A Scheme for Load Balancing in Heterogeneous Distributed Hash Tables*

George Giakkoupis
ggiak@cs.utoronto.ca

Vassos Hadzilacos
vassos@cs.utoronto.ca

Department of Computer Science
University of Toronto
Toronto, Canada

ABSTRACT

We present a scheme for evenly partitioning the key space in distributed hash tables among the participating nodes. The scheme is based on the multiple random choices paradigm [3, 19], and handles both node joins and leaves. It achieves, with high probability, a ratio of at most 4 between the loads of the most and least burdened nodes, in the face or arbitrary node arrivals and departures. Each join or leave operation incurs message cost that is, with high probability, $O(\log^2 n)$, where n is the number of nodes, and causes the relocation of keys from at most one node (for joins) or three nodes (for leaves). A version of our scheme is suitable for heterogeneous systems, where the capacities of nodes to serve keys can vary widely.

Categories and Subject Descriptors

C.2.4 [**Distributed Systems**]: Distributed applications—*load balancing*; G.3 [**Probability and Statistics**]: Stochastic processes; E.1 [**Data Structures**]: Distributed data structures

General Terms

Algorithms, Performance, Theory

Keywords

Peer-to-peer, distributed hash tables, load balancing

1. INTRODUCTION

Distributed Hash Tables (DHTs) [17, 21, 5, 20, 10, 14, 16, 1, 13] are data structures used to organize highly dynamic, massive, decentralized distributed systems, such as peer-to-peer networks. A DHT maps keys (i.e., resource identifiers) into a hash space, which is partitioned among the nodes in the network. Each node is responsible for (the resources whose keys are hashed into) its segment of the hash space. When a node *joins* the system, the partition of the hash space is perturbed slightly, so that the new node can be given its own segment of the hash space, pieces of which previously belonged to a small number of old nodes. Similarly, when a node *leaves* the system, its segment is redistributed among a small number of the remaining nodes. In addition to the join and leave operations, a DHT supports the *lookup* operation which locates the node responsible for (the resources identified by) a given key.

To enable the implementation of these operations, each node maintains a number of links to other nodes in the system, thereby forming an overlay network. Because of the massive scale of the networks for which DHTs are intended, the overlay network must have low degree and the lookup, join and leave operations must have cost (e.g., in terms of messages and time) that is at most polylogarithmic in the number of nodes in the system.

An important goal in the design of DHTs is to achieve a balanced partition of the hash space among the nodes in the system. It is often desirable that each node assumes responsibility for a portion of the hash space that is proportional to its power (measured in terms of its processor speed, available bandwidth, and/or storage capacity), and that this property is maintained as nodes join and leave the system. This is because, typically, the number of keys a node is responsible for serving, and the amount of routing traffic the node handles are proportional to the size of the segment that the node is associated with.

We quantify the notion of load balance in DHTs in the following natural way. Assume each node has a positive *weight*, that is proportional to its power, i.e., for any nodes n_1, n_2 of weights respectively w_1, w_2, with $w_1 \geq w_2$, n_1 can manage a segment of the hash space that is w_1/w_2 times larger than that n_2 can handle. The *weighted size* of a node's segment is the quotient of the segment's size divided by the node's weight. Finally, the *balance* ρ of a DHT structure is the ratio of the maximum weighted size of any node segment to the minimum weighted size. Note that if all nodes have the same weight, then ρ reduces to the usual measure of load balance in DHTs: the ratio of maximum segment size to minimum.

In this paper, we describe a simple and intuitive protocol for managing the partition of a DHT's hash space into node segments, as nodes join and leave the system. The proto-

*Research supported in part by the Natural Sciences and Engineering Council of Canada.

PODC'05, July 17–20, 2005, Las Vegas, Nevada, USA.

col can be used in any DHT structure whose hash space is a line segment or a ring. It employs the multiple random choices paradigm [3, 19], and it has two versions. The first (unweighted) version can be roughly described as follows. To accommodate the addition of a new node, we select *at random* a (logarithmic in the system size) number of points in the hash space,[1] split in half the largest node segment containing at least one of the points selected, and assign one half to the newly arrived node. To cope with the departure of a node, again we choose a logarithmic number of points *at random*, we merge with an adjacent segment the shortest segment containing some of the points selected, and assign the segment of the departing node to one of the two nodes whose segments were merged. The second (weighted) version of the protocol is a natural extension of the above scheme that assumes node weights that are powers of 2, up to a fixed maximum.

The unweighted version of the protocol guarantees that $\rho \leq 4$ with high probability, for any sequence of node joins and/or leaves. Both join and leave operations have message complexity $\mathrm{O}\big(\log n \cdot L_m(n)\big)$, and (parallel) time complexity $L_t(n) + \mathrm{O}(1)$, where n is the system size, and $L_m(n)$, $L_t(n)$ are respectively the message complexity and time complexity of a lookup request. (Typically $L_m(n)$ and $L_t(n)$ have the same asymptotic behavior and are $\Theta(\log n)$ [21, 5, 20, 10], or $\Theta(\log n / \log\log n)$ [6, 4, 11], with high probability.) Finally, a join affects only the segment of a singe node, and a leave affects at most two (adjoining) node segments and the segment of the departing node. The weighted version guarantees that $\rho \leq 8$, with high probability, and has the same (asymptotic) join/leave complexity as the unweighted version. Each join or leave operation incurs only a few localized changes in the partition of the hash space, i.e., at most a constant number of adjacent node segments are modified.

Related Work

In most early DHT structures [21, 10, 13, 16], each node (upon arrival) chooses *at random* a point in the hash space (typically, the unit interval $[0, 1)$), and becomes associated with the points of the hash space closest to the selected point (with respect to some distance function). Assuming random node departures, this scheme guarantees that the ratio of largest to *average* node segment size is $\Theta(\log n)$, with high probability [7, 21]. However, we can show that the ratio of average to smallest segment size is $\Omega(n)$ with constant probability. Therefore, assuming homogeneous nodes, ρ is $\Omega(n \log n)$, with constant probability.

In another early approach [17], a new node splits in half the node segment containing a randomly selected point, and assumes responsibility of one half. In the pure join model, the balance of this scheme is better than the previous one, but we can show that ρ is still $\Omega(\log^2 n / \log\log n)$, with high probability. (The ratio of largest to average segment size is the same as before, while the ratio of average to smallest is $\Omega(\log n / \log\log n)$, with high probability.) In both this and the previous approaches, if every physical node acts as $\Omega(\log n)$ *virtual nodes*, each associated with a distinct segment, then constant ρ is achieved, with high probability [7, 21]. However, this approach amplifies by a factor of $\Omega(\log n)$ the number of links a node should maintain, and (consequently) increases the complexity of join and leave operations.

Five recently proposed load balancing schemes guarantee constant ρ, with high probability, for homogeneous nodes, while assigning a single segment per node. Abraham et al. [1] and Naor and Wieder [16] independently suggested and analyzed the join scheme our (unweighted) protocol uses. However, neither work provides a method for departures that provably achieves constant ρ. Adler et al. [2] propose and analyze a scheme for joins in a hypercubic overlay structure. Roughly, a new node splits in half the longest segment among the ones associated with the node responsible for a randomly selected point, and its $\Theta(\log n)$ neighbors in the overlay structure. They also describe a procedure for leaves, but they do not show it maintains constant ρ, although experimental results suggest that it may do so. The message and (parallel) time complexity of node joins are $\Theta\big(L_m(n) + \log n\big)$ and $L_t(n) + \mathrm{O}(1)$, with high probability, respectively, and the number of nodes affected is constant.

