Routing Networks for Distributed Hash Tables

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

ABSTRACT

Routing topologies for distributed hashing in peer-to-peer networks are classified into two categories: deterministic and randomized. A general technique for constructing deterministic routing topologies is presented. Using this technique, classical parallel interconnection networks can be adapted to handle the dynamic nature of participants in peer-to-peer networks. A unified picture of randomized routing topologies is also presented. Two new protocols are described which improve average latency as a function of out-degree. One of the protocols can be shown to be optimal with high probability. Finally, routing networks for distributed hashing are revisited from a systems perspective and several open design problems are listed.

1. INTRODUCTION

Distributed Hash Tables (DHTs) are currently under investigation in the context of peer-to-peer (P2P) systems. The hash table is partitioned with one participant managing any given partition. This engenders maintenance of a table that maps a partition to its manager's network address. A simple scheme is to let a central server maintain the mapping. However, participants in P2P systems are numerous and span wide-area networks. Their short lifetimes result in frequent arrivals and departures. A central server could ameliorate its load by leasing portions of the mapping table to clients for caching. Still, central servers are single points of failure and potential performance bottlenecks. DHTs obviate the need for central servers altogether by creating an overlay network among the participants. Hash lookups are routed to appropriate managers using the overlay. It is desirable that the number of hops for lookups be small. However, nodes should not be encumbered with large numbers of overlay connections. Thus DHT routing topologies face two conflicting goals: fast lookups but small state. Table 1 summarizes the trade-offs offered by various DHT topologies. All the protocols are scalable and handle dynamic networks. The costs of joining and leaving are also reasonable.

Summary of Paper

a) We classify DHT routing networks into two categories: deterministic and randomized. Overlay connections in a deterministic topology are a function of the current set of node ids. In the case of randomized topologies, there is conceptually a large set of possible networks for a given set of node ids. At run-time, a specific network is chosen depending upon the random choices made by all participants.

b) Existing deterministic DHT routing networks are adaptations of specific parallel inter-connection networks: hypercubes [13, 26, 28], tori [26] and de Bruijn graphs [10, 14, 24]. We present a general technique for building deterministic DHTs that allows us to adapt any of the innumerable parallel routing topologies to handle the dynamic nature of P2P networks. Our construction sheds light on the structure of the solution space, enabling a common proof technique for analyzing deterministic topologies. In the process, we obtain several new DHT routing networks with $k = O(1)$ links and $O(\ln n / \ln k)$ average latency.

c) We identify the common machinery underlying randomized topologies. We describe two new constructions in this space. A simple scheme provides $O(\ln n)$ average latency with only $O(\ln \ln n)$ links per node. A rather sophisticated scheme requires only $3\ell + 3$ links per node for average latency $O(\ln n / \ln \ell)$, which is optimal. Both latency bounds hold with high probability.

d) Using the algorithmic insights obtained, we revisit the problem of building DHTs and identify sub-problems that merit attention as separate black-boxes from a systems perspective. We list several open design problems in the end.

Road map

In Section 2, we summarize previous work. In Sections 3 and 4, we study deterministic and randomized DHTs respectively. In Section 5, we present an optimal randomized protocol. In Section 6, we list research issues that merit further investigation.

2. PREVIOUS WORK

Inspired by the popularity of file-sharing applications like Napster, Gnutella and Kazaa, the research community is exploring the possibility of harnessing computing resources distributed across the globe into a coherent infrastructure for distributed applications. The efficacy of DHTs as a low-level abstraction is currently under scrutiny.

The problem of constructing DHTs is both old and new. Distributed hashing has been studied extensively by the SDDS (Scalable Distributed Data Structures) community,

starting with the seminal work of Litwin, Niemat and Shneider [18]. However, these hash tables have central components and are designed for small-sized clusters. High-performance hash tables over large clusters were recently studied by Gribble et al [12]. Hash tables over peer-to-peer networks present novel challenges. Peer-to-peer networks consist of millions of machines over the wide-area network. Moreover, the set of participants is dynamic with frequent arrivals and departures of nodes with short lifetimes.

CAN [26], Chord [29], Pastry [28] and Tapestry [13] were among the first deterministic DHT proposals. CAN is an adaptation of multidimensional tori while Chord has similarities with hypercubes. Pastry [28] and Tapestry [13] are quite similar to each other and build upon earlier work by Plaxton et al [25]. All schemes provide $O(\ln n)$ latency with $O(\ln n)$ links per node. Recently, three groups [10, 14, 24] independently demonstrated that de Bruijn networks [23] could be adapted for routing in DHTs. Such networks provide $O(\ln n)$ latency with only $O(1)$ links per node. A natural question arises: Is it possible to morph *any* parallel interconnection network into a DHT routing protocol? One of our constructions shows that the answer is yes.

Viceroy [19] was the first randomized protocol for DHT routing. It provides $O(\ln n)$ latency with $O(1)$ links per node. Symphony [20] builds upon previous work by Kleinberg [15] to obtain a protocol that offers $O((\ln^2 n)/k)$ average latency with $k+1$ links per node for small k. Symphony and Chord are the best in terms of simplicity and symmetry.

Parallel interconnection networks [8, 17] have been extensively investigated, resulting in a rich collection of topologies over static sets of nodes. Randomized routing in this context was pioneered by Valiant [16]. Random graphs [5] have also been thoroughly investigated since 1950's. Traditionally, random graphs have been studied for mathematical properties like diameter, connectivity and chromatic number. Routing algorithms for random graphs have been developed only recently [15]. Randomized topologies appear to have been ignored by the parallel architecture community because interconnection networks are fixed in hardware.

Routing schemes for both parallel interconnection networks and random graphs assume that the set of participating nodes is static. The main challenge in adapting these schemes to peer-to-peer networks lies in handling the dynamic nature of participants who leave and join frequently.

3. DETERMINISTIC TOPOLOGIES

Without loss of generality, DHTs can be seen as mapping keys to the unit interval $[0, 1)$. The hash space is partitioned by allowing nodes to choose their ids from the interval uniformly at random. It is convenient to imagine $[0, 1)$ as a circle with unit perimeter. Node ids correspond to points on the circumference. A node maintains connections with its immediate clockwise and anti-clockwise neighbors. A node also establishes links with other nodes far away along the circle. The set of neighbors of a node depends on the parallel routing topology being mimicked.

Parallel interconnection networks consist of families of graphs with members of varying size. On the basis of structural similarities, families can be classified into two broad categories [17]. Shuffle-exchange and de Bruijn constitute one category whereas Butterflies, Cube-Connected-Cycles and Benes form the other. Many variations of these basic networks themselves exist, e.g., k-ary Butterfly, wrapped

Deterministic Routing Topologies

Protocol	# Links	Avg. Latency
CAN [26], Chord [29]	$O(\ln n)$	$O(\ln n)$
Tapstry [13], Pastry [28]	$O(\ln n)$	$O(\ln n)$
D2B [10], Koorde [14]	$k + 1$	$O(\ln n / \ln k)$
Butterfly, CCC, Benes	$\ell + 1$	$O(\ln n / \ln \ell)$

Randomized Routing Topologies

Protocol	# Links	Avg. Latency
Viceroy [19]	7	$O(\ln n)$
Kleinberg [15]	2	$O(\ln^2 n)$
Symphony [20]	$k + 1$	$O((\ln^2 n)/k)$
Bit-Collection	$\ell + 1$	$O((\ln n \ln \ln n)/\ell)$
	$O(\ln \ln n)$	$O(\ln n)$
New Algorithm	$3\ell + 3$	$O(\ln n / \ln \ell)$

Table 1: Comparison of various protocols. The current size of the network is n.

