Location-based Node IDs: Enabling Explicit Locality in DHTs

Shuheng Zhou, Gregory R. Ganger, Peter Steenkiste

September 2003
CMU-CS-03-171

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

Current peer-to-peer systems based on DHTs struggle with routing locality and content locality because of random node ID assignment. To address these issues, we promote the use of location-based node IDs to encode physical topology and improve routing. This gives applications explicit knowledge about and control over data locality at a coarse-grain. Applications can place content in particular regions or route towards a close replica. Schemes to address the difficulties that ensue, particularly load imbalance, are discussed.

We thank the members and companies of the PDL Consortium (including EMC, Hewlett-Packard, Hitachi, IBM, Intel, Microsoft, Network Appliance, Oracle, Panasas, Seagate, Sun, and Veritas) for their interest, insights, feedback, and support.

Keywords: distributed hash table, peer-to-peer, routing locality, hierarchical routing, content locality, load balance

1 Introduction

Structured peer-to-peer overlay networks such as CAN [18], Chord [24], Pastry [20] and Tapestry [25] provide a scalable and robust data location and request routing service for large-scale distributed applications such as decentralized storage [4, 21, 13] and content distribution [2]. They provide a Distributed Hash Table (DHT) function to map keys to overlay nodes using variants of consistent hashing [11] algorithms. A node participating in an overlay is assigned a random identifier from a numeric space, e.g., through hashing of its IP address. A document key associated with a data item obtained from the same numeric space is mapped onto a node whose identifier immediately succeeds the key. This node is called the document root. Storage load is naturally balanced since each node receives roughly the same number of keys. DHT routing algorithms form logical connections between nodes based on their identifiers in the numeric space and messages are routed to their destinations through $O(\log N)$ hops. Different DHT algorithms define their set of routing table entries differently, forming different underlying routing geometries [8], but the size of the routing table entries is normally $O(\log N)$ in a system of N nodes. Finally, when nodes join or leave, $O(\log^2 N)$ messages [24] are exchanged to maintain routing consistency and to transfer keys.

Many problems exist with the deceptively simple DHT structure that could defeat DHTs' purpose of constructing highly available and robust wide-area services. First, current routing algorithms optimize the path length to $O(\log N)$, but the expansion of an overlay path, defined as the ratio between the overlay path communication cost and the shortest network distance between source and destination, can be quite high since the overlay topology is not congruent with the physical topology; each hop can potentially traverse a long-haul link. This issue is normally refered to as routing locality. While locality aware DHTs [17, 18, 20, 25] aim to reduce the overlay path expansion, they achieve it at an increased protocol complexity and overlay construction and maintenance cost. Second, DHTs achieve storage load balance at a high communication cost: physical distances between document roots and document origins can be long, thus causing registration and possibly query traffic to traverse wide area links. This is especially undesirable when content is mostly of local interests such as local traffic updates and news. Third, replication, which is an important aspect of data availability and durability, in current systems [4, 21], relies entirely on randomization, depriving applications of any control of replica locations. We refer to both issues above as the loss of content locality [9], i.e., the capability that systems should possess to explicitly control data locations. Finally, DHTs constantly re-balance load by transferring documents between neighboring nodes in the numeric ring when nodes join or leave, generating even more traffic across wide area long-haul links.

We think the fundamental cause of DHT's struggle with routing and content locality is the random nature of node IDs, through which no information about a node's own physical location or its proximity to other nodes can be deduced. Worse yet, no information can be aggregated regarding any subset of nodes along the numeric ring without active probing. This is against the wisdom [5] that addresses need to be assigned along network topological lines to maximize the reduction in routing overhead such as routing table size and amount of route advertisements for reachable prefixes. Randomness in ID assignment is perfect for storage load balance, but at the cost of increased wide-area communication. In response to this, we propose a new DHT architecture that embeds hierarchy in the node identifier space to address routing and content locality issues while keeping the DHT's desirable properties, especially, load balance, deterministic location, decentralization, and robustness to dynamics of nodes and links.

We construct a structured ID space through a hierarchical location-based node ID assignment. A location-based node ID is a concatenation of a hierarchical prefix assigned to a node's region and a suffix of randomly generated bits. By assigning prefixes in a hierarchical manner, the network topology information can be efficiently embedded into a DHT's logical structure and used by the routing system. Neighborhood relations of regions along the ring reflect their proximity relations in the physical space and the logical distances between regions reflect their physical distances. When DHT routing makes progress in the numeric space, similar progress is made in the physical space and overlay path costs are bounded within a constant factor of physical path costs, thereby achieving an $O(1)$ expansion. A constant expansion is achieved with low protocol complexity and routing table construction cost; messages will traverse the longest physical hop first toward the remote region before it follows smaller steps approaching the destination within the remote region. Document transfers for load-rebalancing now traverse local-area links between neighboring nodes along the ring instead of wide-area long-haul links. In addition, hierarchical routing will further scale the system by reducing the routing state maintained at each node and localizing routing information propagation; especially, the influences of node or link dynamics are largely confined within a leaf region.

Whereas our work is not the first to propose location-based node ID assignment, we are the first to address the important load balancing issues that ensue. We argue for the separation of load balancing concerns for storage from routing. For routing load balance, we assign prefixes to regions with the length of the prefix reflecting their node density and communication capacity. This guarantees a uniform distribution of nodes along the ID space so that each node assumes roughly the same amount of routing cost in terms of incoming and outgoing links, route computation and bandwidth for transit traffic. For storage, we allow applications to embed location prefixes in document keys; instead of sending keys to random locations for global load balance, an application can decide its region of load balance so long as it obeys a localized communication pattern, which ultimately enables true system robustness and scalability. The structured ID space therefore provides an abstraction from which applications can learn physical network topology and make informed decisions to achieve content locality. We inherit the definition of content locality from SkipNet [9], to refer the ability of applications to explicitly place data in specific locations in the system.

The rest of the paper is organized as follows. Section 2 describes the structured ID space and provides a system overview. Section 3 discusses its advantages for routing. Section 4 discusses the explicit control applications gain for content locality. Section 5 discusses prefix coding algorithms. Section 6 compares our approach with related work and Section 7 concludes.

2 Structured ID Space

In this section, we first present an overview of the structure of location-based node IDs and document keys. We then present the construction of a hierarchical location-based ID space and the embedding of network distances in the numeric space. Finally, we argue for the load-balance aware prefix assignment and discuss its flexible usage through subnetting and supernetting.

2.1 Overview

Region	Sub–region	Leaf region	Random bits

Figure 1: Structure of a node ID with hierarchical prefixes

The structure of a location-based node ID is shown in Figure 1. It consists of a prefix with components that are hierarchically related to its containing regions and a suffix of randomly generated bits significant only within the leaf region. The entire ID is a random value from the leaf region identifier space. The boundaries between different components of a prefix are not fixed, since the size of each component is related to the node density of the region it represents. Physical locations and network distances are effectively embedded in the numeric space, inducing congruity between overlay topology and physical topology, and thereby improving routing locality. Routing hierarchies can be built to further scale the routing system.