The next two approaches are not tied to specific overlay structures, and provably guarantee constant ρ in the face of both joins and leaves. Karger and Ruhl [8] present a scheme with message and time complexity respectively $\Theta\big(\log n \cdot L_m(n)\big)$ and $L_t(n) + \mathrm{O}(\log n)$, with high probability. The number of nodes affected by a single operation is $\mathrm{O}(\log n)$, with high probability, and $\Theta(\log\log n)$, on expectation. Manku [12] proposes a very efficient scheme where operations are, roughly, performed as in our protocol, but instead of random segments the scheme examines a logarithmic number of consecutive segments in the neighborhood of a random point. The message and time complexity are respectively $L_m(n) + \Theta(\log n)$ and $L_t(n) + \Theta(\log n)$, with high probability, and constant number of node segments are modified per operation.

Finally, in a very recent paper Kenthapadi and Manku [9] present a load balancing scheme that combines two different approaches: the multiple random choices scheme, and the "neighborhood search scheme" of [12]. Their protocol is overlay independent, and provably guarantees constant ρ only in the pure join model. The complexity of joins depends on two system parameters r and v, with $rv = \Theta(\log n)$, where r represents the number of random points selected, and v represents the number of (consecutive) segments examined in the neighborhood of each random point. The message and time complexity per operation are respectively $rL_m(n) + rv$ and $L_t(n) + v$, with high probability.

All the above approaches assume homogeneous nodes. To our knowledge, the problem of load balancing in DHTs of heterogeneous nodes has not, so far, received special attention. A reason may be that it can be easily reduced to the load balancing problem for homogeneous nodes, by having each (heterogeneous) node run a number of virtual nodes proportional to its power. However, to paraphrase the apt statement of Ratnasamy et al. [18] (made, in their case, about routing rather than load balancing) "instead of merely coping with heterogeneity, [load balancing algorithms] could instead use it to their advantage."

In particular, a balancing scheme that assigns a single (contiguous) segment per node, would result in significant reduction in the number of links in the network compared to the approach that uses virtual nodes. For instance, in a system where half of the nodes are twice as powerful as the other half, a scheme that assigns a single contiguous segment

[1]Throughout this paper, whenever we say "*at random*", we imply "independently and uniformly at random".

protocol	message complexity	time complexity	nodes reloc.	comments
Abraham et al. [1]	$\Theta(\log n \cdot L_m(n))$	$L_t(n) + \mathrm{O}(1)$	$\mathrm{O}(1)$	no departures
Naor and Wieder [16]	$\Theta(\log n \cdot L_m(n))$	$L_t(n) + \mathrm{O}(1)$	–	no proof for departures
Adler et al. [2]	$\Theta(L_m(n) + \log n)$	$L_t(n) + \mathrm{O}(1)$	$\mathrm{O}(1)$	no proof for departures; overlay specific
Karger and Ruhl [8]	$\Theta(\log n \cdot L_m(n))$	$L_t(n) + \mathrm{O}(\log n)$	$\mathrm{O}(\log n)$	–
Manku [12]	$\Theta(L_m(n) + \log n)$	$L_t(n) + \Theta(\log n)$	$\mathrm{O}(1)$	–
Kenthapadi and Manku [9]	$rL_m(n) + rv$	$L_t(n) + v$	$\mathrm{O}(1)$	where $rv = \Theta(\log n)$; no proof for departures
this paper	$\Theta(\log n \cdot L_m(n))$	$L_t(n) + \mathrm{O}(1)$	$\mathrm{O}(1)$	works with heterogeneous nodes

Table 1: Comparison of schemes for load balancing in DHTs.

per node would result in the more powerful nodes having (roughly) half the number of outgoing links and the same number of incoming links as in the virtual nodes approach, while the less powerful nodes would have the same number of outgoing and one third fewer incoming links.

Our Contribution

The idea of employing the multiple random choices paradigm [3, 19] to achieve load balancing in DHTs is not novel. As we saw above, the join protocol we use was proposed by two other groups [1, 16]. However, they didn't provide a protocol for handling node leaves that maintains constant ρ. To our knowledge, our work is the first to propose and analyze a scheme that uses the multiple random choices scheme for both node joins and leaves, and achieves constant ρ. Compared to the other two algorithms that handle both joins and leaves, it has similar complexity, but relocates fewer nodes per operation than [8], and it has worse message complexity, but slightly better parallel time complexity than [12]. Finally, our protocol is the only one we know of that can handle heterogeneous nodes, achieving constant ρ (independent of nodes' relative power) and assigning to each node a single contiguous segment.

The rest of this extended abstract is organized as follows. In Section 2 we present the unweighted and weighted versions of our load balancing protocol, and discuss the pertinent properties of each. In Section 3 we present a detailed overview of the analysis of the unweighted version. Underlying our protocols are complex stochastic processes whose rigorous analysis is quite lengthy; thus, our goal here is to explain the main ideas of the analysis, without providing all the detailed calculations. In Section 4 we describe, sketchily, the analysis of the weighted version of the protocol.

2. PROTOCOLS FOR BALANCED PARTITIONING

2.1 A Note on the Model

The load balancing protocols we present in the next two sections are general enough to be used in any DHT design whose hash space is a line segment or a ring. For concreteness we will assume that the hash space is the unit interval $I = [0, 1)$. For efficiency, we will also assume that every node maintains links to its immediate successor and predecessor nodes in the hash space. As is typical in the analysis of load balancing algorithms for DHTs, we only consider the case where node joins and leaves occur sequentially. We expect the protocols should perform well up to some degree of concurrency, as well.

2.2 S&M Protocol

This scheme assumes that all nodes are equally powerful, and aims to partition the hash space evenly among them. Node joins and leaves are performed in such a way that, at any time, the length of the segment associated with each node is an integer power of 1/2 (potentially a different one for different nodes), and its endpoints are integer multiples of its length. We call such a segment of length $1/2^d$, for some $d \in \mathbb{N}$, a *d-segment*; we also say that this segment has *depth* d. Finally, we define the *sibling* of a d-segment λ, denoted $\overline{\lambda}$, to be the unique d-segment such that $\lambda \cup \overline{\lambda}$ is a $(d-1)$-segment.

Protocol Description

To *join* the system, a new node n, first requests the depth of the segment associated with a node n_0 that is known to be in the system already. Let d be that depth. Then, n issues (via n_0) lookup requests for each of $\lceil \alpha^+ d + \beta^+ \rceil$ keys selected *at random*, where α^+ and β^+ are positive system-wide parameters. Let n' be a node associated with a *longest* segment, among those returned by the lookup requests. Then, the segment of n' is split into two halves, and each of n, n' is associated with one half.

The *leave* procedure is, in a sense, symmetric to the join process. Suppose some node n, who is currently in the system and is associated with a d-segment λ, $d > 0$, wishes to depart. Before n does do, it looks up the nodes that are responsible for $\lceil \alpha^-(d+1) + \beta^- \rceil$ keys selected *at random*, where α^-, β^- are again positive system parameters. Let n' be a node associated with a *shortest* segment among the looked up nodes, and let λ' denote that segment. We have two cases, depending on the length $|\lambda'|$ of λ'.

(a) $|\lambda'| < |\lambda|$: If $\overline{\lambda'}$ is associated with a single node, say n'', (i.e., $\overline{\lambda'}$ is not currently split), then $\overline{\lambda'}$ and λ' are merged, n'' and n' are associated with the resulting segment and λ, respectively, and n leaves the system. If $\overline{\lambda'}$ is not associated with a single node (i.e., $\overline{\lambda'}$ is currently split), then n traverses (sequentially) the nodes associated with subsegments of $\overline{\lambda'}$, until it finds a pair of nodes n_1, n_2 associated with two sibling segments. (Note that we can always find such a pair of nodes.) Then, the two sibling segments are merged, n_1 and n_2 become associated with the newly created segment and λ, respectively, and n departs from the system.

(b) $|\lambda'| \geq |\lambda|$: The procedure is almost identical to that described in case (a) if we consider n in place of n'. So, we first attempt to merge λ with $\overline{\lambda}$, and if this is not possible (because $\overline{\lambda}$ is split) we merge two sibling node segments that are subsegments of $\overline{\lambda}$.

