Decentralized Algorithms using both Local and Random Probes for P2P Load Balancing

Krishnaram Kenthapadi*
Stanford University
kngk@cs.stanford.edu

Gurmeet Singh Manku†
Google Inc.
manku@google.com

ABSTRACT

We study randomized algorithms for placing a sequence of n nodes on a circle with unit perimeter. Nodes divide the circle into disjoint arcs. We desire that a newly-arrived node (which is oblivious of its index in the sequence) choose its position on the circle by learning the positions of as few existing nodes as possible. At the same time, we desire that that the variation in arc-lengths be small. To this end, we propose a new algorithm that works as follows: The kth node chooses r random points on the circle, inspects the sizes of v arcs in the vicinity of each random point, and places itself at the mid-point of the largest arc encountered. We show that for any combination of r and v satisfying $rv \geq c \log k$, where c is a small constant, the ratio of the largest to the smallest arc-length is at most eight w.h.p., for an arbitrarily long sequence of n nodes. This strategy of node placement underlies a novel decentralized load-balancing algorithm that we propose for Distributed Hash Tables (DHTs) in peer-to-peer environments.

Underlying the analysis of our algorithm is *Structured Coupon Collection* over n/b disjoint cliques with b nodes per clique, for any $n, b \geq 1$. 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 succeeds 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 with high probability.

*Supported in part by NSF Grants EIA-0137761 and ITR-0331640, and a grant from SNRC.

†Most of this work was done at Stanford University, supported in part by NSF Grant EIA-0137761 and grants from SNRC and Veritas.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SPAA'05, July 18–20, 2005, Las Vegas, Nevada, USA
Copyright 2005 ACM 1-58113-986-1/05/0007 ...$5.00.

Categories and Subject Descriptors

C.2.4 [**Distributed Systems**]: Distributed Applications [Load Balancing]; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Non-numerical Algorithms and Problems

General Terms

Algorithms, Theory

Keywords

Coupon collection, bins and balls, load balancing, cliques, binary trees, decentralized algorithms, peer to peer systems, distributed hash tables.

1. INTRODUCTION

Consider a sequence of nodes. Each node has to be placed somewhere on the circumference of a circle. Nodes divide the circumference into disjoint arcs. We denote the ratio between the largest and the smallest arc by σ. An interesting tradeoff emerges between σ and the amount of knowledge of existing node-positions available to the ith node in the sequence. If the ith node has complete knowledge of all existing node-positions, then it can be placed such that $\sigma \leq 2$ always. If the ith node has no knowledge of existing node-positions (and is oblivious of its own index in the sequence), we could assign it a position uniformly at random – this results in $\sigma = \Theta(n \log n)$ with high probability[1] [26].

We study the relationship between σ and the amount of knowledge available to the ith node. Knowledge of existing node-positions is gained by two means:

- ✧ Random Probe: A point on the circumference is chosen uniformly at random. The positions of the two nodes which sandwich that point are then learnt.
- ✧ Local Probe: The positions of v nodes adjacent to a known node along the circle are learnt. We call this a "local probe of size v".

The ith node assigns itself that position which splits the largest arc encountered by random and local probes. It is natural to ask: *What combinations of random and local probes are sufficient to guarantee $\sigma = \Theta(1)$?*

Our key result is that with r random probes, each followed by a local probe of size v, $\sigma \leq 8$ with high probability, as long as $rv \geq c \log i$, where c is a small constant, and i denotes the

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

index of the node in the sequence. (Remark: since a node is oblivious of i, its index in the sequence, how do we enforce the constraint $rv \geq c \log i$ when placing the ith node? We *estimate* $\log i$ from the first random probe).

Three previous results can be stated in terms of local and random probes. With one random probe, followed by splitting the arc encountered by the random probe, $\sigma = \Theta(\log n)$ for a sequence of n nodes [2, 40]. Naor and Wieder [40] and Abraham *et al* [1] showed that with $O(\log n)$ random probes (followed by no local probes), $\sigma = \Theta(1)$ (it is assumed that the ith node knows n, the length of the sequence). Manku [32] showed that with one random probe, followed by a local probe of size $O(\log i)$ (where $\log i$ is estimated), $\sigma \leq 4$. In contrast, our result shows that there is a smooth tradeoff between random and local probes. Moreover, our proof technique is quite different from the techniques in [1, 32, 40].

There is a connection between the problem of assigning positions to nodes on the circumference of a circle, and "balanced binary trees" which grow in size stochastically. We explore this connection in the next sub-section.

1.1 Balanced Binary Trees

Consider a binary tree in which we repeatedly walk down the tree (starting at the root), choosing between the left- and the right-child with equal probability. We then split the leaf node encountered. The n leaves of the resulting tree belong to $\Theta(\log \log n)$ levels [2,40]. If we split the shallowest leaf node encountered by $O(\log n)$ random walks, the leaves belong to $\Theta(1)$ different levels [1, 40]. The same property holds for the following strategy too: We take one random walk to reach some leaf node at level ℓ, then start traveling up the tree until the size of the sub-tree exceeds $c\ell$ for some constant c, then split the shallowest leaf in that sub-tree [32].