Similar to node IDs, document keys can be a concatenation of two strings, a prefix that has a semantic meaning and a suffix of randomly generated bits; a string of entirely random bits is still valid. Applications control data locality by exploiting the semantics of key prefixes. For example, registration of information such as local news and traffic update should be constrained within a leaf domain by embedding content publisher's location prefix in the document key. Document roots thus reside in the leaf domain whereas the random bits help spread keys across the leaf domain. On the other hand, document keys comprised of entirely random bits spread popular content across the global region, regardless of the publisher's location, to better serve a uniform access pattern. Access locality for popular data will rely on replication and caching.

This architecture deviates significantly from current DHT designs where node IDs and documents keys are randomly generated and a distributed hash function is used to spread keys evenly among the nodes. A particularly important question concerning embedding location prefixes in node IDs and document keys is: what is the influence on the load balance, the desired property of DHTs? We answer this question from two aspects: routing load and storage load, discussed in Section 3 and Section 4 respectively.

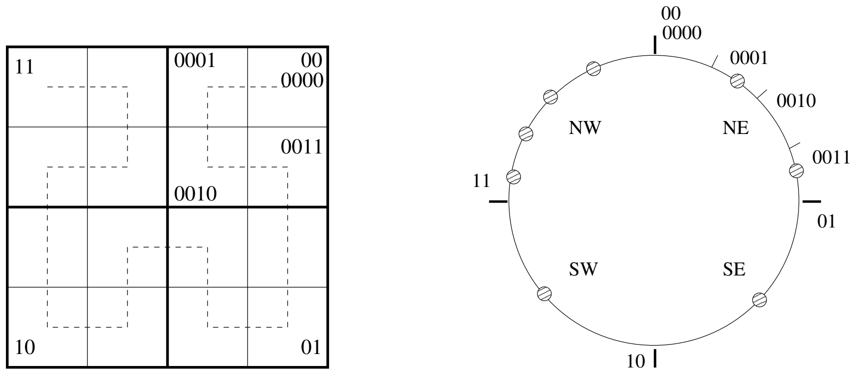
2.2 Node Locations

Region and location can be geographical, network topological, geometric or administrative entities depending on the metric we use to define a space and construct a hierarchy. Locality of nodes refers to their proximity relations in the metric space. One can construct a hierarchy of arbitrary number of levels. We use region, sub-region and leaf region, as shown in Figure 1 liberally to refer to top, intermediate and bottom levels of the hierarchy. We use physical space and distance to refer to the metric space and its metric. Here are some examples of how one can partition the underlying network into a hierarchy based on different metrics.

- Geographical hierarchy: use a quadtree to cover a planar region of interest with a square and then recursively partition squares into smaller squares until each square contains a uniform subset of nodes.
- Geometric hierarchy based on network distance: divide an N-dimensional Euclidean geometric space recursively into sub-spaces, where geometric distances between nodes approximate the Internet network distances, as explored in GNP [16].
- Hybrid hierarchy similar to the Internet: first divide geographical regions along continental boundaries, then divide each continent based on organizational hierarchies with geographically proximate cites of AS domains connecting to regional network service providers as leaf nodes.

The construction of geographical hierarchies is straightforward. The construction of hierarchies reflecting network topology will have to rely on a routing underlay [15] or more generally, a knowledge plane [3], where information such as the AS domain peering graph [15] that represents the coarse-grain connectivity of the Internet are gathered that one can query from. Evaluation of different hierarchies is an important future work.

A hierarchical prefix encoding scheme based on geography without load balancing concern is shown in Figure 2, where 2-D geographical regions at each level are divided into four sub-regions and contiguous prefixes $00, 01, 10, 11$ are assigned to regions according to their spatial order along the Hilbert Curve, with NE quad being 00 and NW quad being 11. Prefixes correspond to segments along the numeric ring with a shorter prefix representing a longer numeric range. When each range is subdivided into smaller ranges corresponding the sub-regions, more prefix bits are appended to the existing range prefix to form a longer prefix representing the sub-region; this process repeats until a leaf region with a pre-determined size is reached, where size could be defined in terms of the node count, geographical coverage or sites of administrative domains. The partition of regions and assignment of prefixes at each level of the hierarchy might be a centralized algorithm, but the overall scheme is decentralized and scalable since each region is responsible for its own subregion partition and prefix assignment; therefore, only a relatively small number of top level



(a) Hilbert curve across 2-D space (b) Numeric ring comprising four US regions

Figure 2: Quadtree-based prefixes

regions need to obtain prefixes from a central authority, and a delegatable method similar to the allocation of network numbers can be adopted. Each node retrieves its region prefix through a bootstrap mechanism by contacting nodes that belong to the same leaf region, which most likely will be a stub AS domain.

2.3 Embed Network Distances in the Ring

We will show in Section 3 how preserving physical locality in the numeric space will improve routing locality without requiring extra probes from individual nodes to the networks. In this subsection, we explore the embedding of physical distances in the numeric space and its implications on routing algorithms. The logical distance between nodes in Chord is the numeric difference between IDs, while for prefix routing, it is the matching of prefix digits, where two IDs are considered closer to each other when they share more high-order bits. Forward progress for Chord is purely numerical steps in the clockwise direction, while for prefix routing, it is the matching of more high-order prefix digits. For Chord [24] to take full advantage of the structured ID space, one can use space filling curves, such as the Hilbert Curve in Figure 2(a), to map regions in a hierarchical high-dimensional physical space onto a one-dimensional numeric ring and assign contiguous prefixes to neighboring regions along the ring. The spatial ordering of regions in the physical space is kept in the numeric space continuously and numeric distances between nodes reflect their physical distances. The concept of space filling curve exists for any number of dimensions.