Butterfly etc. Moreover, it is possible to create products of arbitrary pairs of networks.

A family of graphs is typically defined over a static set of either 2^k nodes (hypercubes and de Bruijn graphs) or $k2^k$ nodes (butterflies). In a dynamic environment, some families are easy to maintain while others are challenging. We illustrate the problems encountered with two examples.

Chord [29] is a variant of hypercubes which constitute a family of graphs defined over 2^k nodes, $k \geq 1$. A Chord node with id $\mathbf{x} \in [0, 1)$ maintains a finger table of connections with managers of points $\langle \mathbf{x} + 1/2, \mathbf{x} + 1/4, \ldots \rangle$. As the number of participants increases from 2^k to 2^{k+1}, two changes in finger tables occur: (a) a new finger of size $1/2^{k+1}$ on average is introduced, and (b) almost all fingers are replaced. However, a new finger points to a node very close to the old finger it replaces. Let us contrast the situation with a Chord-style variant of Butterfly networks which are defined over $k2^k$ nodes, $k \geq 1$. For ease of exposition, let us assume node ids partition the interval $[0, 1)$ into equi-sized sub-intervals. One way to visualize the network is to split the interval $[0, 1)$ into 2^k groups with k node ids per group. Nodes within a group are assigned ranks $0, 1, 2, \ldots, k - 1$ in the clockwise direction. The finger table of a node with rank r consists of just one connection with a node of rank $(r + 1) \bmod k$ belonging to that group which contains the point $\mathbf{x} + 1/2^{(r+1) \bmod k}$. As the network increases in size from $k2^k$ to $(k + 1)2^{k+1}$, almost all of the existing fingers change significantly. This is because the group size changes from k to $k + 1$. With new group boundaries, the rank of a node with id $\mathbf{x}$ is quite different. This problem has actually been encountered by [7] who attempted to emulate a butterfly along the same lines. Note that the emulation of butterfly in Viceroy [19] has a different flavor. It is randomized and is discussed in Section 4.

Emulation of arbitrary families of parallel interconnection networks is challenging primarily due to two sources of uncertainty. First, the size of the network is not known accurately to all participants. Second, the distribution of points is not exactly even. In the context of butterfly networks, the first uncertainty leads to disagreement among nodes about group boundaries. A consequence of the second uncertainty is that certain groups might be empty while some groups have too many members. We address the first issue by de-

veloping an estimation protocol (Section 3.1). The second issue is addressed by clustering (Section 3.3).

3.1 Network Size Estimation

In this section, we develop a distributed scheme for estimating n, the current size of the network. Although different nodes arrive at different estimates of n, each estimate is guaranteed to lie in the range $n/(1 \pm \delta)$ with high probability[1] (w.h.p.) where $\delta \in (0,1)$ is a user parameter.

Could a node with id $\mathbf{x}$ deduce n by simply measuring the density of ids close to $\mathbf{x}$? How large a sub-interval suffices so that w.h.p., the actual number of ids in the sub-interval does not deviate significantly from that expected?

LEMMA 3.1. *Let n points be chosen independently, uniformly at random from the interval $[0,1)$. Let p_α be a random variable that equals the total number of points chosen in a fixed interval of size α. If $\alpha > (8\epsilon^{-2} \ln n)/n$, then $Pr[p_\alpha > (1+\epsilon)\mathbf{E}p_\alpha] < 1/n^2$ and $Pr[p_\alpha < (1-\epsilon)\mathbf{E}p_\alpha] < 1/n^2$.*

Lemma 3.1 follows immediately from Chernoff's Inequality (see Appendix). It suggests that we should measure the size of the interval spanned by $\Omega(\ln n)$ successive points and scale the observed density. Two issues remain: (a) How do we estimate $\ln n$ itself? (b) Exactly how many points suffice to arrive at an estimate for n lying in the range $n/(1 \pm \delta)$? Both issues are addressed by the following scheme: Consider a specific node with id $\mathbf{x}$. Let n_i denote the number of nodes that share the top i (most significant) bits with $\mathbf{x}$. Node $\mathbf{x}$ identifies the largest ℓ such that $n_\ell \geq 16(1+\delta)\delta^{-2} \ln(2^\ell n_\ell)$.

LEMMA 3.2. *(a) $\log_2(2^\ell n_\ell) > 0.5 \log_2 n$ with probability at least $1 - 1/n^2$. (b) $2^\ell n_\ell$ lies in the interval $n/(1 \pm \delta)$ with probability at least $1 - 2/n^2$.*

THEOREM 3.1. *With probability at least $1 - 2/n$, the estimate of network size made by every node lies in the range $n/(1 \pm \delta)$.*

Lemma 3.2 is proved in the Appendix. Theorem 3.1 can be derived from Lemma 3.2 by summing over all n nodes.

3.2 Topology Establishment

In the previous Section, we described a distributed scheme which ensures that the estimate of n by all nodes lies in the range $n/(1 \pm \delta)$ w.h.p. On a log-scale in base two, the difference between the upper and lower bounds is $\log_2[(1+\delta)/(1-\delta)]$. Setting $\delta < 1/3$ makes the range of ambiguity less than 1. Let us see how this moves us a step closer to our goal of emulating arbitrary parallel routing topologies.

We label a node with estimate $\tilde{n}$ by $\langle \lfloor \log_2 \tilde{n} \rfloor, \lceil \log_2 \tilde{n} \rceil \rangle$. At most three integers are used in labeling all nodes and at least one integer is common to all labels. A similar labeling can be done for emulating families of networks defined over $k2^k$ nodes. A labeled node constructs two sets of long links, one set for each integer in its label. A message could initially follow links corresponding to the smaller of the two integers at the source, switching over to the next larger integer along the way if necessary. This idea works except for a caveat that calls for clustering.

A Case for Clustering

a) When emulating certain families of parallel networks, uneven distribution of points causes problems. Recall the emulation of a butterfly network with parameter k. A node with id $\mathbf{x}$ and rank r would try to make a connection with a node of rank $(r+1) \bmod k$ which belongs to the group containing the point $\mathbf{x} + 1/2^{(r+1) \bmod k}$. However, it is quite possible that the target group is empty or the target group has too many points. In fact, it is possible to show using Chernoff bound techniques [22] that w.h.p., there exist groups with no members and groups with $\Omega(\ln^2 n)$ members. In Chord, this does not result in serious problems during topology establishment except that some nodes have $\Omega(\ln^2 n)$ links w.h.p. For other networks like butterflies, the problem needs to be addressed to make emulation feasible.

b) When emulating topologies like Chord and de Bruijn networks, uneven distribution shows up in their analysis. For example, the intuition behind the proof that routes are $O(\ln n)$ on average in either of these networks proceeds as follows. First, show that a majority of the most significant bits become zero because no node is bereft of long-distance links corresponding to these bits. Next, show that the last few steps required for routing in a local neighborhood are not too many because the density of points in a small neighborhood has small variance.

In the next Section, we develop a clustering scheme that not only enables emulation of arbitrary families of parallel networks but also provides a common proof technique for analyzing such networks.