In this paper, we study the following strategy: We take a random walk down the tree to reach a node at level ℓ. We then take $r-1$ additional random walks. For each leaf node encountered, we start moving up the tree until the size of the sub-tree exceeds v. Among the r sub-trees thus encountered, we split the shallowest leaf-node (breaking ties arbitrarily). We show that as long as $rv \geq c\ell$ for a suitably large constant c, leaves belong to at most four different levels.

Our proof technique borrows ideas developed by Adler *et al* [2]. We divide the tree level-by-level into layers. We then define a coupon collection problem over each layer and prove lower- and upper- bounds on its running-time. These bounds help us bound the levels to which leaves belong. In the next sub-section, we describe the coupon collection problem we study and related prior work.

1.2 Structured Coupon Collection

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 [39], for example). If we have to collect b copies each of n/b coupons where b is a constant, then the number of trials needed to collect all copies is sharply concentrated around $\frac{n}{b}(\ln \frac{n}{b} + (b-1) \ln \ln \frac{n}{b})$ [39].

One generalization of the standard coupon collection problem is multiplicity of 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* [2] call this process *Structured Coupon Collection* over graphs. They establish that with w.h.p., $O(n)$ 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 $\Delta = \Omega(\log n)$. Alon [3] 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. In this paper, we study the following problem:

Definition (Structured Coupon Collection over Cliques). *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.*

Our main results are that for any combination of b and d satisfying $bd \geq c \log n$ for some suitably large constant c, the following hold w.h.p.: (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.

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 classic balls-and-bins problem involves bins with infinite capacity and $d = 1$ (see Johnson and Kotz [22] or Kolchin *et al* [27]). 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 [18], Mitzenmacher [37] and Raab and Steger [41]). For the case $d \geq 2$, a breakthrough was achieved by Azar *et al* [6] 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* [38]. 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 cliques was motivated by the desire to analyze the growth of stochastically growing binary trees (§1.1), which in turn were motivated by load balancing concerns arising in Distributed Hash Tables (DHTs). We review DHTs in the next sub-section.

1.3 Distributed Hash Tables

Distributed hash tables (DHTs) in peer-to-peer (P2P) environments have witnessed a surge of interest recently. The goal is to build a giant hash table over a large number of hosts spanning the Internet. The scale of the system encourages *decentralized* algorithms which do not require global knowledge. There are two fundamental problems:

- ✡ Load Balancing: How do participating hosts partition the hash table among themselves?
- ✡ Routing: How is the mapping between partitions and IP addresses of hosts maintained?

Algorithm	σ	*Message Cost*	*Departures?*	*Notes*
Random Binary Tree	$\Theta(\log n)$	R	No	
Adler *et al* [2]	$\Theta(1)$	$\Theta(R + \log n)$	No	Only for hypercubic routing networks.
Naor & Wieder [40]	$\Theta(1)$	$\Theta(R \log n)$	(Yes)	No proof for departures.
Abraham *et al* [1]	$\Theta(1)$	$\Theta(R \log n)$	No	
Karger & Ruhl [25]	$\Theta(1)$	$\Theta(R \log n)$	Yes	$O(\log \log n)$ re-assignments per arrival/departure.
Giakkoupis & Hadzilacos [15]	4	$\Theta(R \log n)$	Yes	1 re-assignment per departure.
Manku [32]	4	$\Theta(R + \log n)$	Yes	1 re-assignment per departure.
	$1 + \epsilon$, $\epsilon \in (0, 1]$	$\Theta(R + \frac{1}{\epsilon^2} \log n)$	Yes	$O(1/\epsilon)$ re-assignments per departure.
Our algorithms	**8**	$\mathbf{\Theta(rR + v)}$ **(any** $\mathbf{rv \geq c \log n}$**)**	**No**	**Smooth tradeoff between r and v.**

Table 1: *R denotes the average number of messages required by the overlay routing layer. Typically, $R = \Theta(\log n)$ or $R = \Theta(\log n / \log \log n)$, w.h.p. σ denotes the ratio of the largest partition to the smallest partition. For the scheme by Adler et al [2], $R = \Theta(\log n)$ since it is tied to a specific overlay routing topology (the hypercube). When $R = \Theta(\log n / \log \log n)$, the optimal message cost using our algorithm is $\Theta(\log n / \sqrt{\log \log n})$.*

Load Balancing: Each participant is assigned an ID in the unit interval $[0, 1)$. It is convenient to imagine the interval as a circle with unit perimeter. Hosts join the system dynamically over time. 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. We define σ, the *partition balance ratio*, to be the ratio between the sizes of the largest and the smallest partitions.

