Balancing Locality and Randomness in DHTs

Shuheng Zhou, Gregory R. Ganger, Peter Steenkiste

November 2003
CMU-CS-03-203

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

Embedding locations in DHT node IDs makes locality explicit and, thereby, enables engineering of the trade-off between careful placement and randomized load balancing. This paper discusses hierarchical, topology-exposed DHTs and their benefits for content locality, and administrative control and routing locality.

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, content locality, load balance

1 Introduction

Overlay networks based on distributed hash tables (DHTs) (e.g., CAN [12], Chord [17], Pastry [14], and Tapestry [18]) can provide a scalable, robust data location and request routing substrate for large-scale distributed applications, such as decentralized storage [2, 8, 15] and content distribution [1]. They map nodes and content items to the same numeric space, and route requests via this numeric space. Typically, node IDs are assigned randomly, and item keys are assigned to nodes nearby in the numeric space. The result is excellent load balancing, scalability, and robustness.

The randomness that yields these benefits, however, creates three difficulties. The first is *routing locality*: although hop counts can be reduced, stretch of an overlay path, defined as the ratio between the overlay lookup path cost and the shortest network distance between source and destination, can be quite high because the overlay topology is not congruent with the physical topology; each overlay hop may traverse the wide-area network. Locality-aware DHTs [11, 12, 14, 18] address this problem with increased protocol complexity and construction cost. A second difficulty is *content locality* [4]. DHTs intentionally map items to random locations (via random node ID assignment), causing most key insertions and lookups to traverse the wide-area network even with efficient routing—even for items with known access localities. Because wide-area communication is so much more expensive than local-area, balanced load cannot be the only performance concern. Third, randomized DHTs lack *administrative controls*, such as over possible content locations and who consumes what resources, that can be crucial to practical deployment of many applications.

This paper promotes an alternate approach, in which node IDs consist of two parts: a location-based prefix and a random remainder. Structuring the node ID space to embed topology information directly allows explicit regional control of content item placement. At the same time, randomizing item keys can retain the existing properties of DHTs. The result is a structured DHT that allows applications using it to decide where they each sit on a spectrum between locality and randomness.

This paper discusses ways of embedding hierarchies into node IDs and the resulting benefits for content locality, administrative control, and routing locality. Explicit setting of key prefixes allows content to be assigned to specific regions, as desired, yet be spread randomly within the region. Similarly, prefix-based routing allows low path stretch directly and, interestingly, can lend itself to Kleinrock and Kamoun's [7] hierarchical network clustering model for evaluation (rather than Plaxton et al.'s constrained growth model [11]).

2 Structured ID Space

Figure 1 illustrates an example location-based node ID comprised of a prefix with components hierarchically related to its containing regions and a suffix of randomly generated bits significant only within the leaf region. The boundaries between prefix components are not identical since the size of each component should be related to node density of the region it represents. One can construct a hierarchy with an arbitrary number of levels via prefix division and summarization.

Region	Sub−region	Leaf region	Random bits

Figure 1: Node ID with hierarchical prefixes

Existing DHT routing algorithms can be applied to a structured ID space. For example, prefix routing [11, 14, 18] sets up routing entries that share an increasing number of common prefixes with the current node. Chord-style [17] fingers connect to nodes in powers of 2 distances away. Given a key, DHTs route lookup messages toward the document root through nodes with IDs increasingly closer to the key.

Hierarchical Partitioning. Region and location can be geographical, topological, administrative or hybrid depending on the metrics one uses to partition the space. Different metrics and techniques can be used to partition the network at different levels of the hierarchy to satisfy an individual optimization goal; e.g., one can use geography to partition the network along continental boundaries first, then partition the AS peering graphs within each continent.