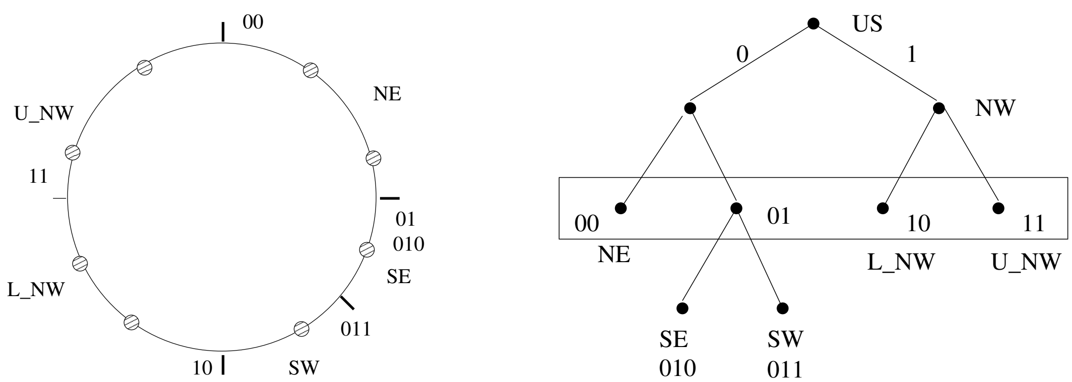
For prefix routing [17, 20, 25], inter-region alignment need not obey a strict order since messages are forwarded through hops that match prefix digits with the destination ID without traversing regions in between. Proximity relations along the numeric ring become discrete since they are disjunctive across certain fixed boundaries along the ring; sub-regions under each parent are relatively adjacent to each other. This discontinuity occurs recursively at a fixed set of boundaries. A fixed division of prefix digits into different components is necessary to guarantee the relative continuity of proximity relations within each region. These fixed boundaries are formed through prefix aggregation and division to be explained in details in the next subsection. More flexible routing algorithms, such as Pastry [20], that combine prefix and chord routing algorithms, send messages toward nodes with IDs closer to the destination ID in numeric distance when entries matching more prefix digits are not available. A strict spatial order, especially among top level regions are necessary to achieve good routing locality. Finally, spatial ordering across the entire physical space as in Figure 2(b) is desirable when one wish to do prefix summarization at any level to form super-regions of arbitrary size while keeping the adjacency property for their containing regions.

2.4 Load-balance Aware Prefixes

Due to the heterogenous node density and capacity distributions across different geographical regions, the naive scheme in Figure 2 results in a skewed node distribution along the numeric ID space. For example, the NW segment comprising both Silicon Valley and Seattle is much more densely populated with nodes than the SW segment. Since a node in the numeric ring is responsible for storing keys mapped onto the range between its previous identifier and itself, SW nodes will assume higher storage load since each node is responsible for a longer segment along the ring. In addition, they will receive large amount of inbound query traffic for keys mapped to themselves and possibly transit traffic from NE and SE toward NW as an intermediate hop depending on the DHT routing algorithms.

A balanced node distribution along the numeric ring is essential for both routing and storage load balance assuming a uniform distribution of document keys. We preserve the uniform node distribution by allocating prefixes to regions with corresponding segment lengths proportional to their node density and capacity, which we expect to be relatively stable, especially for large regions at the top of the hierarchy. We will discuss the implications of non-uniform key distributions intentionally generated for content locality on load balancing in Section 4. Till then, we assume a uniform key distribution in the same numeric space as node IDs.

An example prefix allocation that populates nodes uniformly across the numeric ring is shown in Figure 3(a). Densely populated regions such as NW occupy half a ring and sparsely populated regions such as SE and SW each occupy $1/8$ of the ring. A manifestation of the previous warning that boundaries between different components of a prefix are not fixed is shown: at the same level of hierarchy, prefixes of different lengths `1 3 3 2` are assigned to the four top US regions. For prefix routing or hierarchical routing (to be discussed in Section 3), one might want to divide prefixes at fixed boundaries to generate prefix components of desirable lengths, thereby forming artificial regions with similar node capacity. We use the term artificial region to contrast with natural physical regions, such as AS domains or geographical regions to refer to the regions that we form out of fixed-length prefixes.



(a) Balanced ring reflecting region node density (b) Subnetting and Supernetting

Figure 3: Load-balanced prefixes

We introduce abstractions similar to IPv4 subnetting and supernetting, where division and summarization of prefixes at any location is possible. As shown in Figure 3(b), a subnetting analogy is drawn to divide top level region NW into two sub-regions, Lower NW and Upper NW, each assigned a prefix of 2 bits, 10 and 11 respectively. The supernetting concept is applied to summarize SE and SW prefixes 010 and 011 to a shorter prefix 01, thus forming a super-region of SE+SW. Super-regions must be formed out of proximate sub-regions for routing entries matching summarized prefixes to make sense. This is similar to IP address prefixes summarization [5], where a set of longer prefixes can be summarized into a single short prefix to aggregate multiple routing entries into a single entry. Prefixes that can be summarized must represent sites adjacent in network topology so that one single route entry provides reachability information for all of them as seen from a remote region. Finally, the hierarchical classification of nodes in the US into four equal node capacity top level regions $\{$`NE, SE+SW, L_NW, U_NW`$\}$ leads to a tree representation as shown in Figure 3(b); the tree structure can be easily extended to represent a general multi-level hierarchical partitioning that incorporates sub-regions and leaf-regions.

The division and summarization processes can be recursively applied until regions satisfying certain requirements such as the prefix length, node capacity or geographical coverage are reached. For prefix routing algorithms [20, 25], where fixed-length prefix digits are continuously matched at each routing step to match destination ID, subnetting and supernetting help form prefixes of the same length out of regions of heterogeneous node density and capacity. As an example, when prefix routing routes a message two bits at a time toward its destination, a message to node Id with prefix 11 will be sent to the Upper NW region while a message to node Id with prefix 01 will be sent to the SE+SW super-region at the first step. However, during forwarding, the prefix with the longest match will identify the routing entry to be used, just as in the Internet. In prefix routing, the longest match will be the matching with the most number of high order bits, while Chord selects the entry that has the closest numeric distance to destination ID. Algorithms for prefix encoding are discussed in Section 5.

In hierarchical routing to be discussed in Section 3, subnetting and supernetting really means sub-regioning and super-regioning, through which load balanced regions with the same node capacity are formed to participate in routing at the higher level of the hierarchy. Finally, alternative algorithms such as geographical routing can be applied to route across top level regions; a service that maps prefixes to geographical regions or network topological regions is needed to fully take advantage of the semantic meaning embedded in the node prefix. Construction of such a service is orthogonal to the Subnetting and Supernetting concept, which is concerned with forming artificial regions out of the natural geographical or topological regions.

3 Routing

A decade ago, network address assignment along network topological lines was proposed to maximize the reduction in routing overhead and ultimately scale inter-domain routing systems to the Internet [5]. The specific scaling problems on the Internet are related to memory and computational overhead for routing information, bandwidth for routing information distribution, and the stability of distributed routing computations. Despite the salient differences between IP routing and DHT routing, we explore a similar theme in this section: the opportunities for routing improvement through hierarchical location-based IDs. Specifically, we achieve overlay routing locality by exploiting the topology information embedded in the structured ID space and explore hierarchical routing to further scale DHT systems.

We place nodes and document keys around a ring for the analysis of DHT algorithms, as Chord and Pastry do. This does not influence the applicability of our analysis to other DHT routing algorithms, since the numeric ring is common to all DHTs that use a one-dimensional identifier space. Techniques here can be equally applied to CAN despite its different addressing scheme. We avoid making distinctions between different algorithms as much as we can to generalize our design and analysis.

