Heterogeneity and Load Balance in Distributed Hash Tables

P. Brighten Godfrey and Ion Stoica
Computer Science Division, University of California, Berkeley
{pbg,istoica}@cs.berkeley.edu

Abstract— **Existing solutions to balance load in DHTs incur a high overhead either in terms of routing state or in terms of load movement generated by nodes arriving or departing the system. In this paper, we propose a set of general techniques and use them to develop a protocol based on Chord, called Y_0, that achieves load balancing with minimal overhead under the typical assumption that the load is uniformly distributed in the identifier space. In particular, we prove that Y_0 can achieve near-optimal load balancing, while moving little load to maintain the balance and increasing the size of the routing tables by at most a constant factor.**

Using extensive simulations based on real-world and synthetic capacity distributions, we show that Y_0 reduces the load imbalance of Chord from $O(\log n)$ to a less than 3.6 without increasing the number of links that a node needs to maintain. In addition, we study the effect of heterogeneity on both DHTs, demonstrating significantly reduced average route length as node capacities become increasingly heterogeneous. For a real-word distribution of node capacities, the route length in Y_0 is asymptotically less than half the route length in the case of a homogeneous system.

I. Introduction

During the last few years the distributed hash table (DHT) has emerged as a flexible and general architecture that can support a large variety of applications including file sharing [9], [24], storage systems [21], query processing [16], name services [36], and communication services [39], [6], [32]. A DHT manages a global identifier (ID) space that is partitioned among n nodes organized in an overlay network. To partition the space, each node is given a unique ID x and owns the set of IDs that are "closest" to x. Each object is given an ID, and the DHT stores an object at the node which owns the object's ID. To locate the owner of a given ID, a DHT typically implements a greedy lookup protocol that contacts $O(\log n)$ other nodes, and requires each node to maintain a routing table of size $O(\log n)$.

One central challenge in the DHT design is how to balance the load across the nodes in the system. Even in the case of a homogeneous system where all nodes have the same capacity, DHTs can exhibit an $O(\log n)$ imbalance factor [33]. The imbalance can significantly increase as the heterogeneity of the system increases.

Two classes of solutions have been proposed so far to address this challenge. Solutions in the first class use the concept of *virtual servers* [18], [9]. Each physical node instantiates one or more virtual servers with random IDs that act as peers in the DHT. In the case of a homogeneous system, maintaining $\Theta(\log n)$ virtual servers per physical node reduces the load imbalance to a constant factor. To handle heterogeneity, each node picks a number of virtual servers proportional to its capacity. Unfortunately, virtual servers incur a significant cost: a node with k virtual servers must maintain k sets of overlay links. Typically $k = \Theta(\log n)$, which leads to an asymptotic increase in overhead.

The second class of solutions uses just a single ID per node [26], [22], [20]. However, all such solutions must reassign IDs to maintain the load balance as nodes arrive and depart the system [26]. This can result in a high overhead because it involves transferring objects and updating overlay links. In addition, none of these solutions handles heterogeneity directly, although they could be combined with the virtual server technique.

In this paper, we present a simple DHT protocol, called Y_0, that addresses the above drawbacks. Y_0 is based on the concept of virtual servers, but with a twist: instead of picking k virtual servers with random IDs, a node clusters those IDs in a random fraction $\Theta(k/n)$ of the ID space. This allows the node to share a single set of overlay links among all k virtual servers. As a result, we can show that the number of links per physical node is still $\Theta(\log n)$, even with $\Theta(\log n)$ virtual servers per physical node.

In addition, we show that heterogeneity, rather than being an issue, can be an asset. Higher-capacity nodes have a denser set of overlay links and lower-capacity nodes are less involved in routing, which results in reduced route length compared to the homogeneous case. While both Chord and Y_0 see improvement, Y_0's is more significant because its placement of virtual servers provides more control over the topology.

Like most previous DHT work, we operate under the *uniform load assumption*, that the load of each node is proportional to the size of the ID space it owns. This is reasonable when all objects generate similar load ($e.g.$, have the same size), the object IDs are randomly chosen ($e.g.$, are computed as a hash of the object's content), and the number of objects is large compared to the number of nodes ($e.g.$, $\Omega(n \log n)$). Alternately, we can unconditionally balance the *expected* load over uniform-random choices of object IDs.

Our main contributions are the following.

- We introduce a heterogeneity-aware ID selection algorithm for ring-based DHTs, Low-Cost Virtual Server Selection (LC-VSS). We prove that LC-VSS can balance the ID space partitioning within a factor $(1+\varepsilon)$ of optimal for any $\varepsilon > 0$, and that while the system size and

average capacity remain relatively stable, the amount of load movement to maintain that balance is nearly optimal.
- We prove that LC-VSS can be used with arbitrary overlay topologies while increasing route length by at most an additive constant and outdegree by at most a constant factor, even with $\Theta(\log n)$ virtual servers. Furthermore, our construction provides some flexibility in neighbor selection, even if the underlying topology lacks it.
- We apply LC-VSS to Chord and extensively evaluate the resulting protocol, called Y_0. Simulations in various capacity distributions show that Y_0 ensures that all nodes have less than 3.6 times their fair share of the ID space with no more overlay links than in Chord with a single virtual server. Furthermore, we show that heterogeneity decreases route length in Chord and more significantly in Y_0, with Y_0's route lengths asymptotically roughly 55% shorter in a real-world distribution than in the homogeneous case.

The paper is organized as follows. Section II discusses our model, the ID selection problem, and the technique of virtual servers. Section III introduces LC-VSS and its application to Chord to produce the Y_0 DHT. Section IV gives theoretical guarantees on Y_0's performance when it is generalized to arbitrary overlay topologies. Section V evaluates Y_0 and Chord through simulation. Section VI discusses related work and Section VII concludes.

II. Preliminaries

A. Model

We assume a system with n physical nodes. Node v has a fixed *capacity* c_v. Capacities are normalized so that the average capacity is 1; that is, $\sum_v c_v = n$. Our use of a scalar capacity assumes that there is a single important resource on which nodes are constrained, such as storage space, processing speed, or last-mile bandwidth.

We assume that each node v can estimate c_v and n within a factor γ_c and γ_n, respectively, of the true values, with high probability. We further assume the estimates are unbiased. Estimation of n is discussed in [22]. Since capacities are normalized, to estimate c_v, a node will need an estimate of the average capacity. One may obtain a crude estimate through random sampling of other nodes, such as the successors in the ID space. The techniques of [27] could be applied to DHTs to estimate both n and average capacity and would provide guarantees on the quality of the estimate.

We say that an event happens with high probability (w.h.p.) when it occurs with probability $1 - O(n^{-1})$.

We assume a DHT that manages a unit-size circular ID space, $i.e.$, $[0, 1) \subseteq \mathbb{R}$ employing arithmetic modulo 1. We assume the DHT uses consisting hashing [18] to partition the ID space among the nodes as in Chord. Each node v picks an ID $\mathrm{id}(v) \in [0, 1)$ and is assigned ownership of the region $(\mathrm{id}(w), \mathrm{id}(v)]$ where $\mathrm{id}(w)$ is the nearest preceding node's ID. A node may pick multiple IDs (virtual servers) in which case it owns the union of the associated regions.

B. The ID selection problem

Under the uniform load assumption, the load on a node will be proportional to the size of the ID space it owns. Thus, picking a load-balanced partitioning amounts to selecting appropriate IDs for the nodes. Let the *share* of node v be the fraction f_v of the ID space assigned to it, divided by its "fair share" of the ID space:

$$\mathrm{share}(v) = \frac{f_v}{c_v/n}.$$

A good partitioning has the following properties.