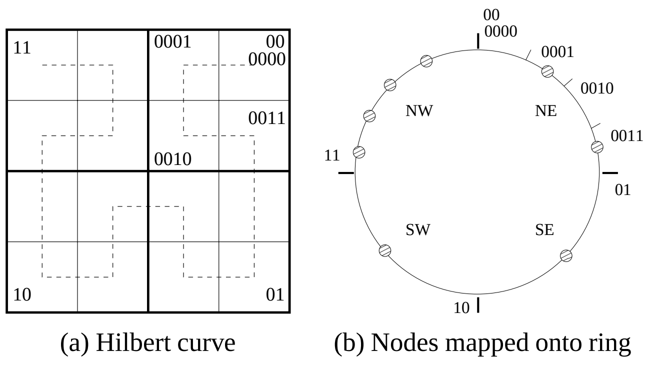


Figure 2: Quadtree-based division of space

For simplicity, we use geography to illustrate the partitioning and prefix encoding process. In Figure2, geographical regions at each level are divided into four sub-regions, and each is assigned a 2-bit prefix $(00, 01, 10, 11)$ based on their spatial order along a space filling curve (e.g., Hilbert curve); this repeats until leaf regions of a certain size are reached. The Hilbert curve passes through each region while reflecting their spatial locality in the high-dimensional physical space, which allows, e.g., a Chord-style [17] overlay path to make consistent physical progress when it makes numeric progress along the ring. For prefix routing, proximity relations across the numeric space can be discrete since each overlay hop always lands within a determined region toward the destination (see details in [19]).

Load Balance and Prefix Encoding. The prefix coding scheme above results in a skewed node distribution in the numeric space, since capacity varies across regions. A node is responsible for storing keys mapped onto the range between its previous identifier and its own. Assuming a uniform key distribution, a SW node in Figure 2 will assume higher load, since it owns a longer segment and thus more keys.

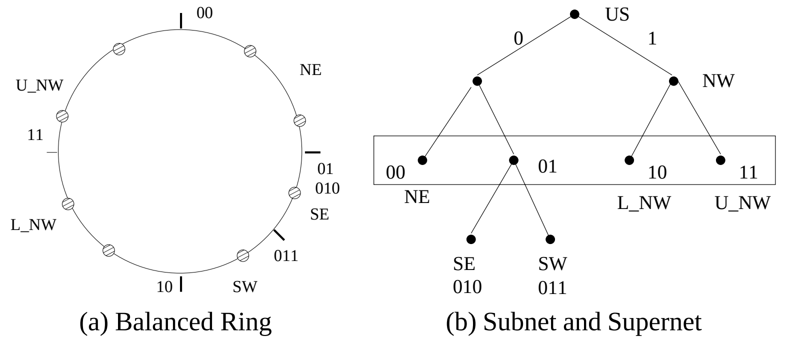


Figure 3: Load-balanced Prefixes

To achieve a uniform node distribution, one can allocate prefixes to regions with corresponding segment lengths proportional to their node density and capacity, as illustrated in Figure3(a), using prefix encoding algorithms [6, 9, 16]; such region sizing is explored in our technical report [19]. Region prefixes can be recursively divided and summarized at any location, following an IPv4 subnet and supernet analogy;

prefixes that can be summarized must represent sites adjacent in network topology so that a single route provides reachability to all nodes sharing that prefix. Figure 3 shows regions with 2-bit prefixes.

Decentralization. The partition of regions and prefix assignment at each level of the hierarchy might be a centralized algorithm, but the overall scheme can be decentralized since each region is responsible for its own sub-partition and prefix assignment. Only a small number of top-level regions need to obtain prefixes from a central authority, and a delegatable method similar to the allocation of network numbers can be adopted. More over, communication for node joins and leaves can be contained within regions [19].

3 Improving Content Locality

Our envisioned DHT infrastructure interprets the document key as a concatenation of two strings: a prefix that has a semantic (e.g., region) meaning and a suffix of random bits. Applications can exploit the semantics of key prefixes to control data placement. For example, embedding a content publisher's location prefix in keys for local weather updates will map keys to nodes (document roots) within the publisher's domain, while the random bits help spread keys (and weather data) across the leaf domain for load distribution. Alternatively, applications can ignore the prefix and choose a completely random document key to spread popular items across the entire Internet. This section discusses support for replication and consequences of explicit placement.

Replication. Applications use replication for various purposes, including availability, durability, performance, and load management [2, 3, 15]. To achieve effective replication, an application decides its policy and the system creates a unique replica root set, i.e., a set of replica keys, for each unique document key. For example, assume that a university has a volume of documents (e.g., technical reports) that they want to replicate both to speed up access and to improve availability, they could make arrangements with other organizations to replicate contents in their subnets. Replica keys can be generated by replacing high order bits of the unique document key by the location prefixes of peering organizations. A copy of the document can also be placed at the document root, i.e. the node corresponding to the unique document key.

A key challenge becomes how clients can find the prefix for an appropriate replica or a set of candidate replicas. One option is to use search engines (e.g., google) or other content discovery systems [3] to resolve high-level names such as attribute-value pairs to both a document key and its associated replica root set. Other approaches can yield the same associations. Once the client has a list of replica keys, it can use proximity in the ID space to identify the closest replica. For certain scenarios, the encoding of a replica root set can be very simple; see [19] for some examples. Each replica can support further replication, e.g., through caching, to address dynamic workloads and flash crowds.