Routing: Routing networks for DHTs have received much attention [1, 5, 12, 14, 19, 20, 23, 28–31, 33–36, 40, 44–47]. Basically, hosts establish TCP connections among themselves as a function of their IDs. These connections are of two kinds: *short-distance* links and *long-distance*. The short-distance links connect nodes with adjacent IDs in $[0, 1)$, thereby forming a *ring*[2] (Chord [45], D2B [12], Koorde [23], SkipNet [20], Symphony [34], Viceroy [30], Ulysses [28], and others [1, 31, 40]). Some DHT routing networks do not form a complete ring (examples: Pastry [44], Tapestry [47], Bamboo [43] and Kademlia [36]). However, almost all of the ring connections do exist in these networks because nodes in the circular ID space $[0, 1)$ are divided into disjoint clusters[3] and nodes within a cluster form a clique. Apart from the ring connections (which constitute the *short-distance* links), each host also makes *long-distance* links with a few other hosts whose IDs are far away along the circle. Taken together, the short-distance and the long-distance links ensure fault-tolerant message delivery in a small number of hops.

Random and Local Probes: We treat the routing network as a black-box that supports two operations: local probes and random probes, as outlined in §1. A local probe of size k costs $2k$ messages. A random probe costs R messages. R depends on the routing network. For networks based on hypercubes, $R = \Theta(\log n)$ when each node makes $\Theta(\log n)$ links per node (examples: Chord [14, 45], Pastry [44] and Tapestry [47]). Concerns for "physical network-proximity" (see Gummadi *et al* [19] for a discussion of this issue) have resulted in the creation of randomized variants of these networks — randomized-Chord [19, 46] and randomized-hypercubes [8, 19]. For these networks, $R = \Theta(\log n / \log \log n)$ when "neighbor-of-neighbors" are used for greedy routing [35]. In fact, $R = \Theta(\log n / \log \log n)$ holds for a variety of networks with $\Theta(\log n)$ links per node: high-degree de Bruijn networks (an observation made by many groups [1, 12, 23, 29, 40]), high-degree butterflies [28], Kleinberg-style randomized butterflies [31], and several randomized networks analyzed by Manku, Naor and Wieder [35] — these include randomized-Chord [19, 46], randomized-hypercubes [19], Symphony [34], skip-graphs [5] and SkipNet [20].

The ID Assignment Problem: Upon arrival, a new host has to be assigned an ID. It is customarily assumed in DHT design that the new host "knows" one existing member of the ring at the outset. 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 (a local probe), 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 random probe). 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.

1.4 Load Balance and ID Assignment in DHTs

Early DHT designs allowed each host to independently choose a number in $[0, 1)$ uniformly at random [12, 20, 23, 30, 34, 42, 44, 45, 47]. Such an algorithm is decentralized and requires zero messages to select an ID. However, $\sigma = \Omega(\log^2 n)$ [40] w.h.p. King and Saia [26] 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 $\Theta(\log n)$ [2, 40]. Further improvement is possible by assigning multiple *virtual* IDs per host (CFS [10] uses this idea). With $\Theta(\log n)$ IDs per host, $\sigma = \Theta(1)$ (see the consistent hashing paper [24], for example). However, by using virtual IDs, 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.

It is natural to ask whether a highly balanced distribution

[2]In practice, a host makes connection with f hosts adjacent to it along the circle for a *fault-tolerant* ring, where f is a small number (see the Chord [45] paper, for example). We will ignore this issue, however, as it does not affect the asymptotes of our algorithms.

[3]In Pastry, these are called *leafsets* with expected size 16.

of partitions can be achieved with only one ID per host if we allow a newly-arrived host to first *learn* the IDs of a few existing hosts, and then choose an ID for itself. Indeed, $\sigma = \Theta(1)$ is possible, as described below.

Two different approaches for ID management have recently emerged. Both guarantee $\sigma = \Theta(1)$ with only one ID per host. The first approach [1, 25, 32, 40] is overlay-independent while the second is overlay-dependent [2]. Naor and Wieder [40] and Abraham *et al* [1] 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. An extension proposed by Karger and Ruhl [25] handles node departures too, albeit at the cost of $O(\log \log n)$ *re-assignment* of IDs of at per arrival and departure, w.h.p. The deletion scheme by Giakkoupis and Hadzilacos [15] necessitates only $\Theta(1)$ re-assignments. Adler *et al* [2] 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 [2] had earlier been proposed as a heuristic in [42]. Manku [32] 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 [1,25,40] or $\Theta(\log n)$ messages [2, 32], for networks with $R = \Theta(\log n/ \log \log n)$.

For load balancing over heterogeneous nodes, see [15, 17].

1.5 Other DHT Load Balancing Problems

Byers *et al* [7] suggest a variant of the power-of-two choices paradigm (see Mitzenmacher *et al* [38] for a survey). In their scheme, nodes continue to choose random numbers in $[0, 1)$ as their IDs. However, an object is 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* [16] 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.

