Frequency-aware Reconstruction of Forwarding Tables in Name-based Routing

Haesung Hwang[†]
h-hwang@ist.osaka-u.ac.jp

Shingo Ata[‡]
ata@info.eng.osaka-cu.ac.jp

Masayuki Murata[†]
murata@ist.osaka-u.ac.jp

[†]Graduate School of Information Science and Technology, Osaka University

[‡]Graduate School of Engineering, Osaka City University

ABSTRACT

In the future Internet, routing on contents or resources is anticipated as the post-Internet Protocol (IP) routing. The final goal of our research is to realize routing based on the name of a resource in the network layer. Towards this purpose, we use the 'name' for routing, particularly the 'fully qualified domain name (FQDN)' to show the feasibility of name-based routing and to generalize it to resource-based routing. When writing the routing information of hierarchically structured FQDN into a hierarchical virtual topology, mismatch between the virtual topology and physical topology can occur if the virtual topology lacks the physical topology's information. In addition, routing tables should be reorganized to reflect the different access frequency among the FQDNs. In this paper, we propose an algorithm for reconstructing routing tables to better map a virtual topology to the physical topology. As a result, we show that using the access frequency and physical topology's information increases the efficiency for searching FQDN.

1. INTRODUCTION

Network architectures such as the ones that perform routing using *information* instead of conventional internet protocol (IP) addresses are anticipated for the future Internet [4]. However, problems such as reserving memory capacity at routers, minimizing topology changes in the network, and devising deployment strategies arise in order to realize routing in the network layer based on resources.

As a first step in showing the feasibility of resource-based routing, we investigate routing on the basis of *names*, which is a typical type of resource. Currently, there are a number of standardized names to represent resource information. Examples are the common language equipment identifier (CLEI) [20], the extensible resource identifier (XRI) [16] , the life science identifiers (LSID) [13], and the digital object identifier (DOI) [6]. A CLEI identifies a communication device and is globally unique. The ID's ten alphanumeric characters represent the device's technology, type, function, and manufacturer, as well as provide complementary data. The XRI is independent of domain, location, and application (syntax (1) in Figure 1). The LSID identifies biologically significant resources using syntax (2) in Figure 1, and the DOI identifies content objects in a digital environment using syntax (3) in Figure 1 where prefix and suffix identify the registrant and single object, respectively. These uniform resource identifiers (URIs) [2] are commonly used to identify abstract and physical resources and are compatible with syntax (4) in Figure 1. Most of the naming schemes have a hierarchical structure, i.e., a tree structure containing a number of domains each of which having names and/or lower-level domains. Particularly, the 'authority' part in the names in Figure 1 show the hierarchical structure, ideally suited to a more distributed registration scheme [12].

```
(1) xri://authority/path?query#fragment
(2) urn:lsid:authority.org:namespace:object:revision
(3) prefix/suffix
(4) scheme name:hierarchical part?query#fragment
```

Figure 1: Syntax of standardized names used to represent resource information

In addition, the hierarchical structure of the naming scheme has an affinity for storing themselves in the forwarding table of routers deployed in a network. Routers in a network also have hierarchical structure: backbone, local gateway, and edge. Information regarding the top-level domain name is stored in backbone routers, second-level domain name is stored in local gateway routers, and third-level domain name is stored in edge routers. We show the case of mapping fully qualified domain names (FQDNs), which is a typical example of hierarchical naming structure, into routers in Figure 2 [1]. To store hierarchical names, a virtual topology is constructed hierarchically shown as the left part of Figure 2. The top level ring topology is constructed for the top level domains, in which each node has a pointer to the ring for the second level domains. Also a node of the second level ring has a pointer to the ring for third level domains. On the other hand, the current Internet topology is also hierarchical as shown in the right part of Figure 2. From the similarity of natures of both topologies, it is easy to map the virtual topology to the physical topology, i.e., a virtual topology is constructed by ring-shaped connections between routers at the same level. Ring-shaped connections are now common in overlay networks such as distributed hash tables (DHT) [17, 19]. However, one of the problems with overlay networks is a potential mismatch between the logical and physical topologies [8, 14]. A mismatch can result in routing delays because the routing may be from a node to an apparent neighbor node based only on their IDs, when in actuality they are physically far apart. Network architectures using