The Good Kind of Load Imbalance. Traditional DHTs randomize node IDs, and thus key locations, which can balance load but tends to place most keys remote relative to clients accessing them. While we keep node distribution, especially those that handle cross-domain lookups, uniform, as discussed in Section 2, we allow key distribution to be skewed for the controlled data placement. For this discussion, we partition DHT communication in a region into four categories: internal (requests on local keys), inbound (external requests on local keys), outbound (internal requests to remote keys), and transit (external traffic routed through nodes in the region to nodes elsewhere in the DHT). Note that requests on keys include both insertions and lookups. Also note that local keys can correspond to both keys generated locally and mapped locally, or keys that are generated remotely but replicated locally. Similarly, remote keys refer to both keys generated remotely and mapped remotely, and keys generated locally but mapped to remote locations.

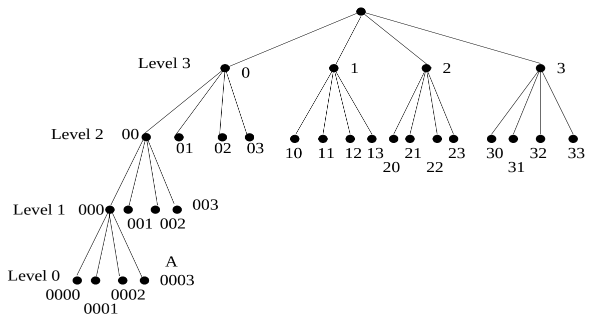
Our envisioned DHT offers informed applications the opportunity to assign keys so as to explicitly place items in specific regions. In particular, if items are placed in the same region as their clients (e.g., through replication), all associated communication will be internal. Although this may create imbalance, it is the good kind—internal traffic is usually much more efficient than non-internal traffic. Placing content near its clients significantly reduces the amount of outbound traffic and, for other regions, reduces the amount of transit and inbound traffic. The result is lower aggregate demand on network interconnection bottleneck points, such as intranet-to-Internet uplinks. Unless per-request server loads are large relative to wide-area communication costs, converting internal traffic to outbound (which still generates intra-region network traffic) would be strictly bad, even if the region's network is overloaded.

Of course, items for which the clients are not localized or not known should be assigned random keys, so as to spread them globally or across a certain region. Between the local placement and global replication, there exists a spectrum of tradeoff space for constrained load balance across regions of various scope. Applications that make bad placement decisions, or that encounter unexpected hot spots, should address them via the same replication and caching techniques used in existing DHT-based system designs. In the end, the transit traffic that traverses the wide-area networks will remain load-balanced (due to uniform node distribution), while most of the traffic is localized and, thus, internal.

4 Routing

By assuming Kleinrock and Kamoun's [7] hierarchical network clustering model, we are able to guarantee low path stretch at low protocol complexity and low routing table construction cost. This model is more intuitive than the bounded growth model [11] assumed in certain locality-aware DHTs [5, 11] to guarantee constant stretch. Furthermore, these DHTs require greater protocol complexity since nodes need to do probes into the networks for optimal neighbor selection.

In their classic paper [7], Kleinrock and Kamoun argued for the scaling properties of hierarchical routing: for a very large network, enormous routing table reduction may be achieved with essentially no increase in network path length. They cluster nodes recursively into a hierarchy where a level k cluster is defined recursively as a set of level $(k-1)$ clusters; this leads to a tree representation as shown in Figure4 (a), where internal tree nodes represent clusters. They assume that all clusters at the same level are of equal degree and that a path exists between any pair of nodes. The diameter of any k^{th} level cluster is upper bounded by a quantity that decreases as k decreases. Finally, any cluster contains a shortest path between any pair of nodes within that cluster.



(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

Prefix routing [11, 14, 18] uses a similar idea for routing table reduction (from $O(N)$ to $O(\log N)$) using a numeric hierarchy that forms a tree as in Figure 4. In a structured-ID space, the underlying numeric hierarchy that clusters nodes based on common prefixes satisfy the constraint on hierarchy defined in

Kleinrock: clusters at the same level have the same degree and their diameters are bounded to a quantity that decreases as one goes down the tree, following the construction as in Section2. The rest of Kleinrock's assumptions can also be made with prefix routing: a logical connection between any pair of nodes exists and the overlay path stays within a region once it reaches there. Hence, under the assumptions of [7], path lengths are bounded within a constant factor of shortest physical paths without requiring nodes individually probe the network to find a nearest neighbor. For Chord-style routing, the relative low stretch is achieved via space filling curves (in a geometric space) as in Figure 2(a).