3.1 Improving DHT Routing Locality

In a structured ID space, one can follow exactly the same routing algorithms as existing DHTs; they scale to $O(\log N)$ in the number of routing table entries and overlay path length to reach a destination. Given a key, a DHT routes the request to a node immediately succeeding the key in the numeric ring. We assume document keys are unique and randomly generated for the moment. In current DHTs, each node keeps a set of links to other nodes satisfying certain requirement, e.g., Chord [24] establishes fingers to nodes in powers of 2 distances away while Pastry and Tapestry [17, 20, 25] maintain routing table entries that share an increasing number of common prefixes with the current node. In addition, Pastry keeps a leaf set composed of nodes with IDs closest in numeric distance from current node, which makes it a hybrid routing algorithm. Messages are forwarded through $O(\log N)$ number of intermediate hops, whose nodes IDs are progressively closer to the destination. Locality-aware DHTs [17, 20, 25] reduce overlay hop cost by picking a proximate subset of the nodes that satisfy the prefix requirement to set up logical connections.

Unlike current locality-aware DHTs, location-based ID assignment improves routing locality without requiring individual nodes to probe the network for proximity information when they establish connections to other nodes. This is due to the appropriate embedding of network distances along the numeric ring as explained in Section 2.3. For Chord routing, whose forward progress is defined as the numerical distance traversed in the clockwise direction, low expansion overlay paths will be achieved via space filling curves as in Figure 2(a); when an overlay path makes forward progress in the numeric space, it makes forward progress along a certain physical direction as well, due to the spatial alignment of regions along the numeric ring as shown in Figure 2(a). At the same protocol complexity, this is a clear advantage over current Chord routing, whose overlay path can be rather erratic, zigzagging from Boston, west to San Francisco, and east back to Europe.

With prefix routing, whose routing table is shown in Figure 4(b), the first overlay hop connects directly to a remote top level region that shares the destination's top order prefix digits and messages stay within a remote region once they reach it. Assuming network distances within each region are shorter than inter-region distances, a low expansion of overlay path is even more tractable than Chord since when DHT routing algorithms make forward progress in the numeric space with non-increasing steps, costs of overlay hops are also non-increasing with a safe landing within the remote destination region at first stride; while for Chord, traversing regions in the middle of the numeric ring could still be slightly deviating from a straight path toward the destination region. This is in clear contrast with current locality-aware DHT routing [20, 25] where communication paths take overlay hops at increasing costs with the last hop being the most expensive and most difficult to optimize. While the number of hops in an overlay path is still $O(\log N)$, the communication cost is bounded within a constant factor to the actual physical distance at low protocol complexity.

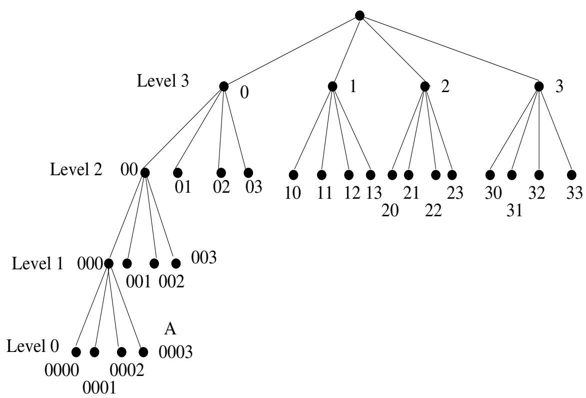
3.2 Routing Hierarchy

In their classical paper [12], Kleinrock et al. presented the scalability properties of hierarchical routing: in the limit of a very large network, enormous routing table reduction may be achieved with essentially no increase in network path length. DHT routing table size, which is logarithmic in number of nodes in the system, has relatively good scaling properties due to the logical hierarchies embedded in the routing table structures; Chord forms a structure that resembles a skip-list while prefix routing encodes a tree structure as shown in Figure 4. The routing table size reduction follows exactly the same idea described in [12]. They keep detailed routing information about nodes that are close-by and coarser information about nodes located further away, both in terms of logical distance, which is realized by providing one entry per destination for the neighboring nodes, and one entry per set of destinations located on remote segments along the ring; the size of this set increase exponentially with the logical distance.

However, certain assumptions are made in Kleinrock [12]'s analysis in order to guarantee no increases in network path lengths despite the hierarchical clustering structure of nodes. Specifically, among others, they assume that the diameter of any k^{th} level cluster subnet is less than or equal to a quantity $d_k, k = 1, ..., m$, with m being the levels of routing hierarchy, d_m representing the diameter of the entire network, and $d_k > d_{k-1} > 0$ for all k. Mapping to DHT prefix routing algorithm, m is $O(\log N)$ and d_k is the diameter of any set of nodes that share $m - k$ common prefix digits; d_m is still the diameter of the entire

Internet and d_0 is the nodes that share all common prefixes. Since worst case distance between any two nodes in current DHTs is the diameter of the entire Internet, $d_k = d_m, k = 1, ..., m$, hierarchical clustering of nodes based on numeric IDs in current DHTs breaks the assumptions in Kleinrock [12] that guarantee bounded network path lengths.

Now that we have aligned the physical topology along the numeric ring, numeric hierarchies in current DHTs map naturally to geographical or network topological hierarchies. Routing entries at the top of the hierarchy, such as the longest Chord fingers or entries that share no common digits with current node in prefix routing, connect to remote regions while entries in the bottom of the hierarchy connect to nodes within the leaf region. Assumptions in Kleinrock [12] are easily satisfied and this again proves our assertion that overlay path lengths are bounded within a constant factor of physical paths.



(a) A tree representation of a 4-level hierarchy

Level 3	0***	1***	2***	3***
Level 2	00**	01**	02**	03**
Level 1	000*	001*	002*	003*
Level 0	0000	0001	0002	0003

(b) Prefix routing table for node A that embeds a natural hierarchy

Figure 4: Prefix routing and its natural hierarchy

The salient difference between DHT's entrenched hierarchy as in Figure 4 and a more explicit hierarchy, as in the Internet inter-domain routing systems, is that every node can become the representative node of its entire top-level cluster, since that responsibility is essentially randomized over the entire set of nodes in that cluster; representative here means that the node appears in the routing table of some other nodes for reaching its top level cluster. Desirable as it is for nodes to assume equal responsibilities, that can only cause problems especially when nodes join or leave, since $O(\log^2 N)$ messages are needed to maintain routing table consistency. We postulate that further reduction in routing overhead is possible through reducing the entire set of nodes in a cluster to a robust subset as representative nodes for that cluster. This maximal reduction in routing table size and update propagations will result in recovery of storage, network capacity, and computation power, especially from those due to the fluctuations of ups and downs of overlay nodes and links, thereby improving system robustness and scalability. This further reduction can come in two forms: implicit and explicit. Notice clusters in current DHTs are formed out of nodes that share common prefixes or nodes that are numerically close, while in our system, clusters naturally map to regions in a hierarchical manner, which enables the further reduction of routing overhead without influencing robustness.