CFI '10 June 16–18, 2010, Seoul, Korea

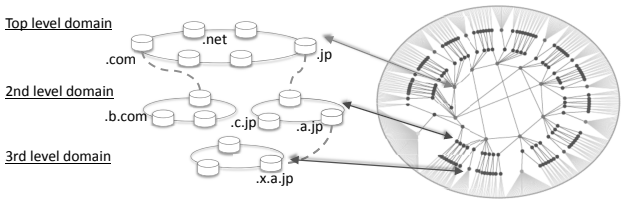


Figure 2: Virtual topology for storing hierarchical names and physical network topology

logical topologies should reflect the physical topology information to minimize delays due to such mismatches.

In our approach, on the other hand, name-based routing is performed *within* the routers, i.e., in the network layer. Routers have the forwarding table for *names*, and packets have *names* for their source and destination addresses. Packets are forwarded according to the name instead of IP addresses. This eliminates the need for servers to resolve resource names to IP addresses. Moreover, the end nodes, which generally have less processing capabilities and reliability than routers, no longer need to forward packets, especially in the overlay network environment. As a result, the overall network reliability increases because the routers, not the end nodes, search for resources and information. However, the mismatch between the location of routing information in the virtual topology and the actual source of the query still remains. In other words, the routing information that the source node requires is not always stored near the source nodes.

In this paper, we develop an algorithm for location-aware virtual topologies that takes into account the physical topology and reconstructs the routing tables so that the frequency of access to objects is considered. Simulation show that using such reconstructed tables reduces the number of hops by about 20%.

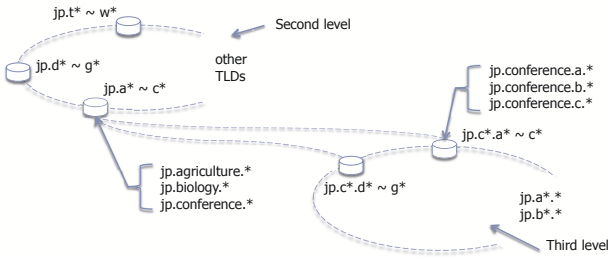
The rest of this paper is organized as follows. Section 2 discusses previous work on location- and access-frequency-aware logical topologies. Section 3 describes the algorithm we developed for reconstructing the routing tables. The simulation settings and evaluation results are presented in Section 4. Section 5 concludes the paper with a summary of the main points and an outlook on a description of future work.

2. RELATED WORK

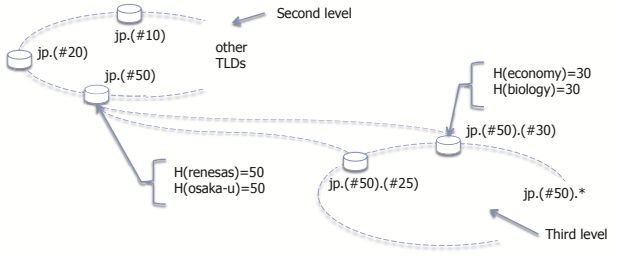
In this section we summarize previous work on location-aware logical topologies and consider how the node load is affected by the frequency of access to objects.

One of the popular methods used in previous work to solve the mismatch between the logical overlay network and physical underlying network is utilizing routing redundancy, where nodes have multiple next-node candidates when forwarding queries. In the algorithm described by Ratnasamy et al. [15], each node forwards packets to the node with the shortest round-trip-time (RTT) and in the algorithm proposed by Datta et al. [5], the next hop node with the lowest load is selected. Landmark clustering is utilized by the algorithm of Xu et al. [21], where neighboring nodes are located close to a certain landmark node. However, these previous efforts are suboptimal solutions in the application layer and what to do with the data itself still remains an open problem.