3.3 Clustering

LEMMA 3.3. *Let k be such that $2^k \leq (\epsilon^2 n)/(8 \ln n)$. With probability at least $1 - 2/n$, the number of points in each of 2^k equi-sized non-overlapping sub-intervals of $[0,1)$ lies in the range $(1 \pm \epsilon)n/2^k$.*

PROOF. From Lemma 3.1, we conclude that with probability at least $1 - 2/n^2$, the number of points in a specific sub-interval lies in the range $(1 \pm \epsilon)n/2^k$. Summing over $2^k \leq n$ intervals, we obtain the desired bound. □

Lemma 3.3 suggests a natural clustering scheme. We label a node with estimate $\tilde{n}$ with a pair of integers $\langle k_1, k_2 \rangle$ where $k_1 = \lfloor \log_2(\epsilon^2 n)/(16 \ln n) \rfloor$ and $k_2 = \lceil \log_2(\epsilon^2 n)/(16 \ln n) \rceil$. Assuming $\delta < 1/3$ in the estimation scheme (Section 3.1), at most three k-values are used for labeling all nodes and at least one k-value is common to all labels. For each k-value used in a label, Lemma 3.3 assures us that each of 2^k clusters will be populated by $(1 \pm \epsilon)n/2^k$ node ids w.h.p.

A family of parallel interconnection networks is emulated by constructing an inter-cluster network as follows. A node with label $\langle k_1, k_2 \rangle$ makes two sets of links. The first set corresponds to using k_1 most significant bits of its id and assuming 2^{k_1} clusters. The second set corresponds to using k_2 most significant bits and assuming 2^{k_2} clusters. When establishing a particular link, a node can choose any node belonging to the destination cluster. Since at least one integer is common to all labels, there is at least one value of k such that the network over 2^k clusters is complete. Each cluster has $\Theta(\ln n)$ nodes.

A hash lookup initially follows links corresponding to the smaller of the two k-values at the source. Along the way, routing switches to the next higher k-value if necessary.

Upon reaching the destination cluster, intra-cluster routing is done by some local routing network. The choice of local routing topology is influenced by several factors like replication, fault tolerance etc. Since each cluster has size $\Theta(\ln n)$, intra-cluster routing takes no more than $O(\ln n)$ hops.

Maintenance of multiple networks for different k-values costs no extra overhead in terms of links in hypercubes and de Bruijn networks. For butterflies, the number of links at most doubles. Global routing could be faster if all nodes could identify the k-value that is common to all labels. Indeed, the common k-value can be estimated quite accurately by sampling a small number of random nodes.

The paradigm of first routing to the destination cluster and then to a node within the cluster underlies the analysis of existing protocols like Chord and Koorde. By making the distinction explicit and breaking the problem into two sub-problems, it is possible for the two to be developed in practice as more or less independent sub-systems. Our construction also supports emulation of Butterflies, CCC and Benes networks [17].

Partition Balance

A partition of $[0, 1)$ is a sub-interval that is managed by a node. From Lemma 3.3, the ratio of cluster sizes is at most $(1+\epsilon)/(1-\epsilon)$ where ϵ is a small constant. This suggests that it might be possible to move nodes around within cluster boundaries in order to obtain almost equi-sized partitions. However, any movement of nodes potentially impacts the estimation scheme. We are currently developing efficient strategies for carrying out partition balancing that work in conjunction with network size estimation.

3.4 Related Work

Estimation Scheme

Our estimation scheme has similarities with Flajolet and Martin's approximate counting technique [9]. Recently, the idea was adapted to estimate distinct values in streaming data [11]. The intuition behind the scheme is also similar in flavor to the argument that the height of random binary search trees on n keys is $\Theta(\ln n)$ w.h.p.

A scheme for estimating $\ln n$ was presented in Viceroy [19]. If x is the difference between two adjacent ids, then $\ln(1/x)$ is a constant-factor approximation of $\ln n$ w.h.p. Moreover, it can be shown that if y denotes the union of sub-intervals managed by $16 \ln 1/x$ nodes, then $1/y$ is a factor-2 approximation of n w.h.p. The motivation for a new scheme stemmed from partition balancing considerations which call for adjustments in node ids.

Partition Balancing

Naor and Wieder [24] and Abraham et al [1] recently showed that the ratio between the largest and the smallest partition can be made $O(1)$ if a node first chooses $O(\ln n)$ points at random and then selects as its id that point which splits the largest partition. Adler et al[2] have devised algorithms to optimize the same metric for CAN [26].

Emulation of Parallel Networks

Abraham et al [1] recently described a construction for emulating families of graphs dynamically. Members of a family are required to possess a certain kind of recursive structure that allows parent-child functions to have a property called child-neighbor commutativity. The authors show that hypercubes, de Bruijn graphs and butterflies can be defined recursively so as to enjoy the property.

The general construction in Section 3 was derived independent of [1]. It appears that the primary advantage of the new construction is that the family of graphs being emulated need not have a recursive structure. In fact, the graphs over 2^k and 2^{k+1} clusters could be quite different, say a torus and a butterfly. The construction has an additional advantage from a systems perspective. It splits the routing problem into two: global and local, which could be architected in a practical system by separate groups. A global routing designer faces a rather unchanging set of clusters with even density. Her concerns include global load balance across clusters, congestion avoidance, deadlock prevention and high throughput. A local routing designer focuses on local issues like manager replication, fault tolerance and last-hop optimizations, independent of global routing.

Abraham et al [1] view the set of node ids as a binary search tree with keys only among the leaves. A key corresponds to the fewest possible number of most-significant bits necessary for a node to distinguish it from its neighbors. The difference in the lowest and the highest leaf levels is called the global gap. The authors show that choosing the shortest key among $O(\ln n)$ randomly chosen node ids results in global gap $O(1)$ w.h.p. This could in fact be exploited to devise a more efficient scheme for estimating n. Also, it seems that clustering (based upon the estimation scheme of Section 3.1) coupled with partition balancing could provide an alternate method for reducing the global gap to $O(1)$.

3.5 A Variant of Chord

A Chord node establishes roughly $\log_2 n$ outgoing links with managers of points lying at distances $\langle 1/2, 1/4, 1/8, \ldots \rangle$ away from itself. A node also has incoming links from managers of points lying at distances $\langle -1/2, -1/4, -1/8, \ldots \rangle$. The total number of TCP connections is $2\log_2 n$ on average. Average latency by using Chord's clockwise greedy routing protocol [29] is $\frac{1}{2}\log_2 n$. Instead, if every node maintains $2\log_3 n$ links at distances $\langle \pm 1/3, \pm 1/9, \pm 1/27, \ldots \rangle$, we get a reduction in both average latency and average degree. The idea is that the distances to any destination can be written in ternary using the digits $\{-1, 0, +1\}$. Only two-thirds of all digits are ± 1 on average. Thus average latency is $(2\log_3 n)/3$ using only $2\log_3 n$ links. The scheme works in conjunction with the partition balancing technique described in Section 3.3 which ensures that $\sigma < 2$. The idea can also be used to define butterfly networks in base-3 which would offer better latency and out-degree as a function of n.

4. RANDOMIZED TOPOLOGIES

A randomized topology is not determined by the set of node ids alone. In fact, there is a large set of possible topologies from which one is chosen at run-time depending upon the random choices made by all participants.

Randomized topologies have three sources of uncertainty: (a) The total number of nodes is not known accurately, (b) The distribution of ids is not even, and (c) Different nodes make different random choices. The intuition underlying randomized topologies has little to do with the first two sources of uncertainty. It is possible to first devise randomized protocols on a cycle graph with n vertices. As a second step, uncertainty in the knowledge of n and uneven distri-

bution of points can be taken into account. We illustrate this approach by first building intuition common to several known randomized topologies over cycle graphs of size n. We then describe a new topology which is quite simple and offers $O(\ln n)$ average latency with only $O(\ln \ln n)$ links. We then build a sophisticated routing protocol that offers the optimal latency vs degree trade-off. The analysis of the last protocol is for the general setting where we deal with all three sources of uncertainty.

