, KR04, M04] is overlay-independent while the second is overlay-dependent [AHKV03]. Naor and Wieder [NW03] and Abraham

et al [AAA+03] proposed that a new host should choose $\Theta(\log n)$ random points from $[0,1)$, identify the managers of these points and split the largest manager into two. Karger and Ruhl [KR04] proposed an elegant variation on the idea that supports departures as well, albeit at the cost of *re-assignment* of IDs of at most $O(\log \log n)$ hosts w.h.p. per arrival and departure. Adler *et al* [AHKV03] analyzed an overlay-dependent scheme that is specific to hypercubes. The idea is to identify the manager of a random point in $[0,1)$, probe other managers it has established overlay connections with, and to split the largest of these managers into two. A scheme for handling departures exists, but it has not yielded to formal analysis yet. The idea in [AHKV03] had earlier been proposed as a heuristic in an early DHT paper [RFHK01]. Manku [M04] recently established that $\sigma \leq 4$ for the following scheme: a new host first randomly identifies the manager of a random number in $[0,1)$, inspects $\Theta(\log n)$ managers in its "vicinity" and splits the largest manager. Departures are handled similarly and cause at most one host ID to be re-assigned. The message complexity for the schemes outlined above is either $\Theta(\log^2 n/\log\log n)$ messages [AAA+03, NW03, KR04] or $\Theta(\log n)$ messages [AHKV03, M04], for networks with $R = \Theta(\log n/\log\log n)$.

Two additional approaches to load-balancing are as follows. Byers *et al* [BCM03] suggest that a node should continue to choose a random number in $[0,1)$ as its ID. However, an object should be stored at one out of several possible locations (determined by multiple hash-values of the object-name). A drawback of this idea is the overhead associated with multiple probes necessary when storing and retrieving objects. Godfrey *et al* [GLS+04] take a systems approach, identifying the *run-time* loads on various nodes. They propose heuristics for re-distributing objects between pairs of lightly- and heavily-loaded nodes.

5 Future Directions

a) Our load-balancing algorithm does not address host departures. Only two known algorithms handle departures [KR04,M04]. Simulations show that a simple variation of the insertion algorithm maintains 3-level trees: *"A departed host is replaced by the deepest leaf in the union of vicinities probed."*

b) If we could establish Theorem 2.2 for any $\beta \in (\frac{1}{2}, 1)$, we could show that leaf nodes belong to at most three different levels, which we believe is optimal.

c) It would be interesting to analyze structured coupon collection over cliques when $bd \not\geq c \log n$, and to explore the impact of multiple choices ($d \geq 2$) over general graphs.

References

[A04] N Alon. Problems and results in extremal combinatorics, II. http://www.math.tau.ac.il/∼nogaa/PDFFS/publications.html, 2004.

[AAA+03] I Abraham, B Awerbuch, Y Azar, Y Bartal, D Malkhi, and E Pavlov. A generic scheme for building overlay networks in adversarial scenarios. *Proc. Intl. Parallel and Distributed Processing Symposium (IPDPS 2003)*, p. ??–??, 2003.

[ABKU99] Y Azar, A Z Broder, A R Karlin, and E Upfal. Balanced allocations. *SIAM Journal of Computing*, 29(1):180–200, 1999.

[AHKV03] M Adler, E Halperin, R M Karp, and V V Vazirani. A stochastic process on the hypercube with applications to peer-to-peer networks. *Proc. 35nd ACM Symposium on Theory of Computing (STOC 2003)*, p. 575–584, 2003.

[AS03] J Aspnes and G Shah. Skip graphs. *Proc. 14th ACM-SIAM Symp. on Discrete Algorithms (SODA 2003)*, p. 384–393, 2003.

[BCM03] J Byers, J Considine, and M Mitzenmacher. Simple load balancing for distributed hash tables. *Proc. 2nd Intl. Workshop on Peer-to-Peer Systems (IPTPS 2003)*, 2003.

[C52] H Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Annals of Mathmematical Statistics*, 23(4):493–509, 1952.

[DKK+01] F Dabek, M F Kaashoek, D Karger, R Morris, and I Stoica. Wide-area cooperative storage with CFS. *Proc. 18th ACM Symposium on Operating Systems Principles (SOSP 2001)*, p. 202–215, 2001.

[DR98] D P Dubhashi and D Ranjan. Balls and bins: A study in negative dependence. *Random Structures and Algorithms*, 13(2):99–124, 1998.

[FG03] P Fraigniaud and P Gauron. (brief announcement) An overview of the content-addressable network D2B. *Proc 22nd ACM Symposium on Principles of Distributed Computing (PODC 2003)*, p. 151–151, 2003.

[G81] G H Gonnet. Expected length of the longest probe sequence in hash code searching. *Journal of the ACM*, 28(2):289–304, 1981.

[GGG+03] K P Gummadi, R Gummadi, S D Gribble, S Ratnasamy, S Shenker, and I Stoica. The impact of DHT routing geometry on resilience and proximity. *Proc. ACM SIGCOMM 2003*, p. 381–394, 2003.