Load balance for range-queries over an ordered list of data elements has also received attention. The goal is to divide a sorted list of elements among a fixed number of blocks in the face of adversarial insertions and deletions of elements, such that the ratio of the largest and the smallest block is $\Theta(1)$. Two operations are allowed: (a) two adjacent blocks are re-balanced, or (b) two adjacent blocks are fused into one block while a large block is split into two. The cost associated with a fusion or a split is proportional to the number of data elements that have to *move*. Ganesan *et al* [13] present an efficient deterministic algorithm that guarantees a constant number of moves per data element, amortized over the lifetime of the data structure. If each block is assigned to one machine in a DHT, and if a skip-graph [5] is maintained over the starting keys of blocks, we obtain an efficient load-balanced range-queriable data structure. The scheme by Ganesan *et al* assumes that the largest (and the smallest) block can be efficiently discovered. It is not clear how this operation can be implemented in a *decentralized* fashion without causing congestion in a DHT. Karger and Ruhl [25] and Aspnes *et al* [4] present efficient congestion-free algorithms that match large blocks with small blocks. Their algorithms are randomized, decentralized and handle a dynamic number of blocks.

1.6 Summary of Results

1. Improvements over Previous Work: Existing DHT load balancing algorithms can be viewed in terms of random probes and local probes (see §2). However, they are either designed for a specific routing network [2] (the hypercube), or use only one random probe ($r = 1$) [2,32], or are limited to local probes of size $v = 1$ [1,25,40]. In contrast, our algorithm is oblivious of the long-distance links of the routing network, making it applicable to DHT designs in which all or almost-all of the short-distance links (along the ring) are present. Moreover, the algorithm enables a smooth tradeoff between r (the number of random probes) and v (the size of local probes).
2. Optimal Combination of Random and Local Probes: The tradeoff between r (the number of random probes) and v (the size of local probes) can be exploited to derive the optimal number of messages for a specific routing network. With $R = \Theta(\log n/ \log \log n)$ messages per random probe, the optimal number of random probes is $r = \Theta(\sqrt{\log \log n})$, for a total of $\Theta(\log n/\sqrt{\log \log n})$ messages. In practice, the gains over an algorithm with $r = 1$ or $v = 1$ are substantial, as we show experimentally in §5. The schemes corresponding to $r = 1$ and $v = 1$ constitute the currently best-known algorithms for load-balancing.
3. Structured Coupon Collection over Cliques: Our algorithm constructs binary-trees which are highly balanced – leaf nodes belong to at most four different levels. Deriving bounds on the height of the deepest and the shallowest leaves requires analysis of an interesting coupon-collection problem defined in §1.2.

2. BALANCED BINARY TREES

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. Some definitions:

Level and vicinity: The *level* of a node is the length of the path from the root to that node. The root has level 0. 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 $[\![x]\!]$, r *and* v: Let $[\![x]\!] \equiv 2^k$ 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.

A summary of results established in earlier papers (n denotes the current number of leaf nodes):

1. At each step, we perform one random walk and split the leaf node encountered. Adler *et al* [2] and Naor and Wieder [40] show that leaf nodes belong to $\Theta(\log \log n)$ different levels w.h.p.
2. At each step, we perform $c \log n$ random walks down

the tree and split the shallowest leaf node encountered. Abraham *et al* [1] showed that after n steps, the leaves lie in $\Theta(1)$ different levels w.h.p. Naor and Wieder [40] analyze a similar stochastic process in which we *estimate* $\log n$ before performing the random walks.

3. For a node at level l, let $v(\ell) = \lceil\!\lceil c\ell \rceil\!\rceil$, where c is a suitably-large constant. First, we perform 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 [32] showed that the leaf nodes in a tree with n leaves belong to at most three different levels w.h.p.
4. Perform a random walk to reach a leaf node $\mathbf{r}$. Then identify the *shallowest hypercubic neighbor*[4] of $\mathbf{r}$, splitting it into two. Adler *et al* [2] established that the resulting tree has leaf nodes in $\Theta(1)$ different levels w.h.p.

Remark: Karger and Ruhl [25] extend the idea underlying Process 2 to handle *both* arrivals and departures, albeit at the cost of $O(\log\log n)$ re-assignments of existing IDs per arrivals/departures. Subsequently, Giakkoupis and Hadzilacos [15] proposed a simple deletion scheme that guarantees $\sigma = 4$ at the cost of only $\Theta(1)$ re-assignments.

The algorithms described above either use just one random probe, or limit themselves to local probes of size one. Moreover, the scheme by Adler *et al* is not designed for arbitrary routing networks. In this paper, we study a natural combination of the two kinds of probes: r random probes, each of size v, where $rv \geq c\log n$ (there is no global knowledge of n though). Formally, we grow the binary tree in a randomized fashion by splitting some leaf node at each step:

> *We first perform a random walk down the tree. Let ℓ denote the level of the leaf node encountered. We then perform $r(\ell)-1$ additional random walks, to obtain a set of leaf nodes X. For each 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 $v(\ell) = \lceil\!\lceil c\ell \rceil\!\rceil$. Process 4 amounts to $r(\ell) = 1$ and $v(\ell) = \ell$ where the vicinity is defined to be the hypercubic neighbors of a node (see [2] 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 2.1. *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.*

[4]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 [2] for a pictorial definition and more details.