Finally, in current DHTs, every node v can be the exchange node of its clusters at different levels, since that responsibility is randomized over the entire set of nodes in each cluster; *exchange* here means that v appears in the routing table of some other nodes for reaching v's cluster at level k. See [19] for discussions of benefits and heuristics for reducing the entire set of nodes in a cluster to a robust subset as exchange nodes for that cluster.

5 Evaluation

To verify the routing locality property, we run simulations using real Internet data to compare stretch in a geometric ID space against a random ID space. The smaller set contains 103 nodes across the US from NLANR's AMP project (http://watt.nlanr.net/); the large dataset contains 869 nodes with IP randomly chosen from the entire address space, located across five continents but concentrated in North America and Europe. Through GNP [10], each node is assigned a coordinate from a N-dimensional Euclidean space, where geometric distances between nodes approximate their network distances. We partition the geometric space recursively and generate a Hilbert curve to align regions along a numeric ring, with segment size proportional to the node density for each region; a random ID is assigned to a node from its region. This results in a uniform node distribution. Table 1 shows the latency statistics and number of Euclidean dimensions used to approximate the two data sets.

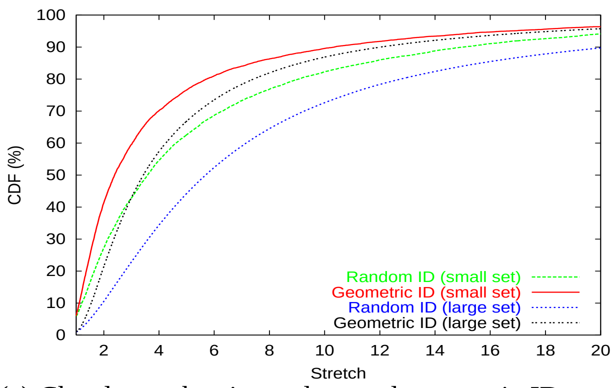
Table 1: Data set

Scheme	nodes	min (ms)	max (ms)	ave (ms)	N
NLANR	103	1.5	336	69.6	5
Random IP	869	0.15	1745	181	7

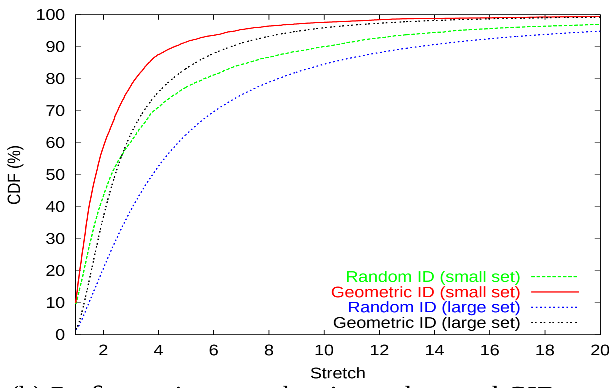
Table 2: Stretch results: (1) for NLANR and (2) for Random IP set

Scheme	Avg.	Median	10%	90%
Chord in GID (1)	5.11	2.40	1.12	10.34
Chrod in RID (1)	6.88	3.60	1.21	14.98
Chord in GID space (2)	6.25	3.44	1.56	12.02
Chord in RID space (2)	10.16	5.70	1.97	20.32
Prefix Routing in GID (1)	2.62	1.74	1.01	4.56
Prefix Routing in RID (1)	4.70	2.30	1.03	9.93
Prefix Routing in GID (2)	3.53	2.44	1.32	6.60
Prefix Routing in RID (2)	6.76	3.79	1.49	13.35

Figure 5 shows the CDF comparison of stretch between Chord-style routing and prefix routing on a random ID (RID) space and a geometric ID space (GID). Table 2 summarizes the stretch statistics. For both datasets and both types of DHT routings, stretch is reduced significantly in the geometric ID space.