Protocol Properties

The above protocol provides strong load balancing guarantees. Roughly speaking, if we start from a sufficiently balanced initial state, then at all times during an arbitrary sequence of k joins and/or split operations, $\rho \leq 4$, with probability $1 - \mathrm{O}(k/N^b)$, where N is the minimum number of nodes in the system during this sequence, and $b > 0$ a constant that can be made arbitrarily large by choosing large enough parameters $\alpha^+, \beta^+, \alpha^-, \beta^-$ (independently of k or N). A more formal statement of this property, and an outline of its proof are presented in Section 3. We note parenthetically that the protocol is "self-correcting" in the sense that starting from an arbitrary (valid) state, a sufficiently balanced state is reached after a large enough number of steps. We will elaborate on this issue in the full version of this paper.

It is easy to see that the message complexity of a join operation in an n-node system with constant ρ is $\mathrm{O}\big(\log n \cdot L_m(n)\big)$. This is not optimal since other approaches require as little as $\mathrm{O}(\log n) + L_m(n)$ messages [12]. However, if the probing lookups are executed in parallel (instead of sequentially), the time complexity of both join and leave procedures reduces to $\mathrm{O}(1) + L_t(n)$. Finally, a node join affects only the segment associated with a singe node, and a node leave affects the segments associated with at most two nodes (in addition to the departing node). Thus, for constant ρ both operations result in relocating keys that fall in an $\mathrm{O}(1/n)$ fraction of the hash space, which is optimal.

2.3 Weighted S&M Protocol

The scheme we present in this section is an extension of the protocol described in Section 2.2 that takes into account the relative power of nodes. In particular, it presumes that each node n has a *weight*, denoted $w(n)$, that is an integer power of 2 between 1 and some system-wide parameter W (also a power of 2), and tries to partition the hash space in such a way that each node is associated with a segment of length proportional to its weight. Roughly speaking, this scheme arranges nodes into (virtual) groups, and uses the same technique as the unweighted S&M Protocol to achieve for each group what that protocol achieved for single nodes. Below we describe the protocol in more detail.

The hash space is partitioned into segments, one for each node, such that at any time there is a (unique) many-to-one mapping of the set of nodes to a set of *groups* with the following properties: (i) the union of the segments of all nodes in a group, called a *group segment*, is a d-segment, for some $d \geq 0$ (potentially, a different d for different groups); and (ii) the sum of the weights of all nodes in a group, called the group's *weight*, is between W and $2W - 1$.[2] We denote the weight of a group G by $w(G)$.

The details of how a group segment is partitioned among the nodes of the group are specified by a part of the balance algorithm we call the *group management protocol*. In addition, the group management protocol describes how to perform the group operations listed below. Note that, these operations may only modify segments associated with nodes of the groups mentioned in each operation.

[2]More precisely, this property applies only if the total weight of all nodes in the system is at least W. Otherwise, there is only one group that contains all the nodes.

T_1: Add node n to group G. If $w(G) + w(n) \geq 2W$, then G is split into two groups, each associated with one half of G's segment.

T_2: Given two groups G, G' associated with sibling segments, and a node n in G, remove n from the system. If $w(G) + w(G') - w(n) < 2W$, then G and G' are merged into a single group.

T_3: Given a group G, a node n in G, and two other groups G_1, G_2 associated with sibling segments, remove n from the system in such a way that either (i) the value of $w(G)$ after the operation is greater or equal to that before, or (ii) G is split into two groups. Moreover, G_1 and G_2 may be merged into a single group.

The group management protocol can be designed independently of the rest of the load balancing algorithm, as long as it adheres to the above specification. In the next two paragraphs, we first describe the Weighted S&M Protocol in detail assuming an arbitrary group management protocol, and then we discuss specific group management protocol designs.

Protocol Description

Besides the fact that it operates on groups, instead of single nodes, the weighted version of the S&M Protocol is almost identical to the unweighted one.[3] To join the system, a node n issues, via some known node n_0, lookup requests for $\lceil \alpha^+ d + \beta^+ \rceil$ keys selected *at random*, where d is the depth of the segment associated with n_0's group. Then, n adds itself (using operation T_1) to a group associated with a longest segment, among the groups of the nodes probed. Note that the segment of a group can be computed by traversing no more than $2W$ nodes.

Consider, now, a node n that belongs to a group G, and let G be associated with d-segment λ, $d \geq 0$. To leave the system, n first looks up the nodes responsible for $\lceil \alpha^-(d+1) + \beta^- \rceil$ keys, selected *at random*. Let G' be a group associated with a shortest segment, among the groups of the nodes probed, and let λ' be that segment. Similarly to the unweighted protocol, we have two cases.

(a) $|\lambda'| < |\lambda|$: If $\overline{\lambda'}$ is associated with a single group, say G'', then n is removed from the system by performing operation T_3 with parameters G, n, G', and G''. If $\overline{\lambda'}$ is not associated with a single group then n traverses the nodes associated with subsegments of $\overline{\lambda'}$, until it finds a pair of group G_1, G_2 associated with sibling segments. Then, n is removed from the system by performing operation T_3 with parameters G, n, G_1, and G_2.

(b) $|\lambda'| \geq |\lambda|$: If $\overline{\lambda}$ is associated with a single group, say G''', then n is removed by executing operation T_2 with parameters G, n, and G'''. Otherwise, n looks for a pair of groups G_3, G_4 associated with sibling subsegments of $\overline{\lambda}$, and leaves the system by performing operation T_3 with parameters G, n, G_3, and G_4.

Group Management

Recall that the group management protocol specifies how a group segment is distributed among the nodes in the group, and describes how operations T_1–T_3 are performed. We

[3]In fact, for $W = 1$ the two protocols are identical.

evaluate the efficiency of such a protocol in terms of the following measures: (i) the balance it achieves among the nodes *in a group*, denoted ϱ; (ii) the number of node segments affected (i.e., have one or both endpoints modified) in each of operations T_1–T_3 ; and (iii) the fraction of hash space whose association to nodes changes as a result of operations T_1–T_3. Ideally, we would like that ϱ be as close to 1 as possible, that operations T_1–T_3 (in most cases) affect a number of nodes that is at most proportional to the weight w of the node added/removed and independent of W, and that each operation changes the association to nodes in a fraction of the segment of each group involved in the operation that is at most proportional to w/W.

The simplest approach is to associate with each node a segment of length *exactly* proportional to the node's weight, i.e., a node of weight w, in a group of weight w' and (group) segment λ, is associated with a segment of length $|\lambda|w/w'$. Such a scheme is optimal in terms of (i), i.e., achieves $\varrho = 1$. However, operations T_1–T_3 modify the endpoints of *all* node segments in at least one (and at most three) groups, even when the weight w of the node added/removed is small. Finally, operations T_1–T_3 may result in changing the association to nodes of a constant fraction (i.e., independent of w and W) of a group segment. There are more complex ways to do group management that are better in terms of (ii) and (iii) while still keeping $\varrho \leq 2$, but these are omitted from here.

Protocol Properties

The protocol guarantees regarding the relative length of *group* segments are the same as those the unweighted protocol provides about the relative length of *node* segments. If, in addition, the group management protocol achieves $\varrho \leq 2$ then, with high probability, $\rho \leq 8$ during any sequence of joins/splits that starts from a sufficiently balanced state and uses large enough protocol parameters (α^+, β^+, α^-, β^-), provided that the number of nodes in the system does not become too small, and the sequence of operations is not too long.