A Case for Randomization

One might wonder whether it makes sense to add more randomness to the system since a system designer already has her hands full dealing with uncertainty about n and the distribution of ids. We argue that randomness in topology contributes to the overall robustness of the system. It makes the system resilient to malicious attacks. Random topologies are typically more flexible since each node chooses its neighbors independently. Deterministic topologies are less flexible as they require coordination among different nodes to guarantee correctness of routing protocols.

4.1 Previous Work

Viceroy [19] was the first randomized protocol for DHT routing. It is an adaptation of butterfly networks. Kleinberg [15] discovered routing protocols over a class of random graphs such that average latency is only $O(\ln^2 n)$ while each node has out-degree two. Kleinberg's construction was inspired by the desire to mathematically model the Small World Phenomenon [21]. Symphony [20] showed how Kleinberg's construction could be adapted to dynamic P2P networks with multiple links per node.

Consider a cycle graph on n nodes where vertices are labeled $0, 1, 2, \ldots, n-1$ and there is an edge between node i and node $(i+1) \bmod n$. A message can be routed clockwise from a node to any other in at most $n-1$ steps. By the introduction of a few more links per node, routing can be made significantly faster.

Assume that a message destined for node $\mathbf{x_{dest}}$ is sitting at node $\mathbf{x_{src}}$. Let $d = (n + \mathbf{x_{dest}} - \mathbf{x_{src}}) \bmod n$, the distance between the nodes. Let h denote the number of 1's in $\mathbf{x_{dest}} \oplus \mathbf{x_{src}}$, the Hamming distance between the two nodes. There seem to be two fundamental themes lying at the heart of existing routing protocols: A route diminishes either the distance d or the Hamming distance h to the destination. CAN, Chord, Kleinberg's protocol, Symphony and Viceroy are designed with d in mind. Pastry, Tapestry and de Bruijn based networks are designed with h in mind. Routes that diminish d do not necessarily diminish h and vice versa. However, the intuition behind both flavors of routing has commonalities, e.g., a protocol gradually diminishes the number of 1's in either d or h. We now present a unified picture of protocols that diminish distance d.

Distance Halving

Consider the function $\mathcal{C}_n(x) = (\ln nx)/\ln n$ for $x \in [\frac{1}{n}, 1]$. This is the cumulative probability distribution of $\mathcal{P}_n(x) = 1/(x \ln n)$ for $x \in [\frac{1}{n}, 1]$. For $x \in [\frac{1}{n}, 1]$, we will say that its *notch value* is $y = \mathcal{C}_n(x)$. While routing, let the current distance to the destination be $x_{current}$ with notch value $y_{current}$. Let $s = 1/\log_2 n$. If the current node has a link with notch value between $y_{current} - s$ and $y_{current}$, then we can forward the lookup along this link such that $x_{current}$

is at least halved and $y_{current}$ diminishes by at least s. The maximum number of times $x_{current}$ can be halved (and $y_{current}$ diminished by s) is at most $1/s = \log_2 n$. This intuition underlies all DHT protocols that diminish distances.

Chord topology corresponds to every node establishing exactly $\log_2 n$ links corresponding to notch values $\langle 1 - s, 1 - 2s, 1 - 3s, \ldots \rangle$. When a node wishes to route to a point $x_{current}$ away (with notch value $y_{current}$), it can immediately forward the lookup along a link such that $x_{current}$ is at least halved and $y_{current}$ diminished by at least $s = 1/\log_2 n$. Lookup latency is thus $O(\ln n)$.

In Kleinberg's construction [15], each node establishes one long link with another node at a distance drawn from a discrete distribution which is quite similar to $\mathcal{P}_n$. This is equivalent to choosing a notch value uniformly at random from $[0, 1]$. Routing proceeds clockwise greedily. If the long link takes us beyond the destination, the request is forwarded to a node's successor. Otherwise, the long link is followed. Let us denote the current distance to the destination by $x_{current}$ with notch value $y_{current}$. With probability $s = 1/\log_2 n$, the long link of the current node has notch value lying between $y_{current} - s$ and $y_{current}$. Thus the expected number of nodes that need to be visited before we arrive at a node which halves $x_{current}$ is $1/s = \log_2 n$. Effectively, in comparison with Chord, there is an inflation in lookup latency by a factor of $O(\ln n)$. Kleinberg's routing scheme requires $O(\ln^2 n)$ steps.

Symphony extends Kleinberg's idea in the following way. Instead of one long-distance per node, there are k long-distance links where $k \leq \log_2 n$. Effectively, a node gets to choose k notches uniformly from $[0, 1]$. Loosely speaking, when we are at $x_{current}$ (with notch value $y_{current}$), we need to examine roughly $(\log_2 n)/k$ nodes before we encounter some link that diminishes $x_{current}$ by at least half. Thus average latency for Symphony is $O((\ln^2 n)/k)$.

Greedy Routing

Barriere et al [4] show that greedy routing using $\mathcal{P}_n$ requires $\Omega(\ln^2 x)$ steps. Aspnes et al [3] study two variants of greedy routing. For ℓ links per node and any fixed distribution, they prove that one-sided routing (clockwise and never overshoot the target) requires $\Omega(\ln^2 n/(\ell \ln \ln n))$ hops. For two-sided routing, they prove a lower bound of $\Omega(\ln^2 n/(\ell^2 \ln \ln n))$ hops and conjecture that this can be improved to match the bound for one-sided routing.

In light of the abovementioned results pertaining to greedy routing and harmonic distributions, the protocol we build next seems interesting. It employs a variant of $\mathcal{P}_n$ but routing is not greedy. For small $\ell \leq \ln \ln n$ links, average latency is only $O((\ln n \ln \ln n)/\ell)$. For large ℓ, average latency is $O((\ln n/\ln \ell) \ln(\ln n/\ln \ell))$.

4.2 Bit-Collection Protocol

Consider a cycle graph on n nodes. Let $b = \lceil \log_2 n \rceil$ bits. A node with id $\mathbf{x}$ chooses an integer r uniformly at random from the set $\{1, 2, \ldots, b\}$ and establishes a link with node $\lceil \mathbf{x} + n/2^r \rceil \bmod n$. The construction can be looked upon as a modification of Chord where each node is restricted to use exactly one entry chosen uniformly at random from its finger table. It is possible to route clockwise in $\Theta(\ln n \ln \ln n)$ steps w.h.p. by using a non-greedy protocol.

Let the distance remaining to the destination be d. Let $b' = \lceil \log_2(4b \ln b) \rceil$ bits. If the long link of the current node

corresponds to one of the top (most significant) $b - b'$ bit positions where d represented in binary has a 1, then forward the message along the long link. Otherwise, forward the message clockwise along the short link. Forwarding along a long link removes some 1 among the top $b - b'$ bits. The lower order b' bits act as a counter that diminishes by 1 whenever a short link is followed.

The protocol is reminiscent of the classic Coupon Collection problem [22]. Essentially, we have to collect at most $b - b'$ coupons where the probability of collecting a coupon in one step is $1/b$. It is well known that w.h.p., all $b - b'$ bits can be collected in $2b \ln b$ steps. Building upon this intuition, it can be shown that on average, routing requires $O(b \ln b)$ hops. Since $b = O(\ln n)$, average latency is $O(\ln n \ln \ln n)$.