PROOF. The proof, described in §3 and §4, borrows ideas from [2] and works as follows: We break the tree into layers (all nodes at the same level belong to one layer). At each layer, we study a coupon-collection problem over disjoint cliques defined over nodes at that layer (Section 3). Using these results, we bound the depth of the shallowest and the deepest leaf nodes (Section 4). □

The relationship between binary trees and host IDs is as follows. Only leaf nodes of the tree correspond to 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 2 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 2.2. *A new host requires $\Theta(\log n/\sqrt{\log\log n})$ messages to obtain an ID w.h.p., where n denotes the current number of hosts and $R = \Theta(\log n/\log\log n)$. The partition balance ratio is $\sigma \leq 8$ w.h.p.*

PROOF. From Theorem 2.1, the tree has leaves in at most four different levels w.h.p. With n leaf nodes, the level of any leaf node 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 $v(\ell) = \lceil\!\lceil c\ell/\sqrt{\log \ell} \rceil\!\rceil$, each vicinity has $\Theta(\log n/\sqrt{\log\log n})$ nodes, requiring as many messages.

Since leaf nodes are in at most 4 different levels, $\sigma = 8$ w.h.p. □

Remarks: Experimental results show that leaf nodes actually belong to at most 3 different levels; therefore, $\sigma \leq 4$ in practice. It is natural to ask: *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].

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

3. STRUCTURED COUPON COLLECTION OVER CLIQUES

Problem definition: 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.

Our main results are that for any combination of b and d satisfying $bd \geq c \log n$ for some suitably large constant c, the following hold w.h.p.: (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.

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 [9] and Hoeffding [21]. The second lemma helps us derive tail bounds for dependent binary random variables under certain conditions.

LEMMA 3.1. *Let Z denote a random variable with a binomial distribution $Z \sim B(n,p)$.*

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

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

LEMMA 3.2. *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*

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

THEOREM 3.1. *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. For all $x > 1$, $(1 - \frac{1}{x})^x < e^{-1} < (1 - \frac{1}{x})^{x-1}$.

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 ith trial. Let ω_i denote the random choices available to the ith ball. Then, $Pr[Z_i = 1 | \omega_1, \ldots, \omega_{i-1}] \geq 1 - 1/e$. Using Lemma 3.2 , we can conclude that $Pr[T > \frac{n(1+\delta)}{1-1/e}] = Pr[Y_{\frac{n(1+\delta)}{1-1/e}} < n_s] \leq Pr[B(\frac{n(1+\delta)}{1-1/e}, \frac{e-1}{e}) < n_s] \leq Pr[B(\frac{n(1+\delta)}{1-1/e}, \frac{e-1}{e}) < n]$. Using Lemma 3.1, 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 3.2 that the number of balls in C stochastically dominates[6] the random variable $B(t_2, db/2en)$. Using $bd \geq c \log_2 n$, Lemma 3.1 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/b 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. ϵ_1 can be made arbitrarily small, and ϵ_2 can be made small by choosing a large c. □

Note that Theorem 3.1 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 3.2. *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, 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. $\Pr[\text{Ball is placed in } C \mid C \text{ is chosen}] = \sum_{i=1}^{d} (\frac{1}{i}) \binom{d-1}{i-1} f^{i-1}(1-f)^{d-i} = (\frac{1}{df}) \sum_{i=1}^{d} \binom{d}{i} f^i (1-f)^{d-i} = \frac{1-(1-f)^d}{df} < \frac{1}{df}$. 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

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

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

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 3.2 and the previous claim, it follows that 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

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

Lemma 3.1, 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* [11]. 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 3.1, 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 3.1 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 ith trial, for any $i \leq \beta n$, is also at most $1/\nu^b$. Therefore, the ith 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. □

Notes: Constant α in Theorem 3.1 can be improved (see Theorem 4.1). In fact, we suspect that a sharp threshold result should hold. Further, we speculate that Theorem 3.2 should hold for any $\beta < 1$, not just $\beta < \frac{1}{2}$.

4. PROOF OF THEOREM 2.1

$\forall \ell : r(\ell)v(\ell) \geq c\ell$ for a suitably large constant c.

LEMMA 4.1. *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 leaf nodes belong to most $a+b+1$ different levels w.h.p.

PROOF. Let $2^k \leq n < 2^{k+1}$ for some integer k. W.l.g., let $\mu_1 < \mu_2^{-1}$ (the other case is similar). There are 3 cases:

1. $2^k \leq n \leq \mu_1 2^{k+1}$:
   Leaves belong to levels $[k-a, k+b]$ w.h.p.
2. $\mu_1 2^{k+1} < n < \mu_2^{-1} 2^{k+1}$:
   Leaves belong to levels $[k-a+1, k+b]$ w.h.p.
3. 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.

Thus 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 3.1). 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 3.2, β can be set to any constant less than half. Constants γ and δ emerge in Lemmas 4.2 and 4.3 respectively. The interplay of all four constants will appear towards the end of this section, when we prove Theorem 2.1.