The message complexity of a single node join/leave is $O\big(\log n \cdot (L_m(n) + W)\big)$. (Note that if we use the simple group management protocol we saw in the previous paragraph, the message complexity per operation becomes $O\big(\log n \cdot L_m(n) + W\big)$, because with that specific protocol a node can determine the length of its group segment based on its own node segment.) If the probing lookups are executed in parallel, though, the resulting time complexity is $L_t(n) + O(W)$. A node join affects only the segments of nodes in a singe group, while a leave affects nodes associated with at most three groups. So, in both cases the number of nodes affected is $O(W)$. Finally, each join/leave changes the association to nodes of a constant fraction of each group segment affected by the operation. Thus, if ρ is constant, at most an $O(W/n)$ fraction of the hash space is affected.

3. ANALYSIS OF S&M PROTOCOL

3.1 Binary Partitions

We say that a partition of the unit interval I into segments is a *binary partition*, if each segment is a d-segment, for some $d \geq 0$ (possibly, different d for different segments). If, moreover, the segments are sorted (from left to right) in decreasing length, then the partition is called *sorted binary partition*.

We use the following notation for the quantities associated with a binary partition B. Let $i \in \mathbb{N}$. The total number of segments that B consists of is denoted by $n(B)$, the number of i-segments of B is denoted by $n_i(B)$, and the sum of the lengths of all i-segments of B is denoted by $\ell_i(B)$. If B does not contain any i-segments then $n_i(B) = 0$ and $\ell_i(B) = 0$. For convenience, we define $\ell_{\geq i}(B) = \sum_{k \geq i} \ell_k(B)$; $\ell_{\leq i}(B)$ is defined similarly. The smallest and largest i for which $\ell_i(B) > 0$ are called the *min depth* and *max depth* of B, and are denoted by $\nu(B)$ and $\xi(B)$, respectively. The difference $\xi(B) - \nu(B)$, denoted $bf(B)$, is called the *balance factor* of B. In all the above notation, we often omit parameter B when it is clear which binary partition we are considering.

3.2 S&M-Processes

We define two families of discrete stochastic processes that describe models for adding and removing segments in binary partitions. The first family, called *actual* S&M*-processes*, simply formulates as a stochastic process the S&M Protocol described in Section 2.2. The second family, called *virtual* S&M*-processes*, describes a slightly different, less complicated model. We introduce virtual S&M-processes because they are easier to analyze, and their analysis yields results that apply to actual S&M-processes, as well.

The sample space of an actual or virtual S&M-process is a set of finite sequences of binary partitions or *sorted* binary partitions, respectively. Both types of processes are parameterized by an *initial partition*, four positive constants called *splitting factor/term* and *merging factor/term*, and a finite binary sequence of $+/-$'s, called *event list*. Actual S&M-processes have an additional parameter, called *list of points*, which is a sequence of points from I of the same length as the event list. We explain the use of these parameters below.

Let $\mathcal{B} = \langle \mathscr{B}_i \rangle_{i=0}^{k}$, $k \geq 0$, be an actual or virtual S&M-process with initial partition B_0, splitting factor α^+, splitting term β^+, merging factor α^-, merging term β^-, and event list $\langle r_i \rangle_{i=1}^{k}$.[4] Moreover, if $\mathcal{B}$ is an actual S&M-process, let $\langle p_i \rangle_{i=1}^{k}$ be its list of points. Then, $\mathscr{B}_0 = B_0$, and for every $0 < i \leq k$, $\mathscr{B}_i$ is determined as follows. If $r_i = +$ then $\mathscr{B}_i$ results from $\mathscr{B}_{i-1}$ by splitting in half a segment of $\mathscr{B}_{i-1}$ that is selected using a probabilistic protocol described below. If $r_i = -$ then $\mathscr{B}_i$ is generated by merging two sibling segments of $\mathscr{B}_{i-1}$ into a single segment. Again, the pair of segments is chosen probabilistically.

If $\mathcal{B}$ is an actual S&M-process, the following *segment selection protocol* is used. Let d be the depth of the segment containing point p_i. If $r_i = +$, we pick $\lceil \alpha^+ d + \beta^+ \rceil$ points of I *at random*, and choose to split the longest among the segments of $\mathscr{B}_{i-1}$ that contain at least one of the points selected. (If more than one such longest segments exists then we deterministically choose one of them.) If $r_i = -$, we pick $\lceil \alpha^-(d+1) + \beta^- \rceil$ points *at random*. Let $\mathcal{L}$ be the set of segments of $\mathscr{B}_{i-1}$ containing at least one of the points selected. Let also λ denote the segment containing p_i, if this segment is shorter or equal to every segment in $\mathcal{L}$, or a (deterministically selected) shortest segment in $\mathcal{L}$, otherwise. Then, if $\overline{\lambda}$ is not currently split, we choose to merge λ with

[4] A note on the use of different fonts: X and x denote fixed values, $\mathscr{X}$ denotes a random variable, and $\mathcal{X}$ denotes a stochastic process or set of random variables.

$\overline{\lambda}$; otherwise, we (deterministically) choose a pair of sibling segments that are subsegments of $\overline{\lambda}$.

If $\mathcal{B}$ is a virtual S&M-process, the segment selection protocol is as follows. If $r_i = +$, we pick $\lceil \alpha^+ \nu(\mathscr{B}_{i-1}) + \beta^+ \rceil$ points of I *at random*. Let d be the depth of the longest segment of $\mathscr{B}_{i-1}$ that contains at least one of the points selected. Then, we choose to split the rightmost d-segment of $\mathscr{B}_{i-1}$. If $r_i = -$, we pick $\lceil \alpha^-(\xi(\mathscr{B}_{i-1}) - 1) + \beta^- \rceil$ points of I *at random*. Let d be the depth of the shortest segment of $\mathscr{B}_{i-1}$ that contains at least one of the points selected. Then, if $n_d(\mathscr{B}_{i-1}) \geq 2$ we choose to merge the two leftmost d-segments of $\mathscr{B}_{i-1}$. Otherwise, we choose the two leftmost d'-segments of $\mathscr{B}_{i-1}$, where d' is the minimum $j > d$ for which $n_j(\mathscr{B}_{i-1}) \geq 2$. (Note that, if $n(\mathscr{B}_{i-1}) \geq 2$ there is always such a d'.)

We conclude this section by comparing the behavior of actual and virtual S&M-processes, with respect to load balancing. Informally, an actual S&M-process provides at least as strong load balancing guarantees as a virtual S&M-process with similar parameters. More precisely, let two partitions B, B' be called *similar* if $n_i(B) = n_i(B')$ for all $i \in \mathbb{N}$. Let also $bf(\mathcal{B})$ denote the *maximum* balance factor of any partition in (actual or virtual) S&M-process $\mathcal{B}$. Then, it is easy to show (using coupling) that the following statement holds.

LEMMA 3.1. *If actual* S&M*-process $\mathcal{B}_1$ and virtual* S&M*-process $\mathcal{B}_2$ have the same splitting and merging factors/terms, the same event list, and similar initial partitions, then $Pr[bf(\mathcal{B}_1) \leq 2] \geq Pr[bf(\mathcal{B}_2) \leq 2]$.*

This lemma allows us to directly apply to actual S&M-processes the bounds we will prove for virtual S&M-processes in the next section.

3.3 Analysis of Virtual S&M-Process

Informally, we will show that in a virtual S&M-process starting from a "sufficiently balanced" initial partition with large enough constant merging and splitting factors, with high probability, the balance factor of all binary sorted partitions in the process is at most 2, provided that the number of segments in the system does not become too small, and the process does not go on for too long (i.e., the number of steps is polynomial in the minimum number of segments during the process). Throughout the remainder of this section, whenever we say "partition", we mean "binary sorted partition".

We consider a partition to be sufficiently balanced if one of the following two conditions applies: (i) $bf \leq 1$, or (ii) $bf = 2$ and $\ell_\nu \leq u$ and $\ell_\xi \leq v$, where u, v are appropriate constants with $0 \leq u, v \leq 1$. A partition that satisfies either of the above conditions is called $\langle u, v \rangle$*-safe*. Note that condition (i) implies that $bf = 0$ if n is a power of 2, and that $bf = 1$ otherwise. So, it describes partitions that are (in some sense) completely balanced. Condition (ii), on the other hand, refers to partitions that deviate from a completely balanced state, by (at most) as much as described by parameters u, v. As expected, we will see that these parameters are closely related to the selected factor/term parameters of the S&M-process, and the probability with which the load balancing guarantees of the process hold.