With $\ell \le \ln b$ links chosen uniformly out of the b possible, it can be shown that average latency diminishes to $O((\ln n \ln \ln n)/\ell)$. With $\ln \ln n$ links, average latency is only $O(\ln n)$. For large values of ℓ, a further improvement is possible. The key idea is that ℓ links can be used to fix $\lfloor \ln_2 \ell \rfloor$ bits in one hop. It can be shown that for large ℓ, routing requires $O((\ln n / \ln \ell) \ln (\ln n / \ln \ell))$ hops.

The basic Bit-Collection protocol works even for degree-3 Chord described in Section 3.5 where routing is not always clockwise. The idea can also be carried over to hypercubes where every node chooses one of the hypercube edges uniformly at random. This would create a variant of Pastry [28]/Tapestry [13] that routes in $O(\ln n)$ with only $O(\ln \ln n)$ links w.h.p.

Towards Optimality

With at most ℓ links per node, we can reach fewer than ℓ^d nodes in $d - 1$ hops. Therefore, average path length for lookups originating at any node is $\Omega(\ln n / \ln \ell)$.

Bit-Collection is only a factor $O(\ln(\ln n / \ln \ell))$ more expensive than the best possible protocol. How could we possibly make it faster? By chaining the bits being collected. We illustrate the idea for a network with n nodes. Consider a node $\mathbf{x}$ with a finger that should point to $\lceil \mathbf{x} + n/2^r \rceil \bmod n$ for some integer r. This finger fixes the rth most significant bit. If we could make it point to a node that fixes the $(r+1)^{th}$ bit, then we could hope to collect bits rapidly in succession. The key idea is to search for a pair of nodes, one each in the vicinity of $\mathbf{x}$ and $\mathbf{x} + n/2^r$ that both fix the $(r + 1)^{th}$ bit. The two searches on average require only b steps each. How would routing work? If $\mathbf{x}$ wishes to send a message to some node, we first search for a node in the vicinity of $\mathbf{x}$ that fixes the top bit. This requires b steps on average. Then, routing proceeds rapidly by fixing successive top-order bits. A problem that emerges is that searches associated with the top order bits collectively introduce a bias of roughly $O(b^2)$. If every node maintains an additional pointer that points a fixed distance b away, the last stretch of length $O(b^2)$ can be covered in only $O(b)$ steps.

The intuition developed in the previous paragraph is exactly how Viceroy [19] would work if all nodes knew n precisely. Using the terminology of notches developed earlier in this Section, Viceroy assigns each node a notch value drawn uniformly at random from the set $\{1 - s, 1 - 2s, 1 - 3s, \dots\}$. The size of the set is $\log_2 n$. The relationship with Chord is the following. A Chord node uses the entire set for link establishment resulting in $\log_2 n$ links per node. However, a Viceroy node at position $p \in [0, 1)$ and notch value y (corresponding to distance $x = \mathcal{C}_n^{-1}(y)$), searches intervals

centered around points p and $p + x$ for a pair of nodes with notch value $y - s$.

We now develop a protocol that requires only $3\ell + 3$ links per node and offers $O(\ln n / \ln \ell)$ average latency. It is based on Kleinberg's idea and employs the intuition we just developed. Kleinberg's construction assumes that a node does not possess any knowledge of random choices made by other nodes. Our protocol demonstrates that if each node were allowed to gather knowledge of a small number, $O(\frac{\ell}{\ln \ell} \ln n)$, of other nodes, we can construct a topology which diminishes average latency to $O(\ln n / \ln \ell)$ w.h.p. It turns out that our protocol has similarities with Viceroy. The main difference lies in the fact that we allow notch values to be anywhere in the continuous interval $[0, 1]$ while Viceroy limits the choices to $\log_2 n$ discrete values.

5. OPTIMAL RANDOMIZED PROTOCOL

In this Section, we describe a randomized topology with $3\ell + 3$ links per node for average latency $O(\ln n / \ln \ell)$ w.h.p.

Let I denote the unit interval $[0, 1)$. It is convenient to imagine I as a circle with unit perimeter. The binary operators $+$ and $-$ wrap around the interval I. In other words, $x + y$ denotes the point that lies clockwise distance y away from x along the circle. Similarly, $x - y$ denotes the point that lies anti-clockwise distance y away from x.

Let n denote the total number of nodes in the system currently. Each node maintains $3\ell + f + 3$ outgoing links where $\ell, f \ge 1$. We will assume that $\ell = O(\text{polylog}(n))$. A node maintains three real numbers: *position p*, *range r* and *estimate $\tilde{n}$*. Position p is chosen uniformly at random from I. An estimate of the network size $\tilde{n}$ is maintained by using the protocol described in Section 3.1. A node chooses as its range r, a real number drawn from a *range probability distribution* $\mathcal{P}_{\tilde{n}} = 1/(x \ln \tilde{n})$ for $x \in [1/\tilde{n}, 1]$. Distribution $\mathcal{P}_{\tilde{n}}$ is simply the continuous version of the discrete distribution in Kleinberg's construction [15]. A node at position p with range r is said to *span* the interval $[p - r, p] \cup [p, p + r]$.

5.1 Link Structure and Routing Protocol

For $\ell \ge 2$, a node establishes $f + 1$ *short links*, 2 *intermediate links*, 2ℓ *long links* and at most ℓ *global links*. When $\ell = 1$, a node maintains $f + 1$ short links, 1 intermediate link, 2 long links and at most 2 global links. In any case, the total number of links is $3\ell + f + 3$ for $\ell \ge 1$. We will assume that $\ell = O(\text{polylog}(n))$.

Short and Intermediate Links

Short links are established with the $f + 1$ immediate clockwise successors of a node. Only one of these links (with the immediate successor) is crucial for routing. Other links are for fault tolerance and do not play any role in routing.

For $\ell \ge 2$, intermediate links are established with two nodes that are $\lceil \ln \tilde{n} \rceil$ and $\lceil \ln \tilde{n} / \ln \ell \rceil$ hops away in the clockwise direction along the circle. When $\ell = 1$, only one intermediate link is established with the node that is $\lceil \ln \tilde{n} \rceil$ hops away in the clockwise direction. Intermediate links are used to route when the target is known to be nearby. In particular, Lemma 5.3 will show that a node that is $O(\ln^2 n / \ln \ell)$ hops away is reachable in only $O(\ln n / \ln \ell)$ steps.

Long Links

Long links are established as follows. A node partitions the interval $[p - r, p]$ into ℓ non-overlapping equisized sub-

intervals and establishes one long link per sub-interval. It establishes ℓ additional links by partitioning the interval $[p, p+r]$ into ℓ non-overlapping equisized sub-intervals. Note that $[p-r, p]$ and $[p, p+r]$ would have more than one point in common if $r \geq 0.5$.

Let us denote a sub-interval by I_{sub}. Its size is $|I_{sub}| = r/\ell$. Let us denote the mid-point of I_{sub} by p_{sub}. We also define an interval I_{search} with $|I_{search}| = 64 \ln^2 \tilde{n}/(\tilde{n} \ln \ell)$, centered at p_{sub}. Note that $|I_{search}|$ is independent of r. If $|I_{search}| \geq |I_{sub}|/2$, we say that I_{sub} is a *small* sub-interval. Otherwise I_{sub} is said to be a *large* sub-interval. If I_{sub} is small, we establish a link with the manager of the point $p_{sub} - r/(2\ell)$. Mathematically, this allows for multiple links to a node and even self loops. In practice, we could easily avoid both. If I_{sub} is large, we invoke a routine called SEARCH. The goal of SEARCH is to discover some node lying within I_{search} whose range lies in the interval $[3r/(4\ell), 7r/(8\sqrt{\ell})]$. Since $|I_{search}| < |I_{sub}|/2$, the range of such a node covers every point of I_{sub} in its span. Lemma 5.5 will prove that w.h.p., all invocations of SEARCH succeed because $|I_{search}|$ is sufficiently large.