LEMMA 4.2. *$\mathcal{A}^{(k)}$ 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 terminate 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 allocate $\alpha 2^i$ time-steps to each $\mathcal{A}_i$ where $k - \log_2 k - j \leq i \leq k$. With probability at least $1 - 1/poly(2^i) = 1 - O(1/poly(2^k))$, $\mathcal{A}_i$ 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 4.3. *$\mathcal{B}^{(k)}$ takes at least $\beta(2-\delta)2^k$ steps to terminate, w.h.p., where δ is an arbitrarily small constant.*

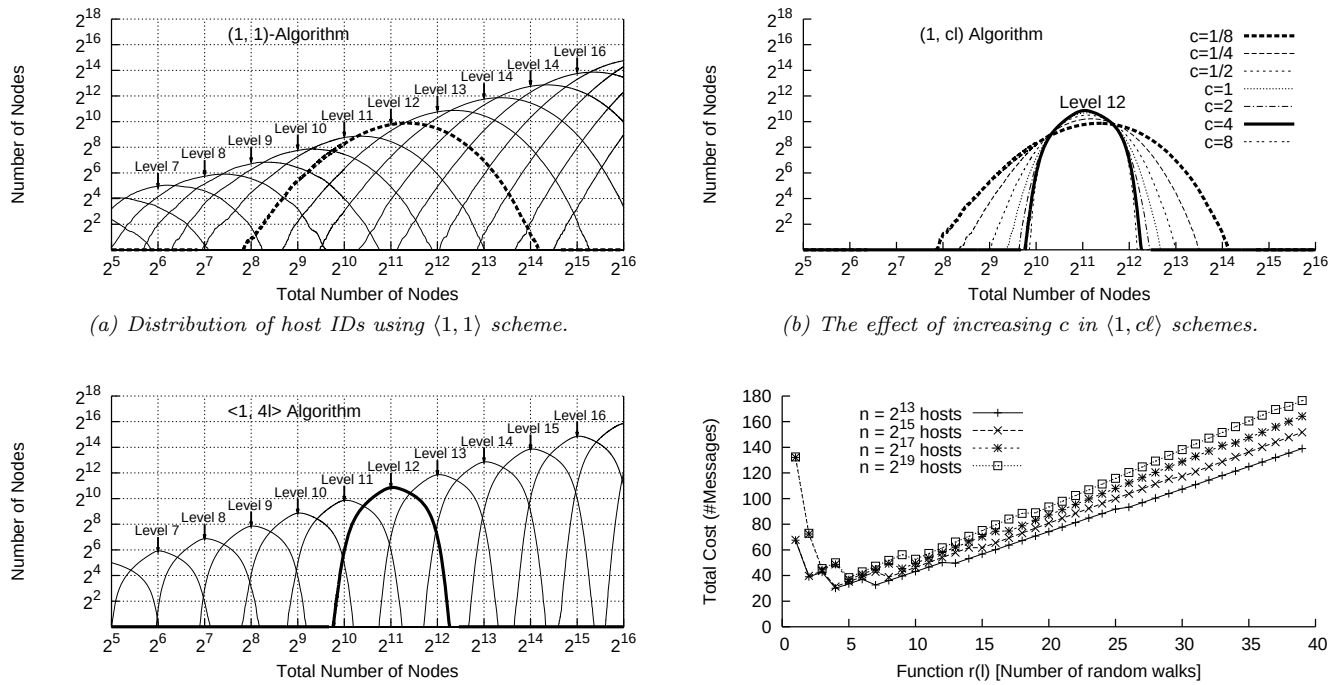
PROOF. Let $j = \lceil \log_2 1/\delta \rceil$, a constant depending upon δ. For $k - j - 1 \leq i \leq k$, the probability that process $\mathcal{B}_i$ runs for less than $\beta 2^i$ steps is at most $O(1/poly(2^i)) = O(1/poly(2^k))$. As a consequence, the series of processes $\langle \mathcal{B}_{k-j-1}, \mathcal{B}_{k-j}, \ldots, \mathcal{B}_k \rangle$ 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 4.4. *When $n > \alpha(2+\gamma)2^L$, no leaf is at level L or less, w.h.p., 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 4.2.

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 4.5. *When $n < \frac{1}{4}\beta(2-\delta)2^L$, no leaf is at level L or more, w.h.p., where δ is an arbitrarily small constant.*



(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. In (c), the $\langle 1, c\ell\rangle$ scheme with $c = 4$ results in 3-level trees. In (d), 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.*

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 $\mathcal{B}^{(L-2)}$, which runs for at least $\frac{1}{4}\beta(2-\delta)2^L$ steps (from Lemma 4.3). $\square$

We can now prove a slightly weaker version of Theorem 2.1: 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 3.1, $\alpha = \frac{e}{e-1} + 2e + \epsilon_1 + \epsilon_2$, 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 3.2), 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 4.1, it 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 2.1, we need a tighter version of Theorem 3.1, which we now prove.