Let $|\mathcal{B}|$ denote the length of S&M-process $\mathcal{B}$, i.e., the number of partitions in $\mathcal{B}$ minus one. Recall also that $bf(\mathcal{B})$ denotes the maximum balance factor of any partition in $\mathcal{B}$. Then, the main result of this section can be formally stated as follows.

THEOREM 3.1. *Let $b > 0$. There are $\alpha^+, \beta^+, \alpha^-, \beta^- > 0$ and $0 \leq u, v \leq 1$, all of which depend only on b, such that if virtual* S&M*-process $\mathcal{B}$ starts from a $\langle u, v \rangle$-safe initial partition and has splitting factor $\geq \alpha^+$, splitting term $\geq \beta^+$, merging factor $\geq \alpha^-$ and merging term $\geq \beta^-$, then $Pr[bf(\mathcal{B}) \leq 2] = 1 - \mathrm{O}(|\mathcal{B}|/N^b)$, where N is the minimum number of segments in any partition of $\mathcal{B}$.*

The proof of this theorem is presented in Section 3.3.2. Before that, in Section 3.3.1, we show some basic facts we will use in the proof.

3.3.1 Basic Results

We begin by introducing some definitions. The random experiment that determines the next partition in a virtual S&M-process from the previous one is called a *v-split* (respectively a *v-merge*) if it splits a segment (respectively merges a pair of segments) of the partition it is applied to. A v-split is *favorable* if it splits a longest segment (of the partition it is applied to), and *non-favorable* otherwise. Similarly, a v-merge is *favorable* if it merges two shortest segments, and *non-favorable* otherwise.

The first two lemmata we present in this section bound respectively the probability that a single v-split splits a segment of length above a given minimum, and the probability a single v-merge merges segments of length below some maximum. The proof of the second lemma is similar to that of the first one, and it is omitted.

LEMMA 3.2. *Let $\alpha, c > 0$. For any partition with $\ell_{\leq i} \geq c/\alpha$, for some $i \geq 0$, a v-split with splitting factor $\geq \alpha$ splits a segment of depth $\leq i$, with probability $\geq 1 - e^{-c\nu}$.*

PROOF. Let $w = \ell_{\leq i}$. The probability the v-split splits a segment of depth $\leq i$ is $\geq 1 - (1-w)^{\alpha\nu} \geq 1 - e^{-w\alpha\nu} \geq 1 - e^{-c\nu}$, since $w \geq c/\alpha$. □

LEMMA 3.3. *Let $\alpha, c > 0$. For any partition with $\ell_{\geq i} \geq c/\alpha$, for some $i \geq 0$, a v-merge with merging factor $\geq \alpha$ merges segments of depth $\geq i$, with probability $\geq 1 - e^{-c(\xi-1)}$.*

The first of the remaining two lemmata bounds the number of consecutive v-splits that should be applied to a given partition until all its longest segments are split, with certain probability. The proof is quite technical and is described in the Appendix. The last lemma bounds the number v-merges required until all the shortest segments in a partition are merged, with a given probability. Its proof is similar to that of Lemma 3.4, and it is omitted.

LEMMA 3.4. *Let $0 < w \leq 1$, $\alpha > 1$, and $c > 0$. For any partition B with $\ell_\nu \leq w$, a sequence of successive v-splits with splitting factor $\geq \alpha$ and sufficiently large splitting term $\beta(w, \alpha, c)$, splits all ν-segments of B after $\leq 2^\nu(\frac{w}{\alpha-1} + \frac{4c+6}{\alpha})$ non-favorable v-splits occur, with probability $\geq 1 - e^{-c\nu}$.*

LEMMA 3.5. *Let $0 < w \leq 1$, $\alpha > 1$, and $c > 0$. For any partition B with $\ell_\xi \leq w$, a sequence of successive v-merges with merging factor $\geq \alpha$ and sufficiently large merging term $\beta(w, \alpha, c)$, merges all ξ-segments of B after $\leq 2^{\xi-1}(\frac{w}{\alpha-1} + \frac{4c+6}{\alpha})$ non-favorable v-merges occur, with probability $\geq 1 - e^{-c(\xi-1)}$.*

3.3.2 Proof of Theorem 3.1

For every partition B and $0 \leq w \leq 1$ let $low(B,w)$ be the minimum j such that $\ell_{\leq j}(B) \geq w$, and $high(B,w)$ be the maximum j such that $\ell_{\geq j}(B) \geq w$. Consider, now, the following variation of $\mathcal{B}$, denoted $\hat{\mathcal{B}}$. $\hat{\mathcal{B}}$ is defined identically to $\hat{\mathcal{B}}$, except for a slight modification in the segment selection protocol. Namely, in each v-split of $\hat{\mathcal{B}}$, applied to some partition B, the segment selection protocol (as defined for $\mathcal{B}$) is repeatedly executed until it selects a segment of depth $\leq low(B,u)$. Similarly, in every v-merge the segment selection protocol is repeated until it selects a segment of depth $\geq high(B,v)$.

The modified v-splits and v-merges of $\hat{\mathcal{B}}$ are called respectively *r-splits* and *r-merges*, and the number of times the segment selection protocol of an r-split/r-merge is executed is called the *multiplicity* of this r-split/r-merge. Finally, let $mult(\hat{\mathcal{B}})$ denote the maximum multiplicity of all r-splits and r-merges of $\hat{\mathcal{B}}$.[5]

Informally, to prove the theorem we first bound the probability that $bf(\hat{\mathcal{B}})$ is at most 2. Then, we bound the probability of the event that all steps of $\hat{\mathcal{B}}$ have multiplicity 1. (Note that this event implies that $\hat{\mathcal{B}}$ and $\mathcal{B}$ are indistinguishable if they make the same random choices.) Finally, we combine the above two results to bound the probability that $bf(\mathcal{B})$ is at most 2.

More precisely, let $c = b\ln 2$, $\alpha^+ = \frac{7}{2}c + 3 (\approx 2.5b + 3)$, $\alpha^- = 4\alpha^+$, $u = c/\alpha^+$, and $v = c/\alpha^-$. Let also $\beta^+ = \beta^+(u, \alpha^+, c)$ be large enough so that Lemma 3.4 holds for $w = u$, $\alpha = \alpha^+$ and $\beta = \beta^+$, and let $\beta^- = \beta^-(v, \alpha^-, c)$ be large enough so that Lemma 3.5 holds for $w = v$, $\alpha = \alpha^-$ and $\beta = \beta^-$. Then, the following claims apply for the above parameters.

CLAIM 3.1. $Pr[bf(\hat{\mathcal{B}}) \leq 2] \geq 1 - 4^b|\mathcal{B}|/N^b$.

CLAIM 3.2. $Pr[mult(\hat{\mathcal{B}}) = 1 \mid bf(\hat{\mathcal{B}}) \leq 2] \geq 1 - 4^b|\mathcal{B}|/N^b$.

It is also easy to see (and prove formally using coupling) that $\Pr[bf(\mathcal{B}) \leq 2] \geq Pr[\{mult(\hat{\mathcal{B}}) = 1\} \cap \{bf(\hat{\mathcal{B}}) \leq 2\}]$. Thus, $\Pr[bf(\mathcal{B}) \leq 2] \geq Pr[mult(\hat{\mathcal{B}}) = 1 \mid bf(\hat{\mathcal{B}}) \leq 2] \cdot Pr[bf(\hat{\mathcal{B}}) \leq 2]$, and using the claims above we obtain $\Pr[bf(\mathcal{B}) \geq 2] \geq 1 - 8^b|\mathcal{B}|/N^b = 1 - \mathrm{O}(|\mathcal{B}|/N^b)$.