Long links lie at the heart of our protocol. For a node at position p with range r, we claim that all points within $[p-r, p] \cup [p, p+r]$ are reachable by short paths. To reach the manager of some point, we identify the sub-interval to which the point belongs and forward the lookup along that long link that corresponds to this sub-interval. If the sub-interval is small, we arrive at a node such that the destination is no more than $64 \ln^2 \tilde{n}/(\tilde{n} \ln \ell)$ away. At this point, intermediate and short links can carry out further routing. Lemmas 5.2 and 5.3 will show that this requires no more than $O(\ln n/\ln \ell)$ steps. If the sub-interval is large, we arrive at a node whose range is at most $7r/(8\sqrt{\ell})$. The idea is that shrinking by a factor of $7/(8\sqrt{\ell})$ limits the number of long links along any path to $O(\ln n/\ln \ell)$. We will prove our claims formally in Section 5.2.

One aspect of our construction remains. A lookup request can originate at a node that does not include the destination in its span. This might happen if $r < 0.5$. In such a case, how do we reach a node with range large enough to include the destination? Global links solve this problem.

Global Links

Global links are established if the range $r < 0.5$. Consider $I - [p-r, p+r]$ where I denotes the full circle. For $\ell \geq 2$, we partition the interval $I - [p-r, p+r]$ into ℓ equisized sub-intervals having size $(1 - 2r)/\ell$ each. For each sub-interval I_{sub}, we invoke SEARCH with the size and location of I_{search} being similar to our earlier description for long link establishment. The only change is that SEARCH looks for a node with range lying in the interval $[3(1 - 2r)/(4\ell), 1]$. When $\ell = 1$, we partition $I - [p-r, p+r]$ into two equisized sub-intervals with size $(1 - 2r)/2$ each. SEARCH is invoked twice to look for a pair of nodes, one in each sub-interval, with ranges lying in $[3(1 - 2r)/8, 1]$.

Lookup Protocol

When a node initiates a lookup request, it forwards it along that long or global link whose range spans the destination. Thereafter, the request is forwarded along a series of long links until we reach a sub-interval that is small. Hereafter, intermediate and short links are used for routing.

5.2 Theoretical Analysis

We will establish that w.h.p., the worst case routing latency is $O(\ln n/\ln \ell)$ for $\ell = O(\text{polylog}(n))$. The overall proof is as follows. We first show that with probability at least $1 - 2/n$, the estimate $\tilde{n} \in [\frac{n}{4}, 4n]$ for all nodes. Next, we show that small sub-intervals do not have high densities. In particular, we will show that w.h.p., no small sub-interval has more than $O(\ln^2 n/\ln \ell)$ nodes. Next, we will establish that with probability at least $1 - 3\ell/n$, all invocations of SEARCH succeed. The resulting topology enjoys the property that path lengths of lookups are guaranteed to be as small as $O(\ln n/\ln \ell)$. Overall, we would have proved that w.h.p., the worst case lookup latency is $O(\ln n/\ln \ell)$.

LEMMA 5.1. *With probability at least $1 - 2/n$, all nodes in a network of size n have $\tilde{n} \in [\frac{1}{4}n, 4n]$.*

PROOF. From Theorem 3.1 (for sufficiently small δ). $\square$

A *small* sub-interval has size less than $64 \ln^2 \tilde{n}/(\tilde{n} \ln \ell)$.

LEMMA 5.2. *With probability at least $1 - 2/n$, no small sub-interval has more than $O(\ln^2 n/\ln \ell)$ nodes.*

PROOF. Using Chernoff Inequality and Lemma 5.1, we can show that with probability at least $1 - 2/n^2$, a particular sub-interval cannot be dense. Summing over all nodes, we obtain the requisite bound. $\square$

The role of intermediate links is to route quickly to any node that is $O(\ln^2 n/\ln \ell)$ hops away.

LEMMA 5.3. *Intermediate and short links can be followed to reach any node that is $O(\ln^2 n/\ln \ell)$ hops away in the clockwise direction in $O(\ln n/\ln \ell)$ steps.*

PROOF. The longer of the two intermediate links can be followed in succession to reach a node that is at most $O(\ln n)$ hops away. This requires $O(\ln n/\ln \ell)$ steps. Then the shorter of the intermediate links can be followed to reach a node within $O(\ln n/\ln \ell)$ hops of the destination. This requires $O(\ln \ell)$ steps. Finally, $O(\ln n/\ln \ell)$ short links can be followed to reach the destination. Since $\ell = O(\text{polylog}(n))$, the total number of steps is $O(\ln n/\ln \ell)$. $\square$

For small sub-intervals, long and global link establishment always succeeds. If the sub-interval is large, there is a chance that SEARCH fails.

LEMMA 5.4. *An invocation of SEARCH fails with probability at most $1/n^2$.*

PROOF. We will prove the lemma for long links. The proof for global links is along the same lines.

SEARCH is invoked only if $|I_{sub}|/2 > |I_{search}|$. This implies $r/2\ell > 64 \ln^2 \tilde{n}/(\tilde{n} \ln \ell)$, where $\tilde{n}$ is the estimate of the node that invoked SEARCH. Thus, $3r/(4\ell) > 96 \ln^2 \tilde{n}/(\tilde{n} \ln \ell) > 16/\tilde{n}$ for large n. From Lemma 5.1, $16/\tilde{n} \geq 4/n$, which is definitely larger than $1/\tilde{n}$ for any node in I_{search} being probed.

When establishing long links, the goal of SEARCH is to discover some node whose range lies in $[3r/(4\ell), 7r/(8\sqrt{\ell})]$. The probability that the range of a node with estimate $\tilde{n}$ lies in this interval is given by $p = \int_{3r/4\ell}^{7r/8\sqrt{\ell}} 1/(x \ln \tilde{n}) dx$. In the preceding paragraph, we showed that $3r/4\ell \geq 1/\tilde{n}$ for any node in I_{search}. Therefore, the value of the integral is

$(\ln \frac{7}{6}\sqrt{\ell})/\ln \tilde{n}$. From Lemma 5.1, this quantity is at least $(\ln \frac{7}{6}\sqrt{\ell})/(2\ln n)$.

$|I_{search}| = 64\ln^2 \tilde{n}/(\tilde{n}\ln \ell)$. Lemma 5.1 yields $|I_{search}| \geq 8\ln^2 n/(n\ln \ell)$ for large n.

Let us fix the position of the node which invoked SEARCH. Consider the sequence of $n-1$ remaining nodes choosing their positions and ranges one by one. With probability $1 - |I_{search}|$, the position does not lie in I_{search}. Otherwise, with probability at least $(\ln \frac{7}{6}\sqrt{\ell})/(2\ln n)$, SEARCH succeeds. Thus the probability that no node makes SEARCH succeed is at most $[1 - |I_{search}| + |I_{search}|(1 - \frac{\ln \frac{7}{6}\sqrt{\ell}}{2\ln n})]^{n-1} \leq [1 - \frac{(8\ln^2 n)(\ln \frac{7}{6}\sqrt{\ell})}{(n\ln \ell)(2\ln n)}]^{n-1} \leq [1 - \frac{2\ln n}{n}]^{n-1} \leq 1/n^2$. $\square$

LEMMA 5.5. *With probability at least $1 - 3\ell/n$, all invocations of* SEARCH *succeed.*

PROOF. Lemma 5.4 shows that the probability that a particular invocation of SEARCH fails is at most $1/n^2$. Since there are at most $3\ell n$ total invocations, the probability that *any* of them fails is at most $3\ell/n$. $\square$

The next lemma shows that although we allow SEARCH to probe all nodes in a rather large sized I_{search}, the expected number of nodes it needs to probe is much smaller.