- **Load balance** The *maximum share* in the system should be as low as possible — ideally, 1 — so that the load on each node is proportional to its capacity.
- **Load movement** To maintain load balance, nodes may need to select different IDs when other nodes arrive or depart. This can be costly because reassigning ownership of the ID space implies data movement and changes to the overlay connections. Thus, we desire little change in the ID space partitioning upon node arrivals and departures.
- **Normalized degree** The *normalized degree* of a node v is $\deg(v)/c_v$, where $\deg(v)$ is the number of distinct nodes to which v maintains connections or which maintain connections to v in the overlay network used for routing. We wish to minimize the average and maximum normalized degree.

The main reason for the last objective is not to reduce the memory footprint of the routing table, but to reduce the overhead of the control traffic required to keep the routing table entries up to date.

C. Basic Virtual Server Selection (Basic-VSS) Scheme

As discussed in the introduction, the virtual server technique can be used to balance the load not only in a homogeneous system, but also in a heterogeneous system [9]. However, the precise way in which the technique is applied to highly heterogeneous systems has not been specified. In the reminder of this section we present a simple strategy that we later adapt in Y_0.

Let $\alpha = \alpha(n)$ be *the number of virtual servers per unit capacity*. When $\alpha = 1$, nodes of average capacity have a single virtual server, but the maximum share is $\Theta(\log n)$. When $\alpha = \Theta(\log n)$, we can achieve $\Theta(1)$ maximum share but the degree of a node increases by a factor $\alpha = \Theta(\log n)$.

The main issue we address in this section is how to handle low capacity nodes. Since the total system capacity is n, there are roughly $n\alpha$ virtual servers. Thus, the expected fraction of ID space associated with a single virtual server is $1/(n\alpha)$. There is a tradeoff in the choice of α: if α is small, very low capacity nodes will be overloaded even if they maintain only a single virtual server. On the other hand, a large α leads to high degree, as even nodes of average capacity must maintain α virtual servers and the associated overlay connections for each. In particular, to ensure that the nodes of minimum capacity c_{min} have close to their fair share c_{min}/n of the ID space, we must have $1/(n\alpha) = O(c_{min}/n)$, $i.e.$, $\alpha = \Omega(1/c_{min})$.

```
1) $\tilde{n}, \tilde{c}_v \leftarrow$ estimates of $n$ and $c_v$
2) $m \leftarrow$ if $\tilde{c}_v < \gamma_d$ then 0 else $\lfloor 0.5 + \tilde{c}_v \alpha(\tilde{n}) \rfloor$
3) Choose $m$ virtual servers with IDs $r_1, \ldots, r_m$ where each
   $r_i$ is chosen uniformly at random $\in [0, 1)$
4) Reselect IDs as above when $\tilde{c}_v$ changes by a factor $\geq \gamma_u$ or
   $\tilde{n}$ changes by a factor $\geq 2$
```

Fig. 1. The Basic Virtual Server Selection Scheme (Basic-VSS), run at each node v.

Parameter	Description
$\gamma_c, \gamma_n \geq 1$	Bound the maximum factor error (w.h.p.) in each node's estimate of its capacity and of n, respectively; see Section II-A.
$\gamma_d < 1$	Capacity threshold below which a node is discarded.
$\gamma_u > 1$	Each node updates its IDs when its estimate of its capacity changes by a factor γ_u (must have $\gamma_u > \gamma_c$ to avoid instability)
$\alpha(n)$	Number of virtual servers per unit capacity

Fig. 2. Parameters of both Basic-VSS and Y_0's LC-VSS.

But since c_{min} may be arbitrarily small, α may be arbitrarily large, implying very high overhead. Moreover, c_{min} may be unstable and hard to estimate.

We avoid this tradeoff by simply discarding nodes whose capacity is lower than some *discard threshold* γ_d. If the capacity of a node is less than γ_d, the node does not instantiate any virtual server, and does not participate in the standard DHT routing protocol. The remaining nodes v with capacity $c_v \geq \gamma_d$ pick $c_v \cdot \alpha$ virtual servers. We call this algorithm the *Basic Virtual Server Selection (Basic-VSS) Scheme* and show the pseudocode in Figure 1. Figure 2 shows the main parameters of the algorithm.

A natural concern with this algorithm is that it might discard too much capacity, overburdening the remaining nodes. But it is easy to see that in the worst case, at most a fraction γ_d of the total capacity is discarded — ignoring estimation error and lazy update to avoid instability and excessive load movement as (normalized) capacity changes. Removing those simplifications, we have the following, which we prove in [13]:

Claim 1: In the Basic-VSS scheme, the fraction of the total capacity remaining in the ring is at least $1 - \gamma_c\gamma_u\gamma_d$ w.h.p.

An optimization deserving of further study is to fix α and find the γ_d which minimizes the maximum share. However, we expect that $\gamma_d = \frac{1}{2}$ will be acceptable for most applications. With this choice, the capacity of any node remaining in the ring is at least half the average capacity, so we do not need to significantly increase α to handle the low-capacity nodes. Although in the worst case 50% of the total capacity is discarded, in our simulations of Section V, less than 20% is discarded in a range of power law capacity distributions and less than 10% in a real-world distribution. If the discarded capacity were excessive in some distribution, we could use a smaller γ_d at the cost of increasing α to maintain the same load balance.

Furthermore, if we cannot afford to discard *any* capacity, we can use discarded nodes for data storage (but not routing) by having each discarded node pick a "parent" in the ring which would assign it data to store.

Discarded nodes may still perform lookup operations in the DHT although they are not part of the ring structure and do not route messages. We have discarded nodes connect to the system through k links to nodes owning the IDs $r + \frac{1}{k}, \ldots, r + \frac{k}{k}$, where $r \in [0, 1)$ is chosen randomly. In our simulations of Section V, we use $k = \lceil 3c_v \log_2 n \rceil$ for node v since with $\alpha = 1$, Chord nodes in the ring have roughly $3c_v \log_2 n$ outlinks as well ($\approx \log n$ fingers and $\approx 2 \log n$ successors for each of the $\approx c_v$ virtual servers).

This raises another natural question: whether the congestion on the nodes in the ring will increase due to lookup operations. However, Claim 1 implies that the increase cannot be too great. Indeed, in Section V-A, we will see a slight drop in congestion due to decreased route length, which arises in part because there are fewer nodes in the ring.

III. The Y_0 DHT

In this section we present the Y_0 DHT, which is composed of a *Low Cost Virtual Server Selection (LC-VSS) Scheme* and simple changes to Chord's successor lists, finger tables, and routing. Later, in Section IV-C, we present formal guarantees for LC-VSS applied to any overlay topology, not just Chord.

A. Low Cost Virtual Server Selection (LC-VSS) Scheme

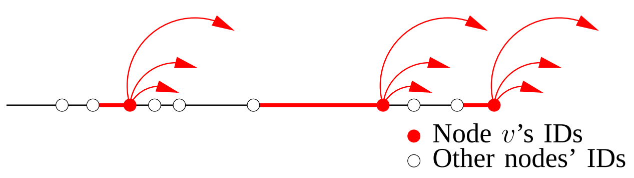
The LC-VSS scheme which selects IDs for Y_0's virtual servers is shown in the illustration of Figure 3 and in pseudocode in Figure 4. As in the Basic-VSS scheme of Section II-C, nodes of capacity less than γ_d are discarded and each remaining node v chooses $\Theta(c_v\alpha)$ IDs, where $\alpha = \Theta(\log n)$. However, these IDs are not selected randomly from the entire ID space. Instead, we pick a random starting point p_v and then select one random ID within each of $\Theta(\log n)$ consecutive intervals of size $\Theta(1/n)$. When n or c_v changes by a constant factor, the ID locations are updated. The algorithm inherits the parameters shown in Figure 2.

It has been proposed to compute a node's ID as a hash of its IP address, which makes it more difficult for a node to hijack arbitrary regions of the ID space [33]. To support this, we could easily replace the randomness in the description of the algorithm with such hashes.