To complete the proof of the theorem it remains to prove the two claims. We start with the second one, whose proof is shorter.

PROOF OF CLAIM 3.2. It is easy to verify that the probability an r-split of $\hat{\mathcal{B}}$ has multiplicity 1, given the partition that it is applied to, is independent of all the other partitions in $\hat{\mathcal{B}}$ (even the one that results from the execution of the r-split). Now, consider an r-split of $\hat{\mathcal{B}}$ that is applied to some partition B with $bf(B) \leq 2$. The probability it has multiplicity 1 is equal to the probability p a v-step (with the same splitting factor/term) applied to B splits a segment of depth $\leq low(B,u)$. Using Lemma 3.2 with $\alpha = \alpha^+$ and $i = low(B,u)$ we get that $p \geq 1 - 1/2^{b\nu(B)}$. Since $bf(B) \leq 2$, $2^\nu \geq N/4$, so, $p \geq 1 - (4/N)^b$. Using a similar reasoning (and Lemma 3.3 instead of 3.2) we can show that an r-merge has multiplicity 1 with probability $\geq 1-(2/N)^b$, for any possible sequence of partitions in $\hat{\mathcal{B}}$, provided it is applied to a partition with balance factor ≤ 2. The claim follows by combining the above two results. □

PROOF OF CLAIM 3.1. Let $\hat{\mathcal{B}} = \langle \mathscr{B}_i \rangle_{i=0}^{\kappa}$, and for every $0 \leq j \leq k \leq \kappa$, let $\hat{\mathcal{B}}_j^k = \langle \mathscr{B}_i \rangle_{i=j}^{k}$, π_j^k be the event "all partitions in $\hat{\mathcal{B}}_j^k$ have balance factor ≤ 2", and σ_j^k be the event "$\mathscr{B}_k$ is the *only* $\langle u,v \rangle$-safe partition in $\hat{\mathcal{B}}_j^k$". We also define events $\pi_{\kappa+1}^\kappa$ and $\sigma_{\kappa+1}^\kappa$ to be always "true". We will show that for every $0 \leq j \leq \kappa$, the following predicate holds. $S(j)$: for any $\langle u,v \rangle$-safe partition B, $Pr[\pi_{j+1}^\kappa \mid \mathscr{B}_j = B] \geq 1 - 4^b(\kappa - j)/N^b$. The claim, then, follows directly by setting $j = 0$.

The proof is by induction on j. Obviously, $S(\kappa)$ holds. Assume $S(k)$ holds for all $j < k \leq \kappa$. We will show that $S(j)$ holds, as well. We have $Pr[\pi_{j+1}^\kappa \mid \mathscr{B}_j = B] = \sum_{k=j+1}^{\kappa} Pr[\pi_{j+1}^k \cap \sigma_{j+1}^k \cap \pi_{k+1}^\kappa \mid \mathscr{B}_j = B] = \sum_{k=j+1}^{\kappa} Pr[\pi_{k+1}^\kappa \mid \{\mathscr{B}_j = B\} \cap \pi_{j+1}^k \cap \sigma_{j+1}^k] \cdot Pr[\pi_{j+1}^k \cap \sigma_{j+1}^k \mid \mathscr{B}_j = B]$. Applying the induction hypothesis we obtain $Pr[\pi_{k+1}^\kappa \mid \{\mathscr{B}_j = B\} \cap \pi_{j+1}^k \cap \sigma_{j+1}^k] \geq 1 - 4^b(\kappa - k)/N^b$. Moreover, we will later show that $\sum_{k=j+1}^{\kappa} Pr[\pi_{j+1}^k \cap \sigma_{j+1}^k \mid \mathscr{B}_j = B] \geq 1 - 4^b/N^b$. Combining these facts we get $Pr[\pi_{j+1}^\kappa \mid \mathscr{B}_j = B] \geq 1 - 4^b(\kappa - j)/N^b$.

To complete the proof we need to show that for any $0 \leq j \leq \kappa$, and any $\langle u,v \rangle$-safe partition B, $\sum_{k=j+1}^{\kappa} Pr[\pi_{j+1}^k \cap \sigma_{j+1}^k \mid \mathscr{B}_j = B] \geq 1 - 4/N^b$. We have the following cases.

Case 1: $bf(B) \leq 1$, or $bf(B) = 2$ and $\ell_\nu(B) < \lfloor u \rfloor_\nu$ and $\ell_\xi(B) < \lfloor v \rfloor_{\xi-1}$.[6] For the selected values of u and v, it is easy to verify that a single r-split or r-merge leads to a $\langle u,v \rangle$-safe partition with certainty, i.e., $Pr[\pi_{j+1}^{j+1} \cap \sigma_{j+1}^{j+1}) \mid \mathscr{B}_j = B] = 1$.

Case 2(a): $bf(B) = 2$ and $\ell_\xi(B) = \lfloor v \rfloor_{\xi-1}$. Let k be the smallest $i > j$ such that (i) $\ell_{\nu(B)}(\mathscr{B}_i) = 0$, or (ii) $\ell_\xi(\mathscr{B}_i) = \ell_\xi(B)$, or (iii) $i = \kappa$. Assume $\hat{\mathcal{B}}_j^k$ involves only r-splits (and no r-merges). Then, applying Lemma 3.4 with $w = u$, $\alpha = \alpha^+$ and $\beta = \beta^+$, we obtain that, with probability $\geq 1 - 1/2^{b\nu} \geq 1 - (4/N)^b$, at most $\phi = 2^\nu(\frac{u}{\alpha^+ - 1} + \frac{4c+6}{\alpha^+})$ of the r-splits during $\hat{\mathcal{B}}_j^k$ are non-favorable. Thus, with probability $\geq 1 - (4/N)^b$, for all partitions in $\hat{\mathcal{B}}_j^k$, $\ell_{\geq \nu+2} \leq v + \phi/2^{\nu+1}$. By substituting the values of α^+, u, v, we obtain $\ell_{\geq \nu+2} \leq 1 - u$, which implies that none of the r-splits splits a ξ-segment. Thus, with probability $\geq 1 - (4/N)^b$, π_{j+1}^k occurs and either condition (i) is met and $bf(\mathscr{B}_k) = 1$, or condition (iii) is met (i.e., σ_k occurs, as well). Now, if $\hat{\mathcal{B}}_j^k$ involves also r-splits, then *all* r-merges before ϕ non-favorable r-splits occur are favorable. So, the previous analysis continues to apply with the slight modification that condition (ii) may be met before the other two, in which case we also have $bf(\mathscr{B}_k) = 1$.

Case 2(b): $bf(B) = 2$ and $\ell_\nu(B) = \lfloor u \rfloor_\nu$. The analysis is "symmetric" to that of the previous case, with the roles of v-splits and v-merges switched. Lemma 3.5 is used instead of 3.4. □

4. SKETCH OF ANALYSIS OF WEIGHTED S&M PROTOCOL

The analysis of the weighted version of the protocol is similar to that of the unweighted version. Again we use an

[5] The notation and terms we defined for virtual S&M-processes and v-splits/v-merges are extended to apply for $\hat{\mathcal{B}}$ and r-splits/r-merges in the obvious way.

[6] Let $\lfloor x \rfloor_i = \lfloor x \cdot 2^i \rfloor / 2^i$, i.e., the largest integral multiple of $1/2^i$ that is at most x; $\lceil x \rceil_i$ is defined similarly.

aggregated representation of the system state, and study the transitions in this simplified (virtual) state space. This time, however, the abstraction we use is slightly more involved than the one we used in the analysis of the unweighed protocol (i.e., sorted binary partitions and virtual S&M-process), since here we need to also take node weights into account.