LEMMA 5.6. SEARCH *probes an average of $O(\ln n/\ln \ell)$ nodes before succeeding.*

PROOF. In the proof of Lemma 5.5, we proved that the probability that a node in I_{search} makes SEARCH invoked for long links succeed is at least $(\ln \frac{7}{6}\sqrt{\ell})/(2\ln n)$. Thus the expected number of nodes to be probed before we succeed is at most $(2\ln n)/(\ln \frac{7}{6}\sqrt{\ell}) = O(\ln n/\ln \ell)$. The same bound can be established for SEARCH invoked for global links. $\square$

We proved that w.h.p., all global and long links successfully get established. The resulting topology enjoys the property that all lookup paths are short.

THEOREM 5.1. *With probability at least $1 - (6 + 3\ell)/n$, the worst case routing latency with $\ell = O(\text{polylog}(n))$ is $O(\ln n/\ln \ell)$ hops.*

PROOF. From Lemmas 5.1, 5.2 and 5.5, we conclude that with probability at least $1 - (6 + 3\ell)/n$, all estimates of n are within a factor of four, no small sub-interval is dense and all long and global links get established. We show that the resulting graph has short diameter.

Routing proceeds in two phases. In the first phase, a lookup is forwarded along some long or global link whose range is guaranteed to contain the destination. The request then moves along a series of long links such that every node along the path has a range large enough to contain the destination in its span. The first phase starts at some node with range at most 1. From 5.1, when the first phase finishes, the last node will have range at least $1/4n$. Since each long link along the path shrinks the range by at least $\frac{8}{7}\sqrt{\ell}$, the first phase requires no more than $O(\ln n/\ln \ell)$ hops.

The second phase starts when we encounter a node with tiny range such that all its sub-intervals are small. At this point, the destination is only $O(\ln^2 n/(n\ln \ell))$ hops away (Lemma 5.2). Intermediate and small links can reach the destination in $O(\ln n/\ln \ell)$ steps (Lemma 5.3).

Total latency is thus $O(\ln n/\ln \ell)$. $\square$

5.3 Arrival, Departure and Re-linking

When a new node x arrives, it chooses its position p uniformly from $[0, 1)$. It then discovers node y, the manager of point p using the Lookup Protocol. Node x establishes a short link with y. The predecessor of y then terminates its short link with y and establishes it with x instead. This completes the insertion of x into the ring.

Node x estimates the network size $\tilde{n}$ using the procedure outlined in Section 3.1. The estimate allows x to identify nodes for establishing intermediate links at a cost of $O(\ln n/\ln \ell)$ messages. The estimate also enables x to select its range r. Thereafter, x establishes 2ℓ long links and at most ℓ global links. This requires at most 3ℓ invocations of SEARCH. From Lemma 5.6, the average cost of all these invocations is $O(\frac{\ell}{\ln \ell}\ln n)$.

When a node arrives or departs, estimates of as many as $O(\ln n)$ nodes change. If all $O(\ln n)$ nodes re-establish their links, the total cost of joining is $O(\frac{\ell}{\ln \ell}\ln^2 n)$. The cost can be reduced by maintaining two numbers, $\check{n}$ and $\tilde{n}$. Estimate $\tilde{n}$ is continually updated using the network size estimation scheme (Section 3.1). Estimate $\check{n}$ is initialized to $\tilde{n}$ when a node joins. If at any point of time, $\check{n} \notin [\frac{1}{2}\tilde{n}, 2\tilde{n}]$, then links are re-established and $\check{n}$ is made equal to $\tilde{n}$. This scheme was shown to work well in practice [20]. The amortized cost of joining reduces to $O(\frac{\ell}{\ln \ell}\ln n)$.

5.4 Reducing the Out-degree

We briefly outline a construction that requires only 4 links per node for $O(\ln n)$ average latency w.h.p. We set $\ell = 1$ and get rid of global links. Note that a fraction $\approx c/\ln n$ nodes will have their range smaller than c'/n for some constants c and c'. These nodes will not establish long links since their range is tiny. They will instead establish two global links each. Routing now requires that a lookup be forwarded to some node with tiny range. Hereafter, the usual protocol works. We can reduce the number of links to only 3 per node by removing the intermediate link as well. The resulting topology has an average latency of $O(\ln n/\ln \ell)$. However, the high probability bound no longer holds.

6. DISTRIBUTED HASHING REVISITED

Guided by the intuition gained while exploring the solution space of routing networks for DHTs, we propose that the problem be broken into four pieces for practical systems:

(a) Network Size Estimation: In practice, if network size does not change too rapidly, central servers, crawlers or traffic monitors might very well be able to provide fine estimates. What schemes work well? Can we and should we adapt them to work in distributed settings?

(b) Partition Balance: Efficient schemes are needed to help a node choose its initial id upon entry into the system. At run-time, nodes could move around to maintain evenness in sizes of hash table partitions. However, repartitioning is costly. The efficacy of repartitioning schemes and their impact on the rest of the system needs investigation.

(c) Global Routing: The space of routing networks for routing among clusters as defined in Section 3.3, appears to be rich. A global routing protocol designer does not have to worry about uncertainty of n or uneven distribution of node ids. From her standpoint, the routing layer over clusters is rather static. Results from parallel architecture literature might be applicable here. However, the load on parallel

machines is primarily scientific computations. The expectations from the routing layer in the context of peer-to-peer applications are not fully understood yet. Perhaps new routing topologies and protocols that change dynamically in response to load are useful in practice.

(d) Local Routing: System design for routing within clusters is different in nature from global routing. The interplay of several issues like fault tolerance, replication and caching contributes to design complexity.

Routing-related Issues

Network Proximity: For mapping nodes in the real world to ids in $[0, 1)$, two contrasting approaches exist. CAN [26, 27] proposes that geographically nearby nodes should be close in id space too. Such a design is problematic. As the network evolves at different rates in different parts of the world, portions of the hash table have to migrate to ensure partition balance. A more serious concern pertains to network partitions caused by physical layer failures which cause large portions of the hash table to vanish.

An alternative design is to allow nodes to choose their ids independent of their geographical position. For long-distance links, a small interval (within the destination cluster defined in Section 3.3) could be searched for a geographically nearby node. The idea is similar to proximity routing in Pastry [6]. By sampling enough points, a link with a sufficiently close neighbor could be established. An interesting consequence is that short links (around the circle) are actually expensive whereas long links are cheap. It is possible to avoid following large-latency short links for the last few hops if there is sufficient replication and caching. Note that replication of managers is necessitated by fault tolerance concerns alone. The exact trade-offs require investigation.

Fault Tolerance: A simple scheme is to make every node manage partitions of a handful of its neighbors. Assuming that nodes choose their ids at random independent of geography, the effect is to replicate managers for a given partition at geographically diverse locations. This makes the hash table resilient to network partitions caused by physical layer failures. Schemes for replica management, reconciliation and possible oscillations arising out of physical network partitions need be worked out.