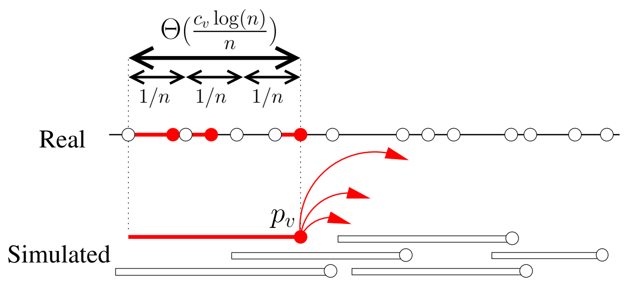
B. Successor Lists

In Chord, each virtual server keeps links to its $2 \log_2 n$ successors in the ring [33]. The purpose is fault tolerance: when each node fails with probability $\frac{1}{2}$, each remaining virtual server will still have a link to its immediate successor w.h.p. To obtain the same bound in Y_0, each node must keep links to the $2 \log n$ *distinct* nodes succeeding each of its virtual servers.

Chord constructs its successor list as follows. Each virtual server vs belonging to node v obtains its successor list from its immediate successor $succ(vs)$ and adds $succ(vs)$ to the list. If the list is of length $> 2 \log n$, it drops the clockwise-farthest



(a) **Basic-VSS** gives a node of capacity c_v ownership of $\Theta(c_v \log n)$ disjoint segments of the ID space. The DHT must maintain a set of overlay links for each.



(b) Y_0's **LC-VSS** scheme results in $\Theta(c_v \log n)$ disjoint segments clustered in a $\Theta(\frac{c_v \log n}{n})$ fraction of the ID space. When Y_0 builds the overlay, it *simulates* ownership of one contiguous interval. The nodes' simulated intervals overlap.

Fig. 3. ID selection illustrated.

Initialization:
1) $p_v \leftarrow$ random ID $\in [0,1)$
2) $\tilde{n}, \tilde{c}_v \leftarrow$ estimates of n and c_v

ID selection:
1) $spacing \leftarrow 2^{-\lfloor 0.5+\log_2 \tilde{n} \rfloor}$ (*i.e.*, roughly $1/\tilde{n}$)
2) $start \leftarrow \lfloor p_v / spacing \rfloor \cdot spacing$
3) $m \leftarrow$ if $\tilde{c}_v < \gamma_d$ then 0 else $\lfloor 0.5 + \tilde{c}_v \alpha(\tilde{n}) \rfloor$
4) Choose m IDs at $start - (i + r_i) \cdot spacing$ for each $i \in \{0, \ldots, m-1\}$ and each r_i chosen u.a.r. $\in [0,1)$

Periodic update:
1) Obtain new estimates $\tilde{c}'_v, \tilde{n}'$
2) If $\tilde{c}'_v$ is at least a factor γ_u from $\tilde{c}_v$, set $\tilde{c}_v \leftarrow \tilde{c}'_v$
3) If $spacing \notin [\frac{1}{2\tilde{n}'}, \frac{2}{\tilde{n}'}]$, set $\tilde{n} \leftarrow \tilde{n}'$
4) If either $\tilde{n}$ or $\tilde{c}_v$ changed, reselect IDs as above

Fig. 4. Y_0's LC-VSS scheme run at each node v.

entry. In Y_0, we drop any virtual server belonging to v before considering dropping the clockwise-farthest entry. Beginning with an empty list at each node, periodic repetition of this algorithm builds a list of $2 \log_2 n$ distinct successors.

Will a node's $\Theta(c_v \log n)$ virtual servers involve $\Theta(c_v \log^2 n)$ successor links, as in Basic-VSS? We show in Section IV-C that since Y_0's IDs are clustered, these $\Theta(c_v \log^2 n)$ *logical* successors fortunately involve only $\Theta(c_v \log n)$ *distinct* nodes, so only $\Theta(c_v \log n)$ network connections need to be maintained. Memory use does increase by a logarithmic factor, but this is not likely to be detrimental. Even when $n = 2^{30}$, a node with $\log n$ virtual servers would maintain $2 \log^2 n = 1800$ logical successors,

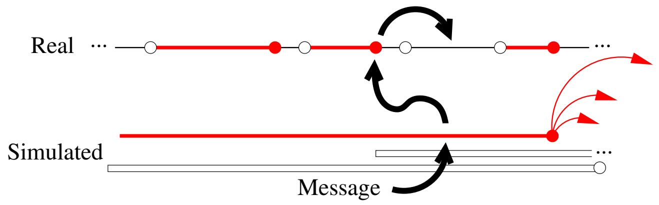


Fig. 5. Routing in Y_0. A close-up of part of the ID space is shown.

which at a generous 1 KB each uses less than 2 MB of RAM.

C. Finger Tables

For each virtual server with ID x, Chord keeps a *finger table* of links to the nodes owning the "target IDs" $(x+b^{-i}) \mod 1$ for $i = 1, 2, \ldots$. Typically $b = 2$.

In Y_0, for the purposes of overlay construction, each node v *simulates* ownership of a contiguous interval I_v of size $\Theta(\frac{c_v \log n}{n})$ enclosing its virtual servers, building one set of overlay links as if it actually owned all of I_v. This is depicted in Figure 3(b). Specifically, v simply keeps one finger table emanating from the clockwise end of I_v (see Section III-A). An important property is that although a node's virtual server IDs may change, p_v is fixed. Thus, the finger table changes only when a target ID changes ownership.

This construction allows us to choose a better overlay topology than Chord in the heterogeneous case. The size of I_v scales with the node's capacity, but this does not affect the size of the finger table. Since we allow each node a number of outlinks proportional to its capacity, we have some unused capacity on high-capacity nodes. We use this to reduce route length by choosing a "denser" set of overlay links on high-capacity nodes: node v chooses fingers at integer powers of b^{1/c_v}, where b is a parameter which we take to be 2 by default as in Chord. This results in $O(\log_{b^{1/c_v}} n) = O(c_v \log_b n)$ fingers w.h.p. Note that, unlike the rest of Y_0, this technique does not generalize to arbitrary overlay topologies.

D. Routing

It turns out that the only change we need to make to Chord's greedy routing is that both the successor list and the finger table are considered, rather than just the latter.

To find the owner of some ID x, Chord routes greedily on the finger table, at each step selecting the neighbor whose ID is clockwise-closest to x. In Y_0, we employ the same algorithm until we arrive at a node v for which $x \in I_v$. At this point, v doesn't truly own x, and we have no fingers which are closer to the true destination. However, due to the clustering of v's IDs, the real owner is not far from one of v's virtual servers. Specifically, they are separated by $O(\log n)$ nodes. We can complete the route using greedy routing over the successor list in $\Theta(1)$ additional hops, as shown in Figure 5.

IV. Analysis

In this section we analyze the performance of Y_0 with respect to the metrics described in Section II-B. The results apply Y_0's techniques to any ring-based DHT, e.g [33], [29], [17], [26], [23]. We prove the following:

- Y_0 can achieve a near-optimal maximum share of $1+\varepsilon$ for any $\varepsilon > 0$ (Section IV-A).
- As long as the number of nodes n and the average capacity remain relatively stable, the amount of load movement that Y_0 incurs to maintain a good balance is close to optimal (Section IV-B).
- Compared to the case of a single virtual server, Y_0 with $\alpha = \Theta(\log n)$ increases the number of distinct links that a node maintains by at most a constant factor, while providing flexibility in neighbor selection for any topology (Section IV-C).
- Compared to the case of a single virtual server, Y_0 with $\alpha = \Theta(\log n)$ increases route length by at most an additive constant (Section IV-D).

We defer all proofs to [13].

A. Load Balance Bounds

Our first theorem says that the maximum share can be made arbitrarily close to 1, even in the heterogeneous case. Throughout the analysis we assume that all nodes have performed their periodic update step since the last event in the system.