THEOREM 4.1. *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 3.1 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 3.1. 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 3.1 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 $\frac{9}{5}n$. $\square$

5. EXPERIMENTAL RESULTS

Figure 1 studies various schemes for growing binary trees. With $\langle r, v\rangle = \langle 1, 1\rangle$, IDs belong to as many as 6 different levels when $n = 2^{11}$. With $\langle r, v\rangle = \langle 1, 4\ell\rangle$, 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 $\langle c\ell, 1\rangle$.

6. FUTURE DIRECTIONS

Our load-balancing algorithm does not address host departures. 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."* A formal proof for this observation would extend the results in [15, 32]. Structured coupon collection over cliques when $bd \not\geq c \log n$, and the impact of multiple choices ($d \geq 2$) over general graphs appear to be interesting problems.

7. REFERENCES

[1] Ittai Abraham, Baruch Awerbuch, Yossi Azar, Yair Bartal, Dahlia Malkhi, and Elan Pavlov. A generic scheme for building overlay networks in adversarial scenarios. In *Proc. Intl. Parallel and Distributed Processing Symposium (IPDPS 2003)*, April 2003.

[2] Micah Adler, Eran Halperin, Richard M Karp, and Vijay V Vazirani. A stochastic process on the hypercube with applications to peer-to-peer networks. In *Proc. 35nd ACM Symposium on Theory of Computing (STOC 2003)*, pages 575–584, June 2003.

[3] Noga Alon. Problems and results in extremal combinatorics, II. Available as `http://www.math.tau.ac.il/~nogaa/PDFFS/publications.html`, 2004.

[4] James Aspnes, Jonathan Kirsch, and Arvind Krishnamurthy. Load balancing and locality in range-queriable data structures. In *Proc. 23rd ACM Symposium on Principles of Distributed Computing (PODC 2004)*, June 2004.

[5] James Aspnes and Gauri Shah. Skip graphs. In *Proc. 14th ACM-SIAM Symposium on Discrete Algorithms (SODA 2003)*, pages 384–393, January 2003.

[6] Yossi Azar, Andrei Z Broder, Anna R Karlin, and Eli Upfal. Balanced allocations. *SIAM Journal of Computing*, 29(1):180–200, 1999.

[7] John W Byers, Jeffrey Considine, and Michael Mitzenmacher. Geometric generalizations of the power of two choices. In *Proc. 16th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA 2004)*, pages 54–63, June 2004.

[8] Miguel Castro, Peter Druschel, Y Charlie Hu, and Antony I T Rowstron. Topology-aware routing in structured peer-to-peer overlay networks. In *Proc. Intl. Workshop on Future Directions in Distrib. Computing (FuDiCo 2003)*, pages 103–107, 2003.

[9] Herman 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.

[10] Frank Dabek, M Frans Kaashoek, David Karger, Robert Morris, and Ion Stoica. Wide-area cooperative storage with CFS. In *Proc. 18th ACM Symposium on Operating Systems Principles (SOSP 2001)*, pages 202–215, 2001.

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

[12] Pierre Fraigniaud and Philippe Gauron. (brief announcement) An overview of the content-addressable network D2B. In *Proc 22nd ACM Symposium on Principles of Distributed Computing (PODC 2003)*, pages 151–151, July 2003.

[13] Prasanna Ganesan, Mayank Bawa, and Hector Garcia-Molina. Online balancing of range-partitioned data with applications to peer-to-peer systems. In *Proc. 30th Intl. Conf. on Very Large Data Bases (VLDB 2004)*, 2004.

[14] Prasanna Ganesan and Gurmeet Singh Manku. Optimal routing in Chord. In *Proc. 15th ACM-SIAM Symposium on Discrete Algorithms (SODA 2004)*, pages 169–178, January 2004.

[15] George Giakkoupis and Vassos Hadzilacos. A scheme for load balancing in hetergoneous distributed hash tables. In *Proc. 24th ACM Symposium on Principles of Distributed Computing (PODC 2005)*, July 2005.

[16] P Brighten Godfrey, Karthik Lakshminarayanan, Sonesh Surana, Richard M Karp, and Ion Stoica. Load balancing in dynamic structured P2P systems. In *Proc. IEEE INFOCOM 2004*, March 2004.

[17] P Brighten Godfrey and Ion Stoica. Heterogeneity and load balance in distributed hash tables. In *Proc. IEEE INFOCOM 2005*, March 2004.

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

[19] Krishna P Gummadi, Ramakrishna Gummadi, Steven D Gribble, Sylvia Ratnasamy, Scott Shenker, and Ion Stoica. The impact of DHT routing geometry on resilience and proximity. In *Proc. ACM SIGCOMM 2003*, pages 381–394, 2003.

[20] Nicholas J A Harvey, Michael Jones, Stefan Saroiu, Marvin Theimer, and Alec Wolman. SkipNet: A scalable overlay network with practical locality properties. In *Proc. 4th USENIX Symposium on Internet Technologies and Systems (USITS 2003)*, 2003.