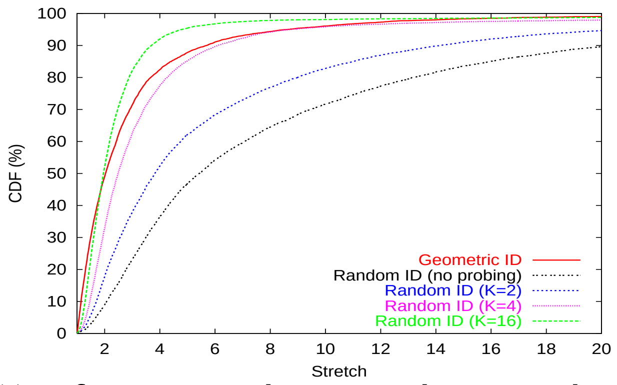


(a) Chord stretches in random and geometric ID space

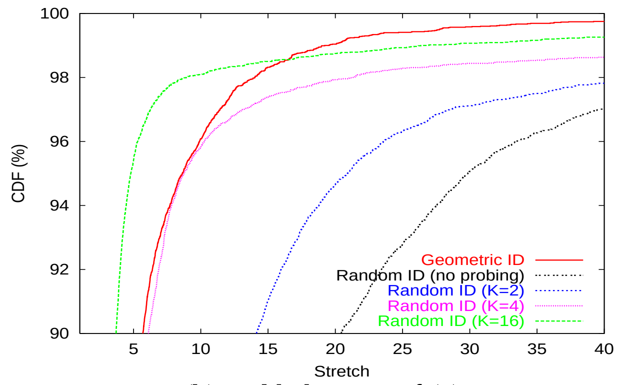


(b) Prefix routing stretches in random and GID space

Figure 5: We run a static (no node joins or leaves) simulation and compute stretch based on Euclidean distances. In (a), we compare the stretch of Chord-style routing on random IDs vs. on a geometric location-based ID space. CDF is over all the pair-wise delays. In (b), we did the same comparison for prefix routing that is not locality-aware.



(a) Prefix routing stretches using probes vs. GID-based



(b) Tail behaviors of (a)

Figure 6: For locality-aware prefix routing, we allow each node to probe K nodes from a prefix range randomly when selecting a neighbor. For GID-based prefix routing, routing table entries are randomly selected from the prefix range. We compute the stretch for 10000 randomly selected source-destination pairs.

To compare with locality-aware prefix routing algorithms in a random ID space (RP-based), we constructed a larger data set by treating each node in NLANR as a representative node and added 100 nodes with small distances to this node to simulate a two-tier hierarchy: under each university, a set of 100 nodes as its subnet is added. As such, we have a network of 10403 nodes. Figure 6 compares prefix routing on a geometric ID space (GID-based), where a neighbor is randomly selected from its prefix range, and RP-based prefix routing, where for each prefix in the routing table, a random set of K nodes from the prefix range are probed to measure their relative distances (latencies) to current node and the closest one is chosen. The CDF results show that, with more probes, RP-based prefix routing approaches GID-based; When 16 probes are used, the average stretches are comparable in two schemes, but RP-based has much longer tail, which indicates the difficulty at optimizing its last hops.

6 Related Work

Rather than having each node probe the network for proximity information independently [11, 12, 14, 18], we abstract distance to build a structured ID space. We achieve low stretch in overlay paths at low complexity. CAN [12, 13] explored topology-aware ID assignment by clustering nodes that are physically close to the same virtual coordinate space. CAN focused more on reducing routing latency than building a structured ID space that enables content locality. They did not address the load balancing issue caused by skewed node distribution in the physical networks. Current replica placement algorithms [2, 15] rely on randomness to spread replicas across geographical areas; explicit naming via encoding of a replica root set provides much more control. Finally, SkipNet [4] uses string names to cluster nodes and control data placement within certain administrative domains. Our content locality scheme is more flexible and applies to different metric spaces such as geometric and topological. In addition, no bound on path stretch can be guaranteed in SkipNet.

7 Summary

This paper promotes a new approach to node ID assignment, with prefixes based on location within DHTs. Doing so simplifies routing locality and enables explicit content locality, while retaining existing DHT properties for non-local documents.

Acknowledgements

The authors thank Bruce Maggs and Jun Gao for valuable discussions. We also thank An-Cheng Huang and Eugene Ng for providing the GNP data sets.

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] Kirsten Hildrum, John Kubiatowicz, Satish Rao, and Ben Y. Zhao. Distributed Object Location in a Dynamic Network. *SPAA - ACM Symposium on Parallel Algorithms and Architectures* (Winnipeg, Canada, August 2002). ACM, 2002.

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

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

[8] 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.

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

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

[11] 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.

[12] 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.

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

[14] 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.

[15] 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.

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

[17] 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.

[18] 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.

[19] Shuheng Zhou, Gregory R. Ganger, and Peter Steenkiste. *Location-based Node IDs: Enabling Explicit Locality in DHTs*. Technical report CMU-CS-03-171. Carnegie Mellon University, September 2003.