Theorem 2: When $\alpha \geq \frac{8\gamma_n\gamma_c\gamma_u}{(1-\gamma_c\gamma_u\gamma_d)\gamma_d\varepsilon^2} \cdot \ln n$, the maximum share of a node is at most $\frac{(\gamma_c\gamma_u)^2}{(1-\varepsilon)^2(1-\gamma_c\gamma_u\gamma_d)} + o(1)$ w.h.p. for any $\varepsilon > 0$.

Proof: (Sketch) When $\alpha = \Theta(\log n)$, we have $\Theta(\log n)$ IDs in each region of the ID space of size $\Theta(1/n)$ w.h.p. regardless of the capacity distribution. With this balanced distribution of IDs, the share associated with each ID is approximately a geometric random variable. Applying a Chernoff bound to their sum yields the result. ■

Despite the frightening plethora of constants in the bound, our simulations of Section V show that we get a maximum share of less than 3.6 with a modest $\alpha = 2\log_2 n$ in various capacity distributions. Furthermore, most of the complexity in the above guarantee is due to varying node capacities. Hard-coding $\tilde{c}_v = 1$ into LC-VSS and a straightforward modification of our analysis yields the following bound:

Theorem 3: If node capacities are homogeneous, when $\alpha \geq \frac{8\gamma_n}{\varepsilon^2}\ln n$, the maximum share of a node is at most $(1-\varepsilon)^{-2} + o(1)$ w.h.p. for any $\varepsilon > 0$.

B. Load Movement Bounds

To maintain load balance as nodes join and leave, nodes occasionally update their IDs. Each move of a virtual server is equivalent to a *leave* followed by a *join* under a new ID. Thus, load balancing effectively increases the churn rate of the system, which involves costly object movement and overlay link changes. In this section we bound this overhead in terms of the amount of churn in the population of nodes in the system.

Given a sequence of node join and leave events, we say that the *(aggregate) underlying churn* is the sum over all events of the fraction change in system capacity due to the event. Specifically, if the system currently has total capacity C and a node of capacity c joins, the underlying churn increases by $c/(C+c)$; if the same node subsequently leaves, churn increases by the same amount again. Similarly, the *(aggregate) effective churn* is the sum over all events of the fraction of ID space which changes ownership due to the event, which depends on the policy of the partitioning scheme.

Definition 1: The *churn ratio* for a sequence of events is the DHT's expected effective churn divided by its underlying churn over those events.

This is equivalent to a metric used in [3].

We are interested the churn ratio after t events for which the underlying churn is $\Omega(t)$ (to avoid degenerate cases such as zero-capacity nodes joining and leaving forever). We also assume the system always has positive capacity. Finally for simplicity we assume LC-VSS's estimation error bounds γ_c, γ_n hold with probability 1.

In [13] we give a simple example which shows the churn ratio is ≥ 1 in the worst case for any scheme — that is, the effective churn is at least the underlying churn. Basic-VSS in a homogeneous environment with $\alpha = 1$ ($i.e.,$ Chord) achieves this lower bound, since in expectation the change in ID space due to a join or leave is exactly the change in total system capacity, and once joined, a node never changes its ID.

Our bound on LC-VSS's churn ratio will apply to event sequences during which the total number of nodes and average capacity don't vary greatly. Note that *the underlying churn rate may be arbitrarily high.*

Definition 2: The system is (β, δ)*-stable* over a sequence of events when the minimum and maximum values of n differ by less than a factor β, and the minimum and maximum values of the average capacity differ by less than a factor δ.

Theorem 4: If the system is $(\frac{2}{\gamma_n}, \frac{\gamma_u}{\gamma_c})$-stable, the churn ratio of LC-VSS is $\frac{1}{1-\gamma_c\gamma_u\gamma_d} + o(1)$.

This reduces to $1 + o(1)$ in the homogeneous case. Thus, during periods wherein n and the average capacity are relatively stable — which is likely the common case — the overhead of LC-VSS's load balancing operations is negligible. The result follows from the fact that in this case virtually no nodes reselect IDs.

C. Overlay Construction and Degree Bounds

In this section we describe how to use LC-VSS with any overlay topology without significantly increasing outdegree, while providing some flexibility in neighbor selection, even if the original topology had none.

Sequential Neighbors. We deal first with what [14] terms *sequential neighbors*. In ring-based DHTs, each node v keeps links to nodes whose IDs are close to each of v's IDs — either the $k = 2\log_2 n$ successors in the ring [33] or the k successors and k predecessors [29]. As discussed in Section III-B, in Y_0, each node keeps links to the k *distinct* nodes closest to each of its IDs.

In our overlay construction we assume $k = \Theta(\log n)$. Since $\alpha = \Theta(\log n)$, this implies that nodes have $\Theta(c_v \log^2 n)$ *logical* sequential neighbors. The following theorem shows that due to the clustering of IDs, the number of *distinct* neighbors is low.

Theorem 5: Each Y_0 node maintains links to $\Theta(c_v\alpha) = \Theta(c_v \log n)$ sequential neighbors w.h.p.

Long-Distance Neighbors. There is a significant amount of variation in the structure of "long-distance" overlay links in DHTs. To analyze the construction of arbitrary topologies on top of Y_0's partitioning of the ID space, we extend and formalize Naor and Wieder's Continuous-Discrete Approach [26]. We assume the overlay topology is specified in continuous form, as a function E which maps a contiguous interval $(p_1, p_2] \subseteq [0, 1)$ to a subset of $[0, 1)$ which constitutes the neighborhood of $(p_1, p_2]$. For example, Chord's finger table would be specified as

$$E(p_1, p_2) = \{p_2 + 2^{-1},\ p_2 + 2^{-2},\ \ldots\}.$$

Chord does not depend on p_1, but general topologies may. The Distance Halving overlay of [26] has

$$E(p_1, p_2) = \left(\frac{p_1}{2}, \frac{p_2}{2}\right] \cup \left(p_1 + \frac{1}{2}, p_2 + \frac{1}{2}\right].$$

The continuous graph E relates to the real world thusly:

Definition 3: A *discretization* of E is a *simulated interval* $I_v \subseteq [0, 1)$ for each node v and a graph G on the nodes in the system such that

$$\forall v\ \forall p \in E(I_v) \quad (\exists (v, w) \in G \quad p \in I_w).$$

That is, if I_v is connected to a point $p \in E(I_v)$ in the continuous graph, then v is connected to some node $w \in G$ simulating ownership of p.

Thus, a discretization guarantees that each edge in the continuous graph can be simulated by the discrete graph. The set I_v defines what part of the continuous graph node v simulates. Note that the I_vs may overlap.

The simplest example is what we call the *Standard Discretization*. For each v we simply let I_v be the subset of the ID space which v owns. The edges of G are the minimal set which satisfies the above definition. That is, the owner of I_v maintains a link to each node which owns a nonempty subset of $E(I_v)$. This edge set is unique because ownership of the ID space is a partitioning: for each $\ell \in E(I_v)$, there is exactly one node w with $\ell \in I_w$. This discretization is equivalent to the operation of Chord and the Distance Halving DHT.

To adapt to our multiple-virtual server setting with low degree, we use the following *Shared Discretization*. For each v we let I_v be the smallest contiguous interval which contains the (disjoint) set of IDs that v owns as well as p_v (recall the definition from Figure 4). Since the I_vs overlap, we have a fair amount of choice in selecting the edge set. We use a simple greedy algorithm. Initially, label all of $E(I_v)$ uncovered. Iterate by picking the uncovered point p which is the first from p_v in the clockwise direction. Contact p's owner, get its successor list, its furthest successor's successor list, etc. until we find a node w for which I_w covers all of $[p, p + \Theta(\frac{\log n}{n})] \cap E(I_v)$. Add an edge from v to w. Terminate when all of $E(I_v)$ is covered.

Assuming eventual termination, the above procedure clearly satisfies Definition 3. The number of overlay links it creates depends on E. Let $f(n)$ be a nondecreasing function such that if $|I| \leq \frac{1}{n}$, then $E(I)$ can be covered by $\leq f(n)$ segments of length $\frac{1}{n}$. Taking $f(n)$ to be as small as possible intuitively provides a lower bound on node degree: even in a homogeneous system with a perfect partitioning of the ID space, the Standard Discretization must give some nodes $\geq f(n)$ outlinks. The following theorem says that in LC-VSS, nodes' outdegrees are inflated by at most a constant factor.