Cache Management: A promising application of DHTs appears to be caching of large volumes of relatively static pieces of information for which leases suffice. There is a strong connection between caching and routing. For effective caching, copies of objects should be placed in such a way that routing paths are shortened. This requires an investigation into the interplay between routing topologies, caching policies and leases.

7. CONCLUSIONS & FUTURE WORK

We presented a unified view of DHT routing protocols, highlighting commonalities and differences among various deterministic and randomized schemes. We hope that our synthesis makes the job of system designers easier when they choose among protocols for implementations. Guided by the intuition gained while exploring the design space, we revisited the problem of constructing DHTs routing topologies from a systems perspective. It appears that routing should be split into several black boxes that can be attacked more or less independently. An implementation exploring some of these design issues is underway at Stanford University.

9. 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. Intl. Parallel and Distributed Processing Symp.*, Apr 2003.

[2] M. Adler, E. Halperin, R. M. Karp, and V. V. Vazirani. A stochastic process on the hypercube with applications to peer-to-peer networks. In *Proc. 35nd ACM Symp. on Theory of Computing (STOC 2003)*, Jun 2003.

[3] J. Aspnes, Z. Diamadi, and G. Shah. Fault-tolerant routing in peer-to-peer systems. In *Proc. 21st ACM Symp. on Principles of Distributed Computing (PODC 2002)*, pages 223–232, Jul 2002.

[4] L. Barriere, P. Fraigniaud, E. Kranakis, and D. Krizanc. Efficient routing in networks with long range contacts. In *Proc. 15th Intl. Symp. on Distributed Computing (DISC 01)*, pages 270–284, 2001.

[5] B. Bollobas. *Random Graphs.* Cambridge University Press, 2nd edition, 2001.

[6] M. Castro, P. Druschel, Y. C. Hu, and A. Rowstron. Topology-aware routing in structured peer-to-peer overlay networks. In *Proc. Intl. Workshop on Future Directions in Distrib. Computing (FuDiCo 2002)*, 2002.

[7] J. Considine and T. Florio. Scalable peer-to-peer indexing with constant state. Technical Report 2002-026, Computer Science Deptt., Boston University, Sep 2002.

[8] J. Duato, S. Yalamanchili, and L. Ni. *Interconnection Networks: An Engineering Approach.* IEEE Press, 1997.

[9] P. Flajolet and G. N. Martin. Probabilistic counting. In *Proc. 24th Annual Symp. on Foundations of Computer Science (FOCS 1983)*, pages 76–82, 1983.

[10] P. Fraigniaud and C. Gavoille. The content-addressable network d2b. Technical Report 1349, LRI, Univ. Paris-Sud, France, Jan 2003.

[11] P. Gibbons. Distinct sampling for highly-accurate answers to distinct value queries and event reports. In *Proc. 27th Intl. Conf. on Very Large Data Bases (VLDB 2001)*, pages 541–550, 2001.

[12] S. D. Gribble, E. A. Brewer, J. M. Hellerstein, and D. Culler. Scalable, distributed data structures for internet service construction. In *Proc. 4th Symposium on Operating System Design and Implementation (OSDI 2000)*, pages 319–332, 2000.

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

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

[15] J. Kleinberg. The small-world phenomenon: An algorithmic perspective. In *Proc. 32nd ACM Symposium on Theory of Computing (STOC 2000)*, pages 163–170, 2000.

[16] L. G. Valiant. A scheme for fast parallel communication. *SIAM J. of Computing*, 11:350–361, 1982.

[17] F. T. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays - Trees - Hypercubes.* Morgan Kaufmann, 1992.

[18] W. Litwin, M. Neimat, and D. A. Schneider. LH* - A scalable, distributed data structure. *ACM Transactions on Database Systems*, 21(4):480–525, 1996.

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

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

[21] S. Milgram. The small world problem. *Psychology Today*, 67(1), 1967.

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

[23] N. de Bruijn. A combinatorial problem. *Proc. Kominklitjke Nederlandse Akademie van Wetenschappen*, 49:758–764, 1946.

[24] M. Naor and U. Wieder. Novel architectures for p2p applications: The continuous-discrete approach. In *Proc. 15th ACM Symp. on Parallelism in Algorithms and Architectures (SPAA-2003)*, Jun 2003.

[25] C. G. Plaxton, R. Rajaraman, and A. W. Richa. Accessing nearby copies of replicated objects in a distributed environment. In *Proc. 9th ACM Symposium on Parallel Algorithms and Architectures (SPAA 1997)*, pages 311–320, 1997.

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

[27] S. Ratnasamy, M. Handley, R. M. Karp, and S. Shenker. Topologically-aware overlay construction and server selection. In *Proc. IEEE INFOCOM-2002*, Jun 2002.

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

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

APPENDIX

Chernoff Inequality: Let $X_1, X_2, \ldots, X_t$ denote independent Bernoulli variables with probability of success $p_i \in [0,1]$ for $1 \leq i \leq t$. Let $X = \sum_{i=1}^{t} X_i$ and $\mu = \mathbf{E}X = \sum_{i=1}^{t} p_i$. Then for any $0 < \epsilon < 2e - 1$, $Pr[X > (1+\epsilon)\mu] < \exp{-\mu\epsilon^2/4}$ and $Pr[X < (1-\epsilon)\mu] < \exp{-\mu\epsilon^2/4}$.

Proof of Lemma 3.2(a): If $2^k n_k \leq \sqrt{n}$, then $(1 - 1/2^k) \leq (1 - n_k/\sqrt{n})$. We assume that $n_k < n$ for otherwise, there is no error in estimate. Let us fix the ids of the nodes that contribute to n_k. The probability that none of the remaining $n - n_k$ nodes chooses its id in the interval of size $1/2^k$ is given by $(1 - 1/2^k)^{n-n_k} \leq (1 - n_k/\sqrt{n})^{n-n_k} \leq e^{-(n-1)/\sqrt{n}} < 1/n^2$ for large n.

Proof of Lemma 3.2(b): We know that $n_\ell \geq 16(1 + \delta)\delta^{-2}\ln(2^\ell n_\ell)$. Part (a) above assures us that $\ln 2^\ell n_\ell > 0.5 \ln n$ with probability at least $1 - 1/n^2$. Therefore, $n_\ell \geq 8\delta^{-2}(1 + \delta))\ln n$ with probability at least $1 - 1/n^2$.

n_ℓ successive points are expected to lie in a sub-interval of size n_ℓ/n. However, we observed n_ℓ to lie in a sub-interval of size $1/2^\ell$. The probability that $1/2^\ell$ does not lie in the range $(1 \pm \delta)n_\ell/n$ is given by $Pr[|1/2^k - n_\ell/n| > \delta n_\ell/n] \leq Pr[1/2^\ell < (1-\delta)n_\ell/n] + Pr[1/2^\ell > (1+\delta)n_\ell/n]$. We now prove that the first term is at most $1/n^2$. The proof for the second term is along similar lines.

Consider the probability $Pr[1/2^\ell < (1-\delta)n_\ell/n]$. This is identical to the probability $Pr[p_{(1-\delta)n_\ell/n} > n_\ell]$ (using the definition of p_α from Lemma 3.1). We can rewrite it as $Pr[p_{(1-\delta)n_\ell/n} > (1+\epsilon)(1-\delta)n_\ell/n]$ where $\epsilon = \delta/(1-\delta)$. From Lemma 3.1, this probability is less than $1/n^2$ as long as $\alpha = (1-\delta)n_\ell/n > (8\epsilon^2 \ln n)/n$, which is indeed true for $\epsilon = \delta/(1-\delta)$.