More precisely, the information we maintain for each system state is the distribution of the group segment lengths, the distribution of group weights for groups associated with longest group segments, when the total length of these segments is below some threshold, and the distribution of group weights for groups associated with shortest group segments, when the total length of these segments is below a threshold.

As in the analysis of the unweighted protocol, we show that in states with a large number of longest group segments joins add nodes to longest group segments almost with certainty (for large enough system parameters). We also bound the number of joins required until all longest group segments are split when starting from states with a small number of longest group segments. The additional complication introduced by the fact that not all joins result in group splits is overcome by assuming "adversarial" weight selection, i.e., we assume nodes added in longest group segments have weight 1, while nodes added in other groups have weight W. Thus, a longest group segment must be selected W times before it is split, while a join in a non-longest group segment always causes a group split. Analogous statements are shown to hold for node leaves. Finally, we combine these results in a manner similar to that in the proof of Theorem 3.1.

This analysis shows that the Weighted S&M Protocol supports for the group segments the same load balancing guarantees the unweighted protocol provides about node segments.

5. REFERENCES

[1] I. Abraham, B. Awerbuch, Y. Azar, Y. Bartal, D. Malkhi, and E. Pavlov. A generic scheme for building overlay networks in adversarial scenarios. In *Proc. International Parallel and Distributed Processing Symposium (IPDPS 2003)*, page 40.2, Apr. 2003.

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

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

[4] P. Fraigniaud and P. Gauron. The content-addressable network D2B. Technical Report 1349, RI, University of Paris-Sud, France, Jan. 2003.

[5] K. Hildrum, J. Kubiatowicz, S. Rao, and B. Zhao. Distributed object location in a dynamic network. In *Proc. 14th ACM Symposium on Parallel Algorithms and Architectures (SPAA 2002)*, pages 41–52, Aug. 2002.

[6] F. Kaashoek and D. Karger. Koorde: A simple degree-optimal hash table. In *Proc. 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)*, pages 98–107, Feb. 2003.

[7] 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 *Proc. 29th Annual ACM Symposium on Theory of Computing (STOC 1997)*, pages 654–663, May 1997.

[8] D. Karger and M. Ruhl. Simple efficient load balancing algorithms for peer-to-peer systems. In *Proceed. 16th ACM Symposium on Parallel Algorithms and Architectures (SPAA 2004)*, pages 36–43, 2004.

[9] K. Kenthapadi and G. Manku. Decentralized algorithms using both local and random probes for P2P load balancing. In *Proc. 17th ACM Symposium on Parallel Algorithms and Architectures (SPAA '05)*, July 2005. (to appear).

[10] D. Malkhi, M. Naor, and D. Ratajczak. Viceroy: a scalable and dynamic emulation of the butterfly. In *Proc. 21st Annual Symposium on Principles of Distributed Computing (PODC 2002)*, pages 183–192, July 2002.

[11] G. Manku. Routing networks for DHTs. In *Proc. 22nd Annual Symposium on Principles of Distributed Computing (PODC 2003)*, pages 133–142, July 2003.

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

[13] G. Manku, M. Bawa, and P. Raghavan. Symphony: Distributed hashing in a small world. In *Proc. 4th USENIX Symposium on Internet Technologies and Systems (USITS '03)*, pages 127–140, Mar. 2003.

[14] P. Maymounkov and D. Mazières. Kademlia: A peer-to-peer information system based on the XOR metric. In *Proc. 1st International Workshop on Peer-to-Peer Systems (IPTPS '02)*, pages 53–65, Mar. 2002.

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

[16] M. Naor and U. Wieder. Novel architectures for P2P applications: the continuous-discrete approach. In *Proc. 15th ACM Symposium on Parallel Algorithms and Architectures (SPAA '03)*, pages 50–59, June 2003.

[17] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proc. ACM SIGCOMM 2001 Conference (SIGCOMM 2001)*, pages 161–172, Aug. 2001.

[18] S. Ratnasamy, I. Stoica, and S. Shenker. Routing algorithms for DHTs: Some open questions. In *Proc. 1st International Workshop on Peer-to-Peer Systems (IPTPS '02)*, pages 45–52, Mar. 2002.

[19] A. Richa, M. Mitzenmacher, and S. Sitaraman. The power of two random choices: A survey of techniques and results, Sept. 2000.

[20] A. Rowstron and P. Druschel. Pastry: scalable, decentraized object location and routing for large-scale peer-to-peer systems. In *Proc. 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001)*, pages 329–350, Nov. 2001.

[21] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer

lookup service for internet applications. In *Proc. ACM SIGCOMM 2001 Conference (SIGCOMM 2001)*, pages 149–160, Aug. 2001.

APPENDIX

A. PROOF OF LEMMA 3.4

First, we prove three lemmata that bound the number of v-splits required until the number of longest segments of a given initial partition drops below some threshold, with certain probability. The main lemma, then, follows by combining the three results.

The first lemma computes a bound on the number of v-splits required until $\ell_\nu \leq 1/\nu$. To prove this statement, we observe that the probability that any of these v-splits is favorable is greater or equal to the probability a v-split applied to the last partition reached is favorable; then, we use Chernoff's inequality to obtain the desired bound assuming each v-split is favorable with that minimum probability.

LEMMA A.1. *Let $0 < w \leq 1$, $\alpha > 1$, and $b > 0$. For any partition B with $\ell_\nu(B) \leq w$, a sequence of successive v-splits with splitting factor $\geq \alpha$ and sufficiently large splitting term $\beta(w, \alpha, b)$ reaches a partition B' with $\ell_\nu(B') \leq 1/\nu$ after $\leq 2^\nu \frac{w}{\alpha-1}$ non-favorable v-splits occur, with probability $\geq 1 - e^{-b\nu}$.*