Theorem 6: Using LC-VSS with $\alpha = \Theta(\log n)$ and the Shared Discretization, each node v has at most $O(c_v f(n))$ outlinks w.h.p. for any capacity distribution.

Flexibility in Neighbor Selection. When the topology allows some choice in a node's neighbors, we can employ proximity-aware neighbor selection (PNS), which can significantly reduce the latency of DHT lookups [14]. Although Chord has a great deal of flexibility, some DHT topologies do not. In [13] we show that in Y_0, the fact that the simulated intervals I_v overlap provides $\Theta(\log n)$ choices for each of a node's long-distance neighbors, *even if there was no choice in the original topology*. Even this fairly limited choice is likely to provide a significant benefit. Simulations suggest that 16 choices for each neighbor provide nearly all the benefit that an unbounded number of choices would provide [10].

Of course, a necessary assumption is that node capacities are at most $O(n/\log n)$. For example, if only one node was not discarded, then there is only one choice for each neighbor.

D. Route Length

To complete our adaptation of the overlay, we show that we can simulate the overlay's routing function while increasing route length by at most an additive constant.

The continuous overlay is accompanied by a *routing function* $r(I, t)$ which, given a current location $I \subseteq [0, 1)$ and a target destination t, returns an ID in $E(I)$. At each step, we apply the routing function to the current location I and move to a neighbor w for which $r(I, t) \in I_w$. Since $r(I, t) \in E(I)$, by definition, any discretization must have an edge to some such w. We set $I \leftarrow I_w$ and iterate. If after $f(m)$ such steps the distance from some $\ell \in I$ to t in the ID space is $\leq 1/m$ for any s and t, we say that E *has has path length* $f(m)$.

Thus, in $f(n)$ hops, any discretization can arrive at a node v such that some ID in I_v is at distance $\leq 1/n$ from the destination t. In the Shared Discretization, unlike the Standard Discretization, v does not own all of I_v. However, due to LC-VSS's placement of IDs at intervals of $O(1/n)$ in the ID space, v will always have a virtual server whose ID is at distance $O(1/n)$ from t. A Chernoff bound shows that when $\alpha = O(\log n)$, the owner of t is $O(\log n)$ successors away, and we route to our destination in $O(1)$ hops using the successor lists which we keep for each chosen ID. Thus, we arrive at the following result.

Theorem 7: Using LC-VSS and the Shared Discretization of any overlay topology with path length $f(n)$, any message can be routed in $f(n) + O(1)$ hops w.h.p.

V. Simulation

We compare Y_0 and Chord using a simple simulator which constructs the ID space partitioning and overlay topology of the two DHTs in a static setting. To provide heterogeneity-aware load balance in Chord, we use the Basic-VSS scheme of Section II-C.

We assume that each node is given its capacity and n; $i.e.$, there is no estimation error. We use $\gamma_d = \frac{1}{2}$. Each data point represents an average over 15 independent trials. We show 95% confidence intervals. We show results for n equal to powers of 2. For intermediate n, there is some variation in the load balance/degree tradeoff space for Y_0 because its spacing between virtual servers is rounded to the nearest power of 2. Specifically, simulations show maximum share increases by up to $\approx 10\%$ when degree decreases up to 10%, or maximum share decreases up to 10% while degree increases up to 25%.

We study three types of node capacity distributions: (1) homogeneous, (2) power law with various exponents, and (3) samples from the uplink bottleneck bandwidths of actual Gnutella hosts measured by Saroiu et al [31], which we call the *SGG distribution*. Thus, for this simulation, we consider the capacity of a node to be the bandwidth of its connection to the Internet. We have discarded points in their raw data set which show bandwidth higher than 1 Gbps, since the authors of [31] believe those to be artifacts of faulty measurement and discard them in their study as well [30].

Our simulation results are as follows.

- Y_0 achieves a maximum share of less than 3.6 with $\alpha = 2\log n$, where α is the number of virtual servers per unit capacity. This is roughly as well-balanced as Chord with $\alpha = \log n$ (Section V-A).
- The degree of nodes in Y_0 with $\alpha = 2\log n$ is as low as Chord with $\alpha = 1$ and even lower in some heterogeneous distributions (Section V-B).
- In both DHTs, route lengths decrease as heterogeneity increases, but Y_0 has a clear asymptotic advantage (Section V-C).

A. Load Balance

Figure 6 summarizes the tradeoff between quality of load balance and degree. We obtain various points on the tradeoff curve for each protocol by selecting different α. Y_0 provides a substantially better set of achievable points. Both algorithms perform significantly better in the SGG distribution, although we will see (Figures 7(b) and 9(b)) that not all capacity distributions are better than the homogeneous case.

Figure 7(a) shows that with $\alpha = 2\log n$, the maximum share of any node in Y_0 is less than 2.7 in a homogeneous environment — nearly as good as Chord with $\alpha = \log n$ but (as we will see) at much lower cost. A larger α provides a slightly better balance. Figure 7(b) shows that in a range of power law capacity distributions with varying exponent, the balance can be slightly worse or better than in the homogeneous case, at all times staying below 3.6.

Once the ID space is balanced, balance of overlay connections and routing load follow closely. We measure maximum

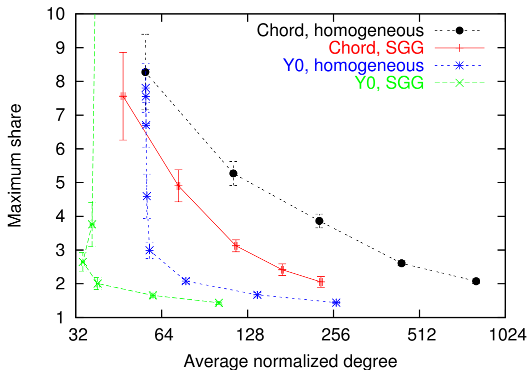


Fig. 6. Tradeoff between maximum share and average normalized degree, achieved through varying α, for $n = 2048$. For Chord, $\alpha \in \{1, 2, 4, 8, 16\}$, and for Y_0, $\alpha \in \{1, 2, 4, \ldots, 128\}$.

degree as defined in Section II-B. To measure balance of routing load, we have each node route a message to a random ID. The *congestion* at a node is the number of messages that flow through it, divided by its capacity. Both metrics are $\Theta(\log^2 n)$ in Chord with $\alpha = 1$: essentially, $\Theta(n \log n)$ messages and $\Theta(n \log n)$ fingers are distributed uniformly in the ID space, and some nodes own a fraction $\Theta(\frac{\log n}{n})$ of the space. Y_0's constant-factor load balance thus implies $\Theta(\log n)$ maximum degree and maximum congestion. This is illustrated in Figure 8. Note that the reduced congestion in the heterogeneous case results from reduced route length, which we cover in Section V-C.

B. Normalized Degree

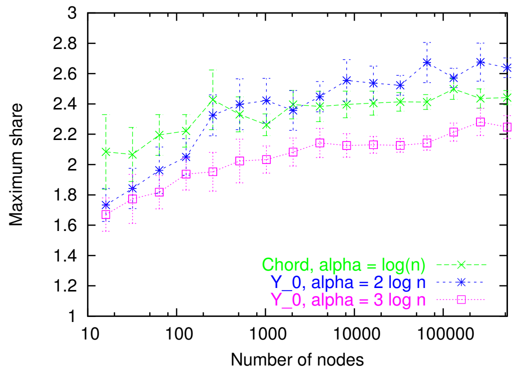
In this section we evaluate the effectiveness of our technique at maintaining low average normalized degree, defined in Section II-B. We average over all nodes that were not discarded. Figure 9(a) shows this metric as a function of α with homogeneous capacities and $n = 2048$. We see essentially no increase in degree in Y_0 until $\alpha > 2\log n \approx 22$. This is because when $\alpha \leq 2\log n$, the additional links associated with the virtual servers are to essentially the same set of nodes with which we are already linked due to the $2\log n$ incoming successor links. Even when $\alpha > 2\log n$, Theorems 5 and 6 imply that Y_0's degree is $O(\alpha)$ rather than $O(\alpha \log n)$ as in Chord.