[GLS+04] B Godfrey, K Lakshminarayanan, S Surana, R M Karp, and I Stoica. Load balancing in dynamic structured P2P systems. *Proc. IEEE INFOCOM 2004*, p. ??–??, 2004.

[GM04] P Ganesan and G S Manku. Optimal routing in Chord. *Proc. 15th ACM-SIAM Symp. on Discrete Algorithms (SODA 2004)*, p. 169–178, 2004.

[H63] W Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[HJS+03] N J A Harvey, M Jones, S Saroiu, M Theimer, and A Wolman. SkipNet: A scalable overlay network with practical locality properties. *Proc. 4th USENIX Symposium on Internet Technologies and Systems (USITS 2003)*, 2003.

[JK77] N L Johnson and S Kotz. *Urn Models and their Applications: An Approach to Modern Discrete Probability Theory*. John Wiley and Sons, 1977.

[KK03] M F Kaashoek and D R Karger. Koorde: A simple degree-optimal hash table. *Proc. 2nd Intl. Workshop on Peer-to-Peer Systems (IPTPS 2003)*, p. 98–107, 2003.

[KMXY03] A Kumar, S Merugu, J J Xu, and X Yu. Ulysses: A robust, low-diameter, low-latency peer-to-peer network. *Proc. 11th IEEE International Conference on Network Protocols (ICNP 2003)*, p. ???–???, 2003.

[KR04] D R Karger and M Ruhl. Simple efficient load balancing algorithms for peer-to-peer systems. *Proc. 16th ACM Symp. on Parallelism in Algorithms and Architectures (SPAA 2004)*, p. ??–??, 2004.

[KS04] V King and J Saia. Choosing a random peer. *Proc. 23rd ACM Symp. on Principles of Distributed Computing (PODC 2004)*, p. ??–??, 2004.

[KSC78] V F Kolchin, B A Sevast'yanov, and V P Chistyakov. *Random Allocations*. V H Winston & Sons, 1978.

[LKRG03] D Loguinov, A Kumar, V Rai, and S Ganesh. Graph-theoretic analysis of structured peer-to-peer systems: Routing distance and fault resilience. *Proc. ACM SIGCOMM 2003*, p. 395–406, 2003.

[M96] M Mitzenmacher. *The Power of Two Choices in Randomized Load Balancing*. PhD dissertation, University of California at Berkeley, Department of Computer Science, 1996.

[M03] G S Manku. Routing networks for distributed hash tables. *Proc. 22nd ACM Symp. on Principles of Distributed Computing (PODC 2003)*, p. 133–142, 2003.

[M04] G S Manku. Balanced binary trees for ID management and load balance in distributed hash tables. *Proc. 23rd ACM Symp. on Principles of Distributed Computing (PODC 2004)*, p. ??–??, 2004.

[MBR03] G S Manku, M Bawa, and P Raghavan. Symphony: Distributed hashing in a small world. *Proc. 4th USENIX Symposium on Internet Technologies and Systems (USITS 2003)*, p. 127–140, 2003.

[MNR02] D Malkhi, M Naor, and D Ratajczak. Viceroy: A scalable and dynamic emulation of the butterfly. *Proc 21st ACM Symposium on Principles of Distributed Computing (PODC 2002)*, p. 183–192, 2002.

[MNW04] G S Manku, M Naor, and U Wieder. Know thy neighbour's neighbour: The power of lookahead in randomized P2P networks. *Proc. 36th ACM Symposium on Theory of Computing (STOC 2004)*, p. 54–63, 2004.

[MR95] R Motwani and P Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[MRS01] M Mitzenmacher, A W Richa, and R Sitaraman. The power of two random choices: A survey of techniques and results. *Handbook of Randomized Computing (Vol 1)*. Kluwer Academic Press, 2001.

[NW03] M Naor and U Wieder. Novel architectures for P2P applications: The continuous-discrete approach. *Proc. 15th ACM Symp. on Parallelism in Algorithms and Architectures (SPAA 2003)*, p. 50–59, 2003.

[RD01] A I T Rowstron and P Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001)*, p. 329–350, 2001.

[RFHK01] S Ratnasamy, P Francis, M Handley, and R M Karp. A scalable Content-Addressable Network. *Proc. ACM SIGCOMM 2001*, p. 161–172, 2001.

[RS98] M Raab and A Steger. Balls into bins – a simple and tight analysis. *Randomization and Approximation Techniques in Computer Science (RANDOM 1998), Lecture Notes in Computer Science 1518*, p. 159–170, 1998.

[SMK+01] I Stoica, R Morris, D Karger, M F Kaashoek, and H Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. *Proc. ACM SIGCOMM 2001*, p. 149–160, 2001.

[ZGG03] H Zhang, A Goel, and R Govindan. Incrementally improving lookup latency in distributed hash table systems. *ACM SIGMETRICS 2003*, p. 114–125, 2003.

[ZHS+04] B Y Zhao, L Huang, J Stribling, S C Rhea, A D Joseph, and J D Kubiatowicz. Tapestry: A resilient global-scale overlay for service deployment. *IEEE Journal on Selected Areas in Communications*, 22(1), 2004.