PROOF. Assume $\ell_\nu(B) > 1/\nu$ (otherwise, the lemma holds trivially). If $2^\nu \frac{w}{\alpha-1} < 1$ the lemma holds for $\beta \geq \frac{9}{8}(b+1)\frac{\alpha-1}{w} > \frac{9}{8}(b+1)2^\nu \geq (b+1)\nu^2$, because then all v-splits until the first partition with $\ell_\nu \leq 1/\nu$ is reached are favorable with probability $\geq 1 - 2^\nu(1-1/\nu)^\beta \geq 1 - 2^\nu e^{-(b+1)\nu} \geq 1 - e^{-b\nu}$. For the rest of the proof we assume that $2^\nu \frac{w}{\alpha-1} \geq 1$. Let $m = n_\nu(B) - \lfloor 2^\nu/\nu \rfloor$, i.e., m is the number of *favorable* v-splits required to reach B'. Let also $\mathscr{B}_0 = B$, and $\mathscr{B}_i$, $i \geq 1$, be the partition that results by applying to $\mathscr{B}_{i-1}$ a v-split with splitting factor $\geq \alpha$, and term $\geq \beta$ (to be determined later). Finally, let $\mathscr{X}_i$, $i \geq 1$, be the indicator random variable of event $\{\ell_\nu(\mathscr{B}_i) = \ell_\nu(\mathscr{B}_{i-1})\} \cap \{\ell_\nu(\mathscr{B}_{i-1}) > 1/\nu\}$. Then, for any integer $k > 0$, the probability it takes $\leq k$ non-favorable v-splits (or equivalently, $\leq m + k$ v-splits) until the first partition with $\ell_\nu \leq 1/\nu$ is reached is equal to $Pr[\sum_{i=1}^{m+k} \mathscr{X}_i \leq k] = 1 - Pr[\sum_{i=1}^{m+k} \mathscr{X}_i > k]$. We will bound $Pr[\sum_{i=1}^{m+k} \mathscr{X}_i > k]$. Note that $Pr[\mathscr{X}_i = 1 \mid \mathscr{X}_1, \ldots, \mathscr{X}_{i-1}] \leq Pr[\ell_\nu(\mathscr{B}_i) = \ell_\nu(\mathscr{B}_{i-1}) \mid \ell_\nu(\mathscr{B}_{i-1}) = \lceil 1/\nu \rceil_\nu] \leq (1-1/\nu)^{\alpha\nu+\beta} \leq e^{-\alpha-\beta/\nu} = q$.[7] Thus, it follows (by use of coupling) that $\sum_{i=1}^{m+k} \mathscr{X}_i$ is stochastically smaller than the sum $\sum_{i=1}^{m+k} \mathscr{Y}_i$, where $\mathscr{Y}_1, \ldots, \mathscr{Y}_{m+k}$ are independent Bernoulli trials with probability of success q. So, $Pr[\sum_{i=1}^{m+k} \mathscr{X}_i > k] \leq Pr[\sum_{i=1}^{m+k} \mathscr{Y}_i > k] = Pr[\sum_{i=1}^{m+k} \mathscr{Y}_i > (1+\delta)\mathrm{E}\{\sum_{i=1}^{m+k} \mathscr{Y}_i\}]$, where $\mathrm{E}\{\sum_{i=1}^{m+k} \mathscr{Y}_i\} = q(m+k)$ and $\delta + 1 = \frac{k}{q(m+k)}$. If $k > \frac{qm}{1-q}$ then $\delta > 0$, and by using Chernoff's bound [15] we obtain $Pr[\sum_{i=1}^{m+k} \mathscr{X}_i > k] \leq [e^\delta/(1+\delta)^{1+\delta}]^{q(m+k)} \leq [e/(1+\delta)]^{(\delta+1)q(m+k)} = [e/(1+\delta)]^k$. So, $Pr[\sum_{i=1}^{m+k} \mathscr{X}_i > k] \leq [eq(1+m/k)]^k \leq e^{(1-\alpha-\beta/\nu+m/k)k} = e^{[m-(\alpha-1)k]-\frac{\beta k}{\nu}}$. Let $k = \lfloor 2^\nu \frac{w}{\alpha-1} \rfloor$ (which is > 0 from the assumption at the beginning of the proof). Then, $k \geq \lfloor \frac{m}{a-1} \rfloor \geq \frac{m-(a-1)}{a-1}$, so, $m - (\alpha-1)k \leq a-1$. If also $\beta \geq \frac{b\nu^2+(a-1)\nu}{k}$, then $Pr[\sum_{i=1}^{m+k} \mathscr{X}_i > \lfloor 2^\nu \frac{w}{\alpha-1} \rfloor] \leq e^{(a-1)-[b\nu+(a-1)]} = e^{-b\nu}$.

[7] See footnote 6.

Thus, for the lemma to hold it suffices to have $\beta \geq [b\frac{\nu^2}{2^\nu} + (a-1)\frac{\nu}{2^\nu}]/\lfloor \frac{w}{a-1} \rfloor_\nu$, or $\beta \geq (\frac{9}{8}b + a - 1)\frac{2(a-1)}{w}$. □

The next lemma considers starting partitions with $\ell_\nu \leq 1/\nu$, and counts the number of v-splits (with splitting factor a) required until $\ell_\nu \leq \frac{1}{\alpha\nu}$. The proof is similar to that of Lemma A.1, and it is omitted.

LEMMA A.2. *Let $\alpha > 1$, and $\beta, b > 0$. For any partition B with $\ell_\nu(B) \leq 1/\nu$, a sequence of successive v-splits with splitting factor $\geq \alpha$ and term $\geq \beta$ reaches a partition B' with $\ell_\nu(B') \leq \frac{1}{\alpha\nu}$ after $\leq 2^\nu \frac{2\alpha}{\beta}(b+1)$ non-favorable v-splits occur, with probability $\geq 1 - e^{-b\nu}$.*

The third lemma applies for starting partitions with $\ell_\nu \leq \frac{1}{\alpha\nu}$ and bounds the number of v-splits until *all* ν-segments are split. In this case we cannot use the same argument as in the proof of the previous lemmata, since the probability of a favorable split when ℓ_ν is close to 0 is very small (i.e., of order $\frac{\nu}{2^\nu}$). Instead, we use an adaptation of the Coupon Collector problem.

LEMMA A.3. *Let $\alpha, b > 0$. For any partition B with $\ell_\nu(B) \leq \frac{1}{\alpha\nu}$, a sequence of successive v-splits with splitting factor $\geq \alpha$ and sufficiently large splitting term $\beta(\alpha, b)$ splits all ν-segments of B after $\leq 2^\nu \frac{4(b+1)}{\alpha}$ non-favorable v-splits occur, with probability $\geq 1 - e^{-b\nu}$.*

PROOF SKETCH. Consider the slightly modified setting where in each v-split we choose to split the longest of the probed segments (instead of the rightmost segment whose length is equal to that of the longest segment probed). If more than one longest segments are probed, then one of them is chosen *at random*. This setting is (stochastically) equivalent to the original in terms of the number of v-splits required until all ν-segments are split. Consider now an additional modification to the segment selection protocol of the original process. If at least one point $\leq \frac{1}{\alpha\nu}$ is selected, then we split the segment containing a (new) randomly selected point in the range $[0, \frac{1}{\alpha\nu})$. (Otherwise the standard selection protocol is applied). The number $\mathscr{T}$ of v-splits required in the last setting until all ν-segments are split is stochastically larger than in the first one and, thus, stochastically larger than in the original. We bound $\mathscr{T}$ as follows. Fix a ν-segment of B. The probability the segment is split during a v-split is $\geq \frac{\alpha\nu}{2^\nu} \cdot \left(1 - (1 - \frac{1}{\alpha\nu})^{\alpha\nu}\right) \geq \frac{\alpha\nu}{2^\nu} \cdot (1 - \frac{1}{e})$. Using Chernoff's bound we can show that the segment is still not split after $2^\nu \frac{4(b+1)}{\alpha}$ v-splits with probability $\leq e^{-(b+1)\nu}$. Thus, the probability at least one ν-segments is not split within the above number of v-splits is $Pr[\mathscr{T} \geq 2^\nu \frac{4(b+1)}{\alpha}] \leq e^{-b\nu}$. □

We are now ready to prove our main lemma.

PROOF OF LEMMA 3.4. Consider a sequence of v-splits with splitting factor $\geq \alpha$ and term $\geq \beta$ (to be determined later) that starts from partition B. Let $r_1 2^\nu$ be the number of non-favorable v-splits until a partition B_1 with $\ell_\nu(B_1) \leq 1/\nu$ is reached. Then, using Lemma A.1 with $b = c + 3$, we obtain that for $\beta \geq$ some sufficiently large $\beta_1(w, \alpha, c+3)$, $Pr[r_1 \leq \frac{w}{\alpha-1}] \geq 1 - e^{-3}e^{-c\nu}$. Assuming B_1 is reached, let $r_2 2^\nu$ be the number of non-favorable v-splits until a partition B_2 with $\ell_\nu(B_2) \leq \frac{1}{\alpha\nu}$ is reached. Then, using Lemma A.2 with $b = c + 3$ and $\beta \geq 2\alpha^2(c+4) = \beta_2$ we obtain that $Pr[r_2 \leq 1/\alpha] \geq 1 - e^{-3}e^{-c\nu}$. Finally, if

$r_3 2^\nu$ is the remaining number of non-favorable v-splits until all ν-segments are split, then using Lemma A.3 with $b = c + 1/4$ we obtain that for $\beta \geq$ some sufficiently large $\beta_3(\alpha, c + 1/4)$, $Pr[r_3 \leq \frac{4c+5}{\alpha}] \geq 1 - e^{-1/4} e^{-c\nu}$. Combining the above probabilities, we get that for $\beta = \max\{\beta_1, \beta_2, \beta_3\}$, $Pr[r_1 + r_2 + r_3 \leq \frac{w}{\alpha - 1} + \frac{4c+6}{\alpha}] \geq 1 - e^{-c\nu}$. □