Due to the normalization by capacity in our degree metric, Y_0's improved load balance gives it a lower average normalized degree than Chord in the heterogeneous case. This is depicted in Figure 9(b).

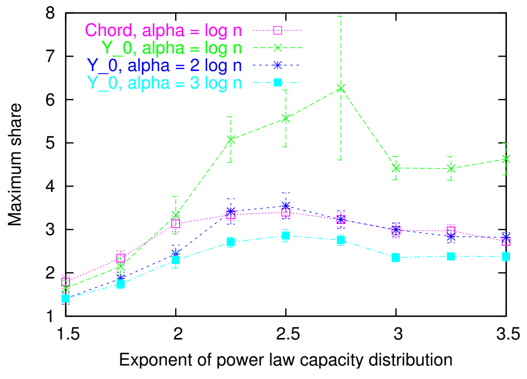
C. Route Length

To measure route length, we have each node route to a random ID, and average over the n resulting hop counts. Figure 10(a) shows decreasing route lengths in the power law distribution as the power law exponent decreases, with slightly greater benefit in Y_0 than in Chord. Note that large power law exponents approach a homogeneous system.

Figure 10(b) examines the asymptotics of route length. In a homogeneous system, Y_0 has very slightly higher route length

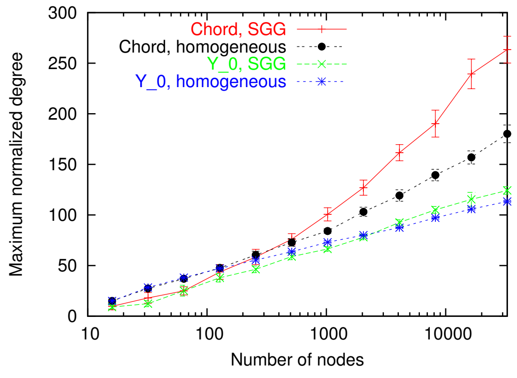


(a) Maximum share vs. n with homogeneous capacities. Chord with $\alpha = 1$ (not shown) increases to a maximum share of ≈ 13.7.

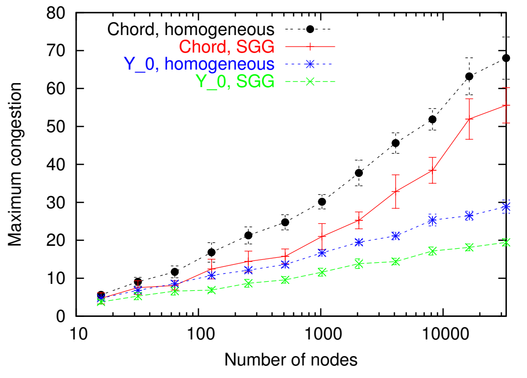


(b) Maximum share vs. capacity distribution in a 16,384-node system.

Fig. 7. Maximum share

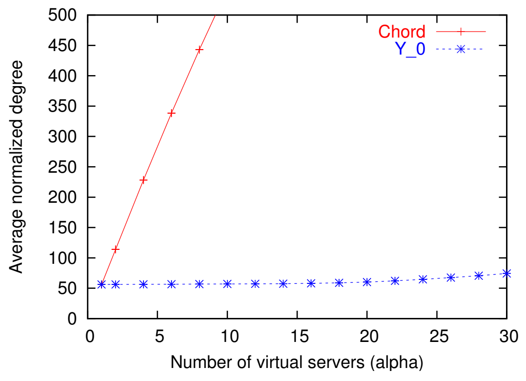


(a) Maximum normalized degree vs. n, with $\alpha = 1$ in Chord and $\alpha = 2 \log n$ in Y_0.

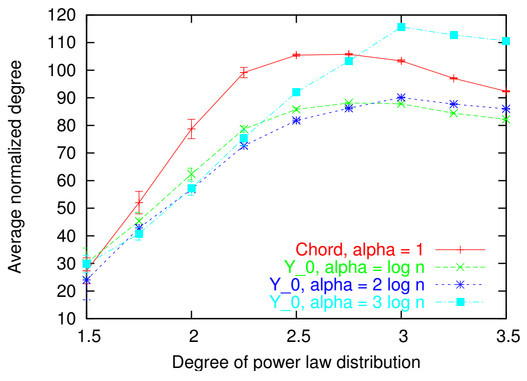


(b) Maximum congestion vs. n, with $\alpha = 1$ in Chord and $\alpha = 2 \log n$ in Y_0.

Fig. 8. Load balance of overlay links and routing load



(a) Average normalized degree vs. number of virtual servers in a homogeneous 2048-node system.



(b) Average normalized degree vs. capacity distribution in a 16,384-node system.

Fig. 9. Average normalized degree

than Chord, but as we showed in Theorem 7 this is only an additive constant. In the SGG distribution, both DHTs see significantly decreased route length. Chord's benefit is primarily due to the fact that there are fewer nodes in the ring: roughly 75% of the nodes in this distribution had capacity less than $\gamma_d = \frac{1}{2}$ and were discarded. This can only decrease route length by an additive constant, but it is significant in small- to medium-sized systems. In Y_0, our technique of increasing the number of fingers at high-capacity nodes provides a constant-factor decrease in route length (note the lower slope). A least-squares fit of our data for networks with 1024 or more nodes yields the following estimated asymptotic route lengths ($i.e.$, we ignore additive terms):

DHT	Capacities	Route length	Normalized
Chord	Homogeneous	$0.450 \log_2 n$	100%
	SGG	$0.348 \log_2 n$	77%
Y_0	Homogeneous	$0.472 \log_2 n$	105%
	SGG	$0.203 \log_2 n$	45%

VI. Related Work

ID space balance for homogeneous DHTs. The simplest way to balance load, under the uniform load assumption, is to obtain a $O(1)$ maximum share. The simplest way to do that is for each node to maintain $\Theta(\log n)$ virtual servers [18], [33]. Since this increases node degree by a factor of $\Theta(\log n)$, multiple proposals followed which give each node just one ID, but need to select and update that ID intelligently. All the proposals of this type are similar in that they consider $\Theta(\log n)$ locations for a node and select the one which gives the best load balance. They differ in which locations they check and when.

In Naor and Weider [26], a node checks $\Theta(\log n)$ random IDs when joining, and joins at the one which gives it the largest share. They show that this produces a maximum share of 2 if there are no node deletions. Handling node deletions requires an additional protocol, not precisely specified or analyzed in [26], whereby nodes are divided into groups of $\Theta(\log n)$ nodes and periodically reposition themselves within each group. In Adler et al [1], a joining node contacts a random node v already in the DHT, and splits in half the largest interval owned by one of v's $O(\log n)$ overlay neighbors. This results in an $O(1)$ maximum share. A simple deletion protocol was given but not analyzed.

Manku's algorithm [22] has a joining node pick a random node v and split in half the largest interval owned by one of the $\Theta(\log n)$ nodes adjacent to v in the ID space. This achieves a maximum share of 2 while moving at most a single extra node's ID for each node arrival or departure. It extends to balancing within a factor $1 + \varepsilon$ but moves $\Theta(1/\varepsilon)$ IDs. In contrast, the results of Section IV-B imply that Y_0 moves *no* extra IDs, even while achieving a maximum share of $1 + \varepsilon$, as long as n and the average capacity are relatively stable. The algorithm is more complicated than Y_0 and requires assignment of IDs to arbitrary locations in the ID space, so we cannot use the security mechanism of requiring a node's ID to be one of several hashes of its IP address [9].