With implicit hierarchical routing, one instance of DHT routing is running across the entire system. However, each node can reduce its full set of routing table entries that flat DHT routing forms to a useful subset based on the levels of routing they wish to participate in. For example, a Chord node that has advertised itself to remote regions and established a long link to a node $N/2$ distance away in a system of N nodes has decided to route at the highest level of the hierarchy. In the contrary, a node has only short fingers reaching nodes within the same leaf region participates routing only in the base level. A node can choose to have an arbitrary set of long or short incoming links and/or outgoing links, depending on its willingness to receive and/or send traffic across regions. A message should always be forwarded toward the destination while taking advantage of the longest routing table entry in each intermediate node preceding the destination ID. A similar implicit routing hierarchy can be designed based on prefix routing.

Similar to IP address prefixes in CIDR, prefixes of location-based IDs provide a hierarchical abstraction that enables explicit hierarchical routing in DHTs. For example, we can build an explicit two-tier routing hierarchy based on the hierarchy encoded in Figure 3(a). We partition US regions logically into several DHT routing domains, where an instance of DHTs such as Chord or Pastry is running to maintain reachability information for the entire set of node identifiers within each region. The partition of routing domains can be based either on a natural partition of the physical space, or can be based on the artificial regions formed through prefix division and summarization as in Figure 3(b) so that each routing domain contains roughly the same number of nodes. A subset of nodes from each region form a virtual top-level DHT ring where an inter-region routing protocol is running to establish the connections across regions.

This inter-region routing protocol could be another DHT routing protocol, or could combine geographical and Internet inter-domain routing notions, which we will not elaborate on in this work further. The size of each subset that participates in high-level routing should be proportional to its region node density so that the top-level ring is uniformly populated with nodes as well. These nodes, similar to BGP border routers, advertise reachability information for the entire region that share a common prefix and possibly regions that it want to carry traffic for. They propagate route changes into remote regions and build up routing table entries that interconnect regions so that messages can be routed across the entire numeric ring. In fact, one can build a routing hierarchy of an arbitrary number of levels by partitioning regions recursively and by having each node choose randomly how many levels of routing to participate in based on its communication capacity and computation power, which is more robust than having a static subset of nodes participating in the routing at each level.

Routing hierarchy does not influence robustness in our system since the logical links established between a set of adjacent nodes in one leaf region and the mirroring set of nodes in a remote region tend to share the same underlying physical paths; therefore only those that traverse different physical paths should be kept. Communication costs and processing of routing updates, however, is reduced to the complexity based on the number of regions or the number of the subset of nodes that participate in routing at each level, rather than on the total number of nodes in the system. In addition, the impact of node joins and leaves will be confined within the highest level region in which they participate in routing. Thus a low degree of dynamics within any leaf region or subregion does not sum to a high degree of dynamics across the entire ring. The direct measure for gains obtained from hierarchical routing will be a reduction in routing table size and routing overhead caused by node joins or leaves, which translates to recovery in storage, network capacity and computational power. Finally, different DHT algorithms can be deployed in different regions so long as there exists a clean interface between them and the inter-region routing algorithms. This enables region autonomy and isolates faults within each routing region effectively.

3.3 Routing Load Balance

Nodes participate in routing, message forwarding, document publishing and query processing for keys and their associated documents. We separate the role of nodes acting as an endpoint of a communication path from the role of acting as a transit hop, and leave the first issue to be discussed at Section 4. DHT routing overhead involves the storage for routing table entries, the processing of route updates when nodes join and leave, and the passing of transit traffic through the node for key registration and query. The actual data retrieval associated with keys normally does not go through overlay hops once an IP address is resolved. By populating nodes uniformly across the identifier space, we achieve routing load balance since each node has roughly the same number of incoming and outgoing logical connections and carries roughly the same amount of transit traffic given a uniform and random key distribution. For hierarchical routing, nodes with more communication bandwidth and computational power will participate in higher levels of the routing hierarchy. With a good randomized algorithm, statistically, the number of nodes in each region participating in a certain level of DHT routing should be proportional to its node density, ensuring routing load balance across regions at each level.

3.4 Dynamic Behavior and Evolution

In the absence of large-scale correlation among node join or leave activity in a region, the influence of individual join or leave will naturally balance out so that the node density distribution remains relatively stable most of the time. Especially, a subset of nodes from each region are expected to have long alive time so that they can perform cross-region routings, document storage and retrievals even when other nodes within the region are dormant or disconnected during off-peak hours. One potential drawback of the system is that once one allocates a certain segment size to a region, the ID space for nodes in that region is fixed. Thus, when segments change sizes due to node density changes, a certain amount of documents will have to be moved. At the top level, though, the relative ratio of node density, computing power and network capacity do not change dramatically in a short period of time. Therefore, the region node densities should be relatively stable at both a macro and micro scale of time intervals so that document transfers across regions, especially across top level regions, will not be tremendous over a short period of time. This will be important to verify in future work.

4 Content Locality

As mentioned in Section 2, applications control data locations through the manipulation of document keys. In this section, we illustrate the advantages a structured ID space gains for content locality through three example replication scenarios. Applications use replication for different purposes such as availability, performance and durability. Current replica placement schemes [4, 21] put replicas in nodes sequential in the numeric space, hoping the random distrubtion of identifiers will spread replicas across geographical regions. Instead of relying on nodes sequential in the ID space, we will have to use explicit names to represent a replica root set, thus avoiding the correlation of node failures in the same leaf region. This is not a disadvantage since at a moderate lookup cost for discovering names of a replica root set, applications gain explicit control over replica locations, which is unattainable in current DHTs of the same complexity. We assume users obtain keys and their associated replica root set through a separate naming service using systems such as CDS [7], which resolves a unique document key and a replica root set from a high-level descriptive name such as attribute value pairs. The encoding of a replica root set can be very simple in our design. Separation of naming and location service is a conscious design decision following [22], and we focus on a location service that resolves a unique key identifier to a replica location.

4.1 Variations on a Theme of Load Balance

Whereas random hashing is great at spreading load evenly across nodes in the entire system, the downside is the loss of content locality since applications have no control over data placement. Worse yet, communications caused by registration and query of keys, and the actual data store and retrieval is a vast waste of wide-area bandwidth. In order to give applications the control for data locality and to form a better communication pattern, we draw a clear distinction between load balance for DHT routing which is desirable due to its shared nature, and load balance for storage despite their intricate interactions.