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

[22] Norman Lloyd Johnson and Samuel Kotz. *Urn Models and their Applications: An Approach to Modern Discrete Probability Theory*. John Wiley and Sons, 1977.

[23] M Frans Kaashoek and David R Karger. Koorde: A simple degree-optimal hash table. In *Proc. 2nd Intl. Workshop on Peer-to-Peer Systems (IPTPS 2003)*, pages 98–107, 2003.

[24] David R Karger, Eric Lehman, Frank Thomson Leighton, Matthew S Levine, Daniel Lewin, and Rina Panigrahy. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the World Wide Web. In *Proc. 29th ACM Symposium on Theory of Computing (STOC 1997)*, pages 654–663, May 1997.

[25] David R Karger and Matthias Ruhl. Simple efficient load balancing algorithms for peer-to-peer systems. In *Proc. 16th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA 2004)*, pages 36–43, June 2004.

[26] Valerie King and Jared Saia. Choosing a random peer. In *Proc. 23rd ACM Symposium on Principles of Distributed Computing (PODC 2004)*, pages 125–130, July 2004.

[27] Valentin F Kolchin, Boris A Sevast'yanov, and Vladimir P Chistyakov. *Random Allocations*. V H Winston & Sons, 1978.

[28] Abhishek Kumar, Shashidhar Merugu, Jun Jim Xu, and Xingxing Yu. Ulysses: A robust, low-diameter, low-latency peer-to-peer network. In *Proc. 11th IEEE International Conference on Network Protocols (ICNP 2003)*, pages 258–267, November 2003.

[29] Dmitri Loguinov, Anuj Kumar, Vivek Rai, and Sai Ganesh. Graph-theoretic analysis of structured peer-to-peer systems: Routing distance and fault resilience. In *Proc. ACM SIGCOMM 2003*, pages 395–406, 2003.

[30] Dahlia Malkhi, Moni Naor, and David Ratajczak. Viceroy: A scalable and dynamic emulation of the butterfly. In *Proc 21st ACM Symposium on Principles of Distributed Computing (PODC 2002)*, pages 183–192, 2002.

[31] Gurmeet Singh Manku. Routing networks for distributed hash tables. In *Proc. 22nd ACM Symposium on Principles of Distributed Computing (PODC 2003)*, pages 133–142, July 2003.

[32] Gurmeet Singh Manku. Balanced binary trees for ID management and load balance in distributed hash tables. In *Proc. 23rd ACM Symposium on Principles of Distributed Computing (PODC 2004)*, pages 197–205, July 2004.

[33] Gurmeet Singh Manku. *Dipsea: A Modular Distributed Hash Table*. PhD dissertation, Stanford University, Department of Computer Science, August 2004.

[34] Gurmeet Singh Manku, Mayank Bawa, and Prabhakar Raghavan. Symphony: Distributed hashing in a small world. In *Proc. 4th USENIX Symposium on Internet Technologies and Systems (USITS 2003)*, pages 127–140, 2003.

[35] Gurmeet Singh Manku, Moni Naor, and Udi Wieder. Know thy neighbor's neighbor: The power of lookahead in randomized P2P networks. In *Proc. 36th ACM Symposium on Theory of Computing (STOC 2004)*, pages 54–63, June 2004.

[36] Petar Maymounkov and David Mazières. Kademlia: A peer-to-peer information system based on the XOR metric. In *Proc. 1st Intl. Workshop on Peer-to-Peer Systems (IPTPS 2002)*, pages 53–65, 2002.

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

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

[39] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[40] Moni Naor and Udi Wieder. Novel architectures for P2P applications: The continuous-discrete approach. In *Proc. 15th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA 2003)*, pages 50–59, June 2003.

[41] Martin Raab and Angelika Steger. Balls into bins – a simple and tight analysis. In *Randomization and Approximation Techniques in Computer Science (RANDOM 1998), Lecture Notes in Computer Science 1518*, pages 159–170, October 1998.

[42] Sylvia Ratnasamy, Paul Francis, Mark Handley, and Richard M Karp. A scalable Content-Addressable Network. In *Proc. ACM SIGCOMM 2001*, pages 161–172, 2001.

[43] Sean Rhea, Denis Geels, Timothy Roscoe, and John Kubiatowicz. Handling churn in a DHT. In *Proc. 2004 USENIX Annual Technical Conference*, pages 127–140, June 2004.

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

[45] Ion Stoica, Robert Morris, David Karger, M Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for Internet applications. In *Proc. ACM SIGCOMM 2001*, pages 149–160, 2001.

[46] Hui Zhang, Ashish Goel, and Ramesh Govindan. Incrementally improving lookup latency in distributed hash table systems. In *Proc. ACM SIGMETRICS 2003*, pages 114–125, June 2003.

[47] Ben Y Zhao, Ling Huang, Jeremy Stribling, Sean C Rhea, Anthony D Joseph, and John D Kubiatowicz. Tapestry: A resilient global-scale overlay for service deployment. *IEEE Journal on Selected Areas in Communications*, 22(1), January 2004.