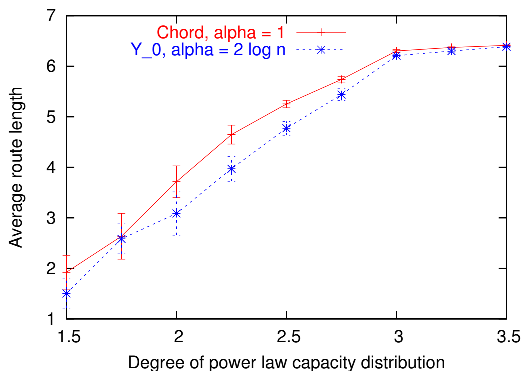
In one simple algorithm of Karger and Ruhl [20], each node has a fixed set of $O(\log n)$ possible IDs (so the security technique can be employed) and periodically reselects among them. This has maximum share $2 + \varepsilon$, but it requires reassignment of $O(\log \log n)$ IDs per arrival or departure in expectation. Bienkowski et al [2] later gave a similar algorithm which reduces the number of reassignments to a constant, but they show only $O(1)$ maximum share. A second algorithm of [20] adapts to uneven distributions of objects in the ID space (which Y_0 and the previously mentioned algorithms cannot), but requires unrestricted ID selection and a special overlay topology if the object distribution is very skewed, and its maximum share is 64.

ID space balance for heterogeneous DHTs. Comparatively few schemes handle node heterogeneity. Dabek et al [9] suggested that each physical node have a number of virtual servers proportional to its capacity, which we have developed into Basic-VSS in Section II-C. Surana et al [34], [12] balance load dynamically by transferring virtual servers between physical nodes. This approach can handle node heterogeneity and load which is not distributed uniformly in the ID space, and was shown through simulation to produce a very good balance. But it is more complicated than Y_0, cannot employ the aforementioned security mechanism, cannot take advantage of heterogeneity to reduce route length as much as Y_0, and has higher degree due to its use of multiple virtual servers per node.

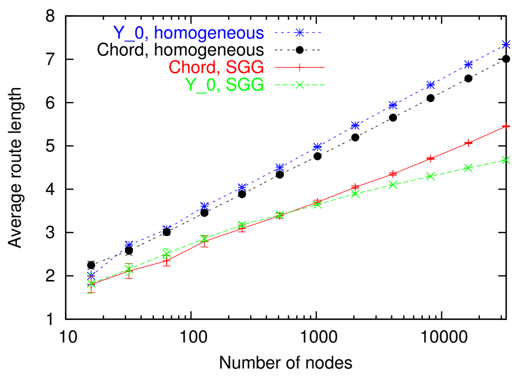
In a DHT-related work, Brinkmann et al [3] develop two schemes which divide an ID space fairly among a set of nodes of heterogeneous capacities, providing efficient ID lookup and node join and leave algorithms. However, they assume a centralized system with no overlay network. Their SHARE strategy is very similar to our LC-VSS: in both, each node selects an interval of the ID space of size $\Theta(\frac{\log n}{n})$, and ownership of a segment is "shared" among the nodes whose intervals overlap that segment. However, they employ this technique to handle nodes of very low capacity. If they were willing to discard low-capacity nodes as we do, the trivial Basic-VSS scheme of Section II-C would be acceptable. In contrast, we cluster a node's IDs in order to share overlay links. Moreover, the way in which the ID space sharing is performed in [3] is more complicated than in our scheme; notably, nodes need $\Theta(\log^2 n)$ IDs, rather than $\Theta(\log n)$.

Load balance by object reassignment. The above strategies balance load by changing the assignment of IDs to nodes. Another approach is redirection: store a pointer from an object's ID to the arbitrary node currently storing it. This can balance the load of storing and serving data, but not load due to routing or maintenance of the overlay — and if objects are small, routing dominates the bandwidth and latency of storing and finding an object. It also requires maintenance of the pointers and adds one hop to each lookup.

Karger and Ruhl [19] can handle heterogeneous capacities and obtain a constant-factor load balance. Each node periodically contacts another, and they exchange objects if one node's load is significantly more than the other's. But their bound



(a) Route length vs. capacity distribution in a 16,384-node system.



(b) Route length vs. n. In Chord, $\alpha = 1$ and in Y_0, $\alpha = 2 \log n$.

Fig. 10. Route length

on movement cost depends on the ratio of the maximum and minimum node capacities. Byers et al [4], [5] use a "power of two choices" approach, hashing an object to $d \geq 2$ IDs and storing it on the corresponding node with least load, which results in a maximum load of $\log \log n / \log d + O(1)$. Swart [35] places object replicas on a lightly-loaded subset of nodes in Chord's successor list. Neither [4], [5], nor [35] provide results for the heterogeneous case.

Exploiting heterogeneity in P2P systems. Ratnasamy et al. [28] posed the question of whether heterogeneity could be leveraged to improve routing performance in DHTs. Nearly all the DHTs which have followed up on this suggestion use a two-level hierarchy, dividing nodes into a set of high-capacity "superpeers" and low-capacity peers. Mizrak et al. [25] build a clique of $\sqrt{n}$ superpeers, each with $\sqrt{n}$ low-capacity peers attached. This produces a graph of constant diameter but requires that superpeers have polynomially many neighbors. Its scalability, particularly with respect to maintenance traffic, was not demonstrated for a wide range of capacity distributions. Zhao et al. [38] and Garces-Erice et al. [11] organize peers into groups based on network locality. Each group chooses a superpeer based on reliability, bandwidth, or latency, and the superpeers across all groups form a DHT. We argue that these two-level techniques cannot adapt to or take full advantage of arbitrary node capacity distributions, because peers within each of the two classes are treated equally. In fact these schemes are complementary to ours: peers at each level of the hierarchy are likely to have nonuniform capacities, so our techniques could be applied to the DHTs used at each level to further improve performance.

To the best of our knowledge, the only heterogeneity-aware DHT designs not employing a two-level hierarchy, other than Y_0, are SmartBoa [15] and Xu et al [37]. In SmartBoa, nodes are divided into up to 128 levels based on capacity, and a node at level k maintains roughly $n/2^k$ neighbors. A heterogeneity-aware multicast algorithm allows efficient dissemination of node join or leave events, so nodes need not continually ping their neighbors, enabling much larger routing tables and thus lower route length than traditional DHTs. However, SmartBoa does not adapt the ID space partitioning ($i.e.$, object storage load) to nodes' capacities, nor was it evaluated experimentally or theoretically. Xu et al [37] adapt the DHT topology to fit the underlying network connectivity, which may be heterogeneous, thus obtaining low stretch. But the ID space is again not adapted to node capacity.

In the world of unstructured P2P systems, Freenet's Next Generation Routing [8] employs heuristics to route messages to nodes with observed faster responses. Chawathe et al [7] propose an unstructured Gnutella-like system which adapts topology, flow control, and data replication to nodes' capacities, and found that their system performed significantly better in a heterogeneous environment. The techniques used in these systems, and the services they provide, are quite different than those of Y_0.

VII. Conclusion

We have proposed a scheme to assign IDs to virtual servers, called Low Cost Virtual Server Selection (LC-VSS), that yields a simple DHT protocol, called Y_0, for which node degree does not increase significantly with the number of virtual servers. Y_0 adapts to heterogeneous node capacities, can achieve an arbitrarily good load balance, moves little load, and can compute a node's IDs as $O(\log n)$ hashes of its IP address for security purposes. The techniques behind Y_0 generalize to arbitrary overlay topologies while providing some flexibility in neighbor selection, even if the underlying topology did not. We demonstrated the effectiveness of our techniques through simulation, showing a maximum share less than 3.6 in a range of capacity distributions with no greater degree than in Chord with a single virtual server.