Routing table entries and routing update processing remain invariant regardless of key distribution since they are determined by the interconnections among nodes based on numeric IDs. Key forwarding cost will be influenced largely by registration and query locality. When a large number of keys are mapped to a particular region, nodes along the lookup path will be influenced by the corresponding query traffic. Nodes in remote regions and earlier in the lookup path will be less influenced than nodes with IDs closer to the keys, which will be inundated by the query traffic from all directions.

Whereas we allow an application to decide its region of storage load balancing, we require registration and query traffic generated by the application to be contained within the load balancing region. We present several simple replication mechanisms to illustrate how applications can satisfy this requirement while achieving content locality. By localizing registration and query traffic for keys, rippling effects along the registration and lookup path will be minimized. Within a load balancing region, traffic concentration can be reduced by the provision of extra storage and communication capacity. Finally, when nodes join or leave, DHTs re-balance load by transferring documents between nodes that are adjacent in numeric space; with location-based node IDs, document transferring becomes a local behavior since nodes that are sequential in the numeric ring are also close in the physical distance.

4.2 Deterministic Replica Locations

In order to achieve deterministic replication, an application should decide its replication policy and communicate that to the system so that the system creates a unique replica root set for each document key. For example, an application can decide to replicate for load balance across entire systems or it may decide to replicate across different geographical regions, which the system can deduce from the mapping between prefix and region. In addition, obtaining failure distributions across a well structured identifier space will be much more tractable, which is important for applications that demand high availability [1]. Finally, caching instead of replication should be used to handle flash crowds, and each replica can in turn act as a proxy root for deeper replication and caching to address dynamic access pattern. We introduce three examples that support efficient replication and retrieval to illustrate different policies.

Uniform Replication with Bit Mask. We introduce a concept called *bit mask* to let an application specify its desired level of replication for a particular document key and allow a client to access a close-by replica. For example, a bit mask with four preceding 0s followed by all 1s such as `00 00 11 11` for an 8-bit ID space, would specify a replica root set of 16 nodes that are uniformly distributed across the numeric ring as shown in Figure 5(a). Nodes in any of these regions that best match the document key in bits not masked off will be picked as a replica root. The registration and discovery of the tuple $\langle key, bitmask \rangle$ as the replica root set is another level of indirection to be addressed by the naming service mentioned earlier. After the client gets the tuple, it sends a request directly towards its local replica root. The ID is calculated from the key by replacing the masked-off prefix bits indicated by zeros with its own prefix of the corresponding length. Without a bit mask, the client would have to do an expanding search to discover a close-by replica, replacing its own prefixes with the document key prefix following a leaf to region order.

Replication Favoring Adjacent Domains. Some applications might prefer placing more replicas in close-by domains relative to the document origin rather than remote regions as shown in Figure 5(b). There is a simple way to achieve this: first, encode the publisher domain prefix in the document key; then replicate the document to a set of nodes with IDs $\{2^i | \ 0 \leq i < logN\}$ distances away from the root, where N is the number of nodes in the system. After the client gets the unique document key, it can calculate the node IDs in the replica root set and send a request to a replica whose ID is numerically closest to its own. A bit mask can be combined with this scheme to indicate the level of replication instead of using a full set of $O(\log N)$ replicas.

Locality for Key Partitions. So far, we have ignored a possible skewed distribution of hashed keys. In some peer-to-peer applications such as the Content Discovery System [7], some keys can be very popular

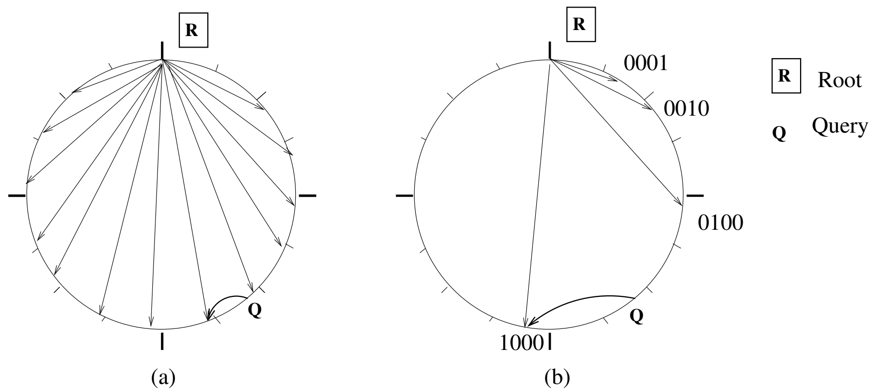


Figure 5: Illustration of Replication Patterns

and partitions will be used to distribute entries mapping to the same key to different storage nodes through rehashing. Location can thus be used as a factor to decide the specific locations for holding the partition set. As an example, all music files that have an attribute value pair of {`composer = Mozart`} will be hashed to the same key and therefore will initially be stored in the same node. When partitioning happens, all entries originating from a leaf region will be assigned to a partition node within the same region. The query is directed to the local partition as well, for access locality, since there is almost no need to return all the matching entries for a popular key. Entries hashed to one popular key are thus partitioned across different domains obeying registration and access locality.

5 Prefix Encoding Algorithms

Prefix encoding algorithms [10, 14, 23] assign codewords, which are sequences of bits, to symbols such that no codeword is a prefix of another; codewords are assigned in a way that the most frequent symbols are given shorter code lengths for information compression. Instead of assigning prefixes to symbols, we assign prefixes to regions. The equivalence of the frequency of a symbol is the relative node density/capacity in a region; nodes belonging to the same region count as the same symbol and share the same prefix. Thus, the more nodes a region has, the shorter its prefix is. This matched exactly to our observation in Section 2, where prefix lengths were exactly $-log(p_i)$ when probability distribution p_is is dyadic, i.e., integral powers of 2, with p_i denoting the relative region node density. We prove that segments assigned to regions are disjoint along the numeric ring.

Because of the ordering constraints of regions along the numeric ring as stated in Section 2, Huffman codes [10] do not directly apply. We can however use the code lengths generated via Huffman to determine the segment lengths for each region and assign prefixes contiguously to regions according to their predetermined order while fulfilling their desirable segment lengths; if no available short prefix exists for a region, we split the region to subregions and encode subregions with as many longer contiguous prefixes as necessary. Huffman codes are optimal in their efficiency of code lengths. Prefixes assigned to segments corresponding to different regions at the same level of the hierarchy via Huffman have the following properties.

PROPERTY 1 Each region occupies a segment of size 2^{n-l_i} with 2^n being the size of the entire numeric ring, and l_i being the length of the prefix assigned to the region. The shorter the prefix length assigned, the larger the segment size is. This satisfies our desire to allocate a larger segment for a region with higher node capacity.

PROPERTY 2 Segments corresponding to codewords are disjoint along the numeric ring.

PROOF 1 A property of prefix coding. □

PROPERTY 3 The sum of the segment sizes occupied by all the regions is 2^n, size of the entire numeric ring.