Y_0 has several drawbacks. It uses $\Theta(\log^2 n)$ memory per node, but we expect this will be acceptable in nearly all circumstances. If a particularly good balance is desired, the number of links needed to maintain the ring structure of the

DHT increases by a constant factor. Node join and leave operations will be slowed somewhat by the fact that a node owns $\Theta(\log n)$ virtual servers, although the overlay links need to be constructed only once. Perhaps most significantly, Y_0 requires good estimates of n and the average capacity, and a good balance is guaranteed only under the uniform load assumption.

We also illustrated the potential of taking heterogeneity into account, with route lengths in our DHT in a real-world capacity distribution less than half those of a homogeneous distribution due to our adaptation of the Chord overlay topology to a heterogeneous environment. An interesting open problem is to study the effect of heterogeneity in other DHT topologies, such as de Bruijn graphs [17], [26]. More generally, we hope that this work will motivate further exploration of how heterogeneity can benefit distributed systems.

Acknowledgments

We thank Stefan Saroiu, Krishna Gummadi, and Steven Gribble for supplying the Gnutella bandwidth data, and David Molnar and Lakshminarayanan Subramanian for helpful comments on a draft of this paper. This material is based upon work supported under a National Science Foundation Graduate Research Fellowship.

References

[1] M. Adler, E. Halperin, R. M. Karp, and V. Vazirani. A stochastic process on the hypercube with applications to peer-to-peer networks. In *Proc. STOC*, June 2003.

[2] M. Bienkowski, M. Korzeniowski, and F. M. auf der Heide. Dynamic load balancing in distributed hash tables. Manuscript, 2004.

[3] A. Brinkmann, K. Salzwedel, and C. Scheideler. Compact, adaptive placement strategies for non-uniform capacities. In *Proc. ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, Winnipeg, Canada, 2002.

[4] J. Byers, J. Considine, and M. Mitzenmacher. Simple Load Balancing for Distributed Hash Tables. In *Proc. IPTPS*, Feb. 2003.

[5] J. Byers, J. Considine, and M. Mitzenmacher. Geometric generalizations of the power of two choices. In *Proc. SPAA*, 2004.

[6] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. SplitStream: High-bandwidth content distribution in a cooperative environment. In *Proc. of IPTPS*, February 2003.

[7] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker. Making Gnutella-like P2P systems scalable. In *Proceedings of ACM SIGCOMM*, 2003.

[8] I. Clarke. Freenet's next generation routing protocol, 2003. http://freenet.sourceforge.net/index.php?page=ngrouting.

[9] F. Dabek, F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Wide-area cooperative storage with CFS. In *Proc. ACM SOSP*, 2001.

[10] F. Dabek, J. Li, E. Sit, F. Kaashoek, R. Morris, and C. Blake. Designing a DHT for Low Latency and High Throughput. In *Proc. NSDI*, 2004.

[11] L. Garces-Erice, E. W. Biersack, K. W. Ross, P. A. Felber, and G. Urvoy-Keller. Hierarchical P2P systems. In *Proceedings of ACM/IFIP International Conference on Parallel and Distributed Computing (Euro-Par)*, Klagenfurt, Austria, 2003.

[12] B. Godfrey, K. Lakshminarayanan, S. Surana, R. Karp, and I. Stoica. Load balancing in dynamic structured P2P systems. In *Proc. IEEE INFOCOM*, Hong Kong, 2004.

[13] P. B. Godfrey and I. Stoica. Heterogeneity and load balance in distributed hash tables (full version of this paper), 2004. Available at www.cs.berkeley.edu/~pbg/.

[14] K. Gummadi, R. Gummadi, S. Gribble, S. Ratnasamy, S. Shenker, and I. Stoica. The Impact of DHT Routing Geometry on Resilience and Proximity. In *Proc. ACM SIGCOMM*, 2003.

[15] J. Hu, M. Li, W. Zheng, D. Wang, N. Ning, and H. Dong. SmartBoa: Constructing p2p overlay network in the heterogeneous internet using irregular routing tables. In *Proc. IPTPS*, 2004.

[16] R. Huebsch, J. Hellerstein, N. Lanham, B. T. Loo, S. Shenker, and I. Stoica. Querying the Internet with PIER. In *Proc. of VLDB*, September 2003.

[17] M. F. Kaashoek and D. R. Karger. Koorde: A simple degree-optimal distributed hash table. In *Proc. 2nd International Workshop on Peer-to-Peer Systems (IPTPS)*, 2003.

[18] D. Karger, E. Lehman, T. Leighton, M. Levine, D. Lewin, and R. Panigrahy. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web. In *ACM Symposium on Theory of Computing*, pages 654–663, May 1997.

[19] D. Karger and M. Ruhl. New Algorithms for Load Balancing in Peer-to-Peer Systems. Technical Report MIT-LCS-TR-911, MIT LCS, July 2003.

[20] D. Karger and M. Ruhl. Simple Efficient Load Balancing Algorithms for Peer-to-Peer Systems. In *Proc. SPAA*, 2004.

[21] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao. OceanStore: An Architecture for Global-scale Persistent Storage. In *Proc. ASPLOS*, Boston, MA, November 2000.

[22] G. Manku. Balanced binary trees for ID management and load balance in distributed hash tables. In *Proc. PODC*, 2004.

[23] G. Manku, M. Bawa, and P. Raghavan. Symphony: Distributed hashing in a small world, 2003.

[24] MetaMachine. eDonkey, Accessed July 2004. http://www.edonkey2000.com/.

[25] A. T. Mizrak, Y. Cheng, V. Kumar, and S. Savage. Structured superpeers: Leveraging heterogeneity to provide constant-time lookup. In *Proc. IEEE Workshop on Internet Applications*, 2003.

[26] M. Naor and U. Wieder. Novel architectures for P2P applications: the continuous-discrete approach. In *Proc. SPAA*, June 2003.

[27] S. Nath, P. B. Gibbons, S. Seshan, and Z. R. Anderson. Synopsis diffusion for robust aggregation in sensor networks. In *ACM SenSys*, 2004.

[28] S. Ratnasamy, S. Shenker, and I. Stoica. Routing algorithms for DHTs: Some open questions. In *Proc. IPTPS*, 2002.

[29] A. Rowstron and P. Druschel. Pastry: Scalable, Distributed Object Location and Routing for Large-scale Peer-to-Peer Systems. In *Proc. Middleware*, 2001.

[30] S. Saroiu. Private communication, 2003.

[31] S. Saroiu, P. K. Gummadi, and S. D. Gribble. A Measurement Study of Peer-to-Peer File Sharing Systems. In *Proc. MMCN*, San Jose, CA, USA, January 2002.

[32] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana. Internet indirection infrastructure. In *Proc. ACM SIGCOMM*, August 2002.

[33] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of ACM SIGCOMM'01*, pages 149–160. ACM Press, 2001.

[34] S. Surana, B. Godfrey, K. Lakshminarayanan, R. Karp, and I. Stoica. Load balancing in dynamic structured P2P systems. In *Performance Evaluation: Special Issue on Performance Modeling and Evaluation of Peer-to-Peer Computing Systems*, 2005.

[35] G. Swart. Spreading the load using consistent hashing: A preliminary report. In *International Symposium on Parallel and Distributed Computing (ISPDC)*, 2004.

[36] M. Walfish, H. Balakrishnan, and S. Shenker. Untangling the web from DNS. In *Proc. Symposium on Networked Systems Design and Implementation (NSDI)*, 2004.

[37] Z. Xu, M. Mahalingam, and M. Karlsson. Turning heterogenity into an advantage in overlay routing. In *Proc. IEEE INFOCOM*, 2003.

[38] B. Zhao, Y. Duan, L. Huang, A. Joseph, and J. Kubiatowicz. Brocade: Landmark routing on overlay networks, 2002.

[39] S. Q. Zhuang, B. Y. Zhao, A. D. Joseph, R. H. Katz, and J. D. Kubiatowicz. Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination. In *Proceedings of NOSSDAV*, June 2001.