PROOF 2 Define a complete binary code tree [6] as a finite order tree in which each intermediate node has 2 nodes of the next higher order stemming from it. A complete code tree is one in which the Kraft inequality [6] is satisfied with equality. Since a Huffman code tree is a complete code tree, we have $\sum 2^{-l_i} = 1$ applying Kraft inequality, where l_i is codeword length for $1 \leq i \leq n$ and n is the size of the set of input symbols.

The set of input symbols to Huffman encoding are relative region node capacities and the set of codeword lengths l_is are the lengths of prefixes assigned to regions. The sum of the segment lengths $\sum 2^{n-l_i}$ is therefore 2^n, which is the size of the entire numeric ring. □

PROPERTY 4 For a dyadic node capacity distribution, segment size of each region is exactly $p_i * 2^n$, where p_i is the relative node capacity for region i and $\sum p_i = 1$

PROOF 3 Segment size for region i is $2^{n-l_i} = 2^{n+\log(p_i)} = p_i * 2^n$. □

From Property 4, we can see segment size of a region is proportional to its relative node capacity for a dyadic distribution, therefore achieving a balanced node distribution along the numeric ring when such algorithms are recursively applied within each region to assign prefixes to its subregions. However, if relative node capacities are not dyadic, the numeric space is not perfectly load balanced: certain segments are longer, and certain segments are shorter than their fair share, especially for a skewed distribution. A simplistic approach is to approximate each input probability with a sequence of dyadic numbers through binary expansions, that Huffman code can do well with, with the possibility that a sequence of longer prefixes will be assigned to a region instead of a single short prefix. Designing algorithms and heuristics to bound the code imbalance factor induced by Huffman is an ongoing work.

Before introducing Property 5, we use an example to illustrate the code splitting algorithms given above. If the sequence of relative node capacities of regions are $0.25, 0.15, 0.2, 0.15, 0.25$, with that order required along the numeric ring, Huffman code lengths are $2, 3, 2, 3, 2$. After assigning 00, 010 to 0.25 and 0.15, in that order, there is no two-bits prefix immediately following 010 for 0.2. The best one can do is to assign 011 and 100 to 0.2. The two code words are contiguous along the ring, but do not sharing the same parent in a code tree. This process is the same as splitting one physical region into two sub-regions and assign one longer prefix to each of the sub-regions. For completeness, the entire code for the input distribution that obeys the specific ordering are: $00, 010, 011 + 100, 101, 11$. The split of codeword for 0.2 is caused by the non-adjacency of two least likely probabilities, 0.15 and 0.15, imposed by the input ordering of node capacity distributions. The set of new prefix lengths are $2, 3, 3, 3, 3, 2$, specifically one node of 2^{nd} order is splitted to two 3^{rd} order nodes on the code tree, each of which is assigned to two different regions or sub-regions.

PROPERTY 5 The new code tree corresponds to the above construct is still a binary code tree.

PROOF 4 This new tree is isomorphic to the original Huffman code tree, ignoring the specific bit each branch is assigned, with one or more leaves each splitting into two children. The new tree is still a complete binary code tree, since each leaf that is turning into an intermediate node has two children stemming from it. □

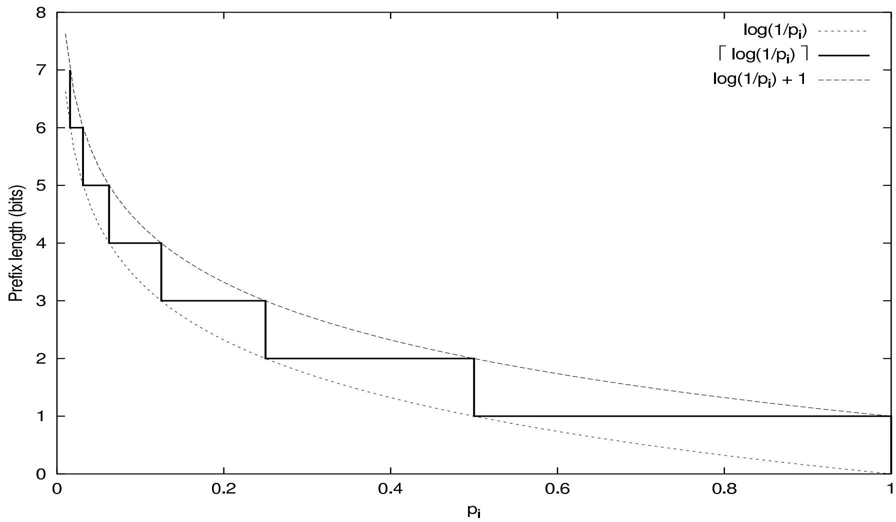


Figure 6: Prefix Length of Shannon Codes

All properties above therefore still apply to the new codes.

Shannon codes [23] have lengths $l_i = \lceil \log 1/p_i \rceil$, which is bounded by: $\log 1/p_i \leq l_i \leq \log 1/p_i + 1$ as shown in Figure 6.

PROPERTY 6 The segment length s_i for a region prefix allocated through Shannon code is between its half and full share of the numeric ring.

PROOF 5 Given $s_i = 2^{n-l_i}$ and $\log 1/p_i \leq l_i \leq \log 1/p_i + 1$, we have $p_i * 2^{n-1} \leq s_i \leq p_i * 2^n$. □

Gaps exist between the segments assigned to regions and we will have to grow a region's segment that is shorter than its fair share to occupy gaps via assignment of extra prefixes. Each region therefore not only occupies its assigned segment, but also some surrounding segments; this approximately balance load which we will prove in future work. Finally, instead of encoding prefixes at each level separately, arithmetic encoding [14] can assign prefixes that blend region, sub-region and domain codewords. Although this might be a theoretically optimal approach with respect to load balancing, it is non-trivial to decentralize and difficult to aggregate the small codes into meaningful shorter prefixes representing high level regions.

6 Related Work

Rather than having each node probe the network for proximity information independently [17, 18, 20, 25], we abstract distance in a systematic and scalable way to build a structured ID space. Our protocol can be as simple as Chord [24] while achieving low expansion routes. CAN [18, 19] explored topology-aware ID assignment by clustering nodes that are physically close to the same virtual coordinate space. A skewed node distribution in the physical networks will result in a skewed node distribution in the logical space. CAN focused more on reducing routing latency than building an entirely structured ID space so that applications can explicitly control data locality. They did not address the load balancing issue either. Current replica placement algorithms [4, 21] rely on randomness to spread replicas across geographical areas while we provide much more control.

Finally, SkipNet [9] organized data by lexicographic ordering of string names to achieve content locality within an administrative domain. Our content locality is more general and applies to different metric spaces such as geographical and topological. Another property that SkipNet addresses is path locality, which refers to the ability to guarantee that traffic between two overlay nodes within the same organization is routed within that organization only. This property is easily guaranteed in our system, since when an AS domain corresponds to a leaf region, traffic between nodes within the leaf region will stay within that AS domain. In fact, traffic between overlay nodes that share any common prefix will stay within the region corresponding to that prefix.

7 Summary

In this paper, we propose abstracting a physical space into a structured ID space where the logical topology is congruent to the physical topology. We propose prefix assignment schemes that guarantee load balancing of the shared routing substrate. We discuss the improvement in routing locality and the feasibility of constructing a routing hierarchy to further reduce routing overhead. Specifically, we localize communications due to dynamic node behavior, without sacrificing robustness. We show that a structured ID space provides applications the ability to control locality and deterministically place replicas, while forming a better communication pattern. Finally, we discuss source coding algorithms for prefix assignment in our system. In future work, we will design detailed algorithms for the construction of location-based ID space and hierarchical routing. We will provide detailed analysis regarding how applications can control content locality via a structured ID space. Lastly, feasibility and load balancing property of different prefix encoding algorithms will be quantified.

References

[1] Ranjita Bhagwan, David Moore, Stefan Savage, and Geoffrey Voelker. Replication strategies for highly available peer-to-peer storage. *International Workshop on Future Directions in Distributed Computing* (Bertinoro, Italy, June 2002), 2002.

[2] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, Animesh Nandi, Antony Rowstron, and Atul Singh. SplitStream: High-bandwidth multicast in a cooperative environment. *ACM Symposium on Operating System Principles* (Lake Bolton, NY, October 2003), 2003.

[3] David D. Clark, Craig Partridge, J. Christopher Ramming, and John Wroclawski. A knowledge plane for the Internet. *ACM SIGCOMM Conference* (Karlsruhe, Germany, 25–29 August 2003). ACM, 2003.

[4] Frank Dabek, M. Frans Kaashoek, David Karger, Robert Morris, and Ion Stoica. Wide-area cooperative storage with CFS. *ACM Symposium on Operating System Principles* (Chateau Lake Louise, AB, Canada, 21–24 October 2001). Published as *Operating System Review*, **35**(5):202–215, 2001.

[5] Peter S. Ford, Yakov Rekhter, and Hans-Werner Braun. Improving the routing and addressing of the Internet protocol. *IEEE Network*, **7**(3):10–15. IEEE, May 1993.

[6] Robert G. Gallager. *Information theory and reliable communication*. John Wiley and Sons, Inc, 1968.

[7] Jun Gao and Peter Steenkiste. Design and evaluation of a distributed scalable content discovery system. Published as *IEEE JSAC Special Issue on Recent Advances in Service Overlay Networks*, 2003.

[8] Krishna P. Gummadi, Ramakrishna Gummadi, Steven D. Gribble, Sylvia Ratnasamy, Scott Shenker, and Ion Stoica. The impact of DHT routing geometry on resilience and proximity. *ACM SIGCOMM Conference* (Karlsruhe, Germany, 25–29 August 2003). ACM, 2003.

[9] Nicholas J. A. Harvey, Michael B. Jones, Stefan Saroiu, Marvin Theimer, and Alec Wolman. SkipNet: A Scalable Overlay Network with Practical Locality. *USENIX Symposium on Internet Technologies and Systems* (Seattle, WA, March 2003). ProUSENIX Association, 2003.

[10] D. A. Huffman. A method for the construction of minimum-redundancy codes. Published as *Proceedings of the IRE*, **40**:1098–1101, 1952.

[11] David Karger, Eric Lehman, Tom Leighton,, Matthew Levine, Daniel Lewin, and Rina Panigrahy. Consistent hashing and random trees: tools for relieving hot spots on the World Wide Web. *ACM Symposium on Theory of Computing* (El Paso, TX), pages 654–663. ACM, 1997.

[12] Leonard Kleinrock and Farouk Kamoun. Hierarchical Routing for Large Networks, Performance Evaluation and Optimization. Published as *Computer Networks*, **1**(3):155–174, 1977.

[13] John Kubiatowicz, David Bindel, Yan Chen, Steven Czerwinski, Patrick Eaten, Dennis Geels, Ramakrishna Gummadi, Sean Rhea, Hakim Weatherspoon, Westley Weimer, Chris Wells, and Ben Zhao. OceanStore: an architecture for global-scale persistent storage. *Architectural Support for Programming Languages and Operating Systems* (Cambridge, MA, 12–15 November 2000). Published as *Operating Systems Review*, **34**(5):190–201, 2000.

[14] G. G. Langdon. Arithmetic coding. (March 1979). Published as *IBM J. Res. Develop.*, **23**(2):149–162. IBM, 1979.

[15] Akihiro Nakao, Larry Peterson, and Andy Bavier. A routing underlay for overlay networks. *ACM SIGCOMM Conference* (Karlsruhe, Germany, 25–29 August 2003). ACM, 2003.

[16] T. S. Eugene Ng and Hui Zhang. Predicting Internet network distance with coordinates-based approaches. *IEEE INFOCOM* (New York, June 2002). IEEE, 2002.

[17] C. Greg Plaxton, Rajmohan Rajaraman, and Andrea W. Richa. Accessing nearby copies of replicated objects in a distributed environment. *ACM Symposium on Parallel Algorithms and Architectures* (Newport, RI, June 1997). Published as *Proceedings of the 9th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 311–320. ACM, 1997.

[18] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content-addressable network. *ACM SIGCOMM Conference* (San Diego, CA, 27–31 August 2001), pages 161–172. ACM, 2001.

[19] Sylvia Ratnasamy, Mark Handley, Richard Karp, and Scott Shenker. Topologically-aware overlay construction and server selection. *IEEE INFOCOM*, 2002.

[20] Antony Rowstron and Peter Druschel. Pastry: scalable, decentralized object location and routing for large-scale peer-to-peer systems. *IFIP/ACM International Conference on Distributed Systems Platforms* (Heidelberg, Germany, 12–16 November 2001), pages 329–350, 2001.

[21] Antony Rowstron and Peter Druschel. Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility. *ACM Symposium on Operating System Principles* (Chateau Lake Louise, AB, Canada, 21–24 October 2001). Published as *Operating System Review*, **35**(5):188–201. ACM, 2001.

[22] Jerome H. Saltzer. On the naming and binding of network destinations. RFC 1498, August 1993.

[23] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, **27**:379–423. Bell Systems, July 1948.

[24] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: a scalable peer-to-peer lookup service for internet applications. *ACM SIGCOMM Conference* (San Diego, CA, 27–31 August 2001). Published as *Computer Communication Review*, **31**(4):149–160, 2001.

[25] Ben Y. Zhao, John Kubiatowicz, and Anthony D. Joseph. *Tapestry: an infrastructure for fault-tolerant wide-area location and routing*. UCB Technical Report UCB/CSD–01–1141. Computer Science Division (EECS) University of California, Berkeley, April 2001.