Caching and replication of the data key is used to solve the hot-spot problem, which occurs when certain nodes have a high load due to having popular objects. Serbu et al. [18] suggested using the load of each node to reconstruct the routing tables. The forwarding



(a) Hierarchical Longest Alphabet Match (HLAM)



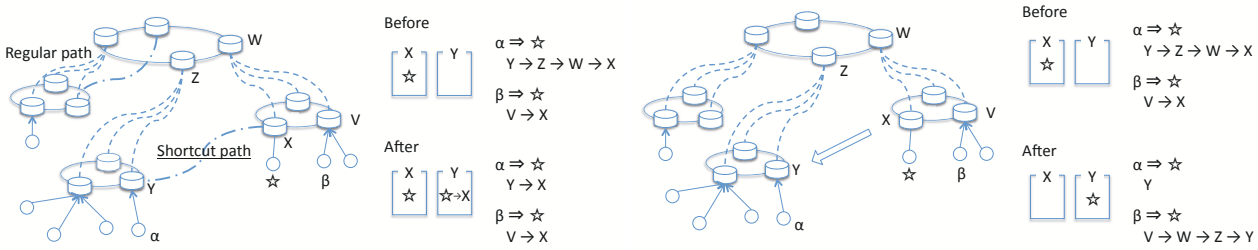
(b) Hybrid Distribution (HD)

Figure 3: Examples of FQDN entry distribution using HLAM and HD algorithms

load of a node A that is receiving many requests is reduced by deleting the path to A from the previous node B and adding another entry with the same prefix as node A to the routing table in B. However, there is additional overhead of keeping track of the load on each other node and maintaining this information in its routing table.

In this paper, we propose an algorithm that reconstructs the routing tables considering the frequency of access to each entry. Unlike the algorithm in [18], maintaining the status of each node's load is unnecessary, therefore reducing the storage overhead.

Before mentioning this paper's contribution and algorithm in the following sections, we briefly summarize our previous work. We demonstrated in [9] the feasibility of storing the routing information in FQDNs in ternary content addressable memory (TCAM), which is the high-speed memory used in modern routers. The required number of routers for the storage is less than two orders of magnitude compared to the number of currently deployed routers [9]. The algorithms used to distribute the routing tables which contain domain names to multiple routers were *hierarchical longest alphabet match (HLAM)* and *hybrid distribution (HD)*, which are illustrated in Figure 3. The HLAM algorithm was inspired by the longest prefix match [7] algorithm and expresses domain names using American standard code for information interchange (ASCII). Names that start with the same letter, i.e., that have bits in common, can be stored in the same router. For example, since the ASCII for `a`, `b`, and `c` of domain names `jp.a*`, `jp.b*`, and `jp.c*` are `1100001`, `1100010`, and `1100011`, respectively, they can all be expressed as `11000**` using the don't care bit '`*`' of TCAM. The major advantage of HLAM is using this don't care bit, which can be either 0 or 1, makes high-speed search possible in TCAM. The HD algorithm first groups FQDNs by their top-level domain and hashes the second level domain and below. Domain names that have the same hash value are stored in the same router. For example, if the hash values of `abc` (part of `abc.co.jp`) and `xyz` (part of `xyz.ac.jp`) are the same, the FQDNs are stored in the same router. The use of this hash function means that FQDNs are



(a) Shortcut path

(b) Entry migration

Figure 4: Scenarios 1 and 2

distributed more equally than with the HLAM algorithm, so fewer routers are needed. We also evaluated the feasibility of robustly updating the routing information database entries and showed that there are no drastic changes in the routers' forwarding tables [10].

3. RECONSTRUCTION OF ROUTING TABLES

In this section, we discuss scenarios for quickly responding to queries regarding routing information for highly accessed FQDNs. The routing table's *initial state* represents the state of the routing tables immediately after database distribution using the HLAM or HD algorithm and before routing table reconstruction. Here, *request frequency* means the degree of requests from the source and *access frequency* means the degree of accesses that the destination receives. The gateway router of the source or the routers along the path to the object receive 'requests,' and the router with the object receives 'accesses.'

3.1 Scenario 1: Shortcut Path

One scenario for quickly answering queries regarding routing information for highly accessed FQDNs is to add entries to the routing table. These entries are shortcuts to routers with high access frequencies (i.e., the routers that have routing information for the destination FQDN) from a router that has a high request frequency. As shown in Figure 4(a), router Y adds an entry to reach router X directly depending on the number of requests to the node/resource '$\star$'. After the shortcut is established, other routers in the same ring as router Y can use it to effectively reach router X. Therefore, the other nodes in the ring are notified about Y $\rightarrow$ X.

Adding entries for shortcuts makes it unnecessary to traverse the upper layers in the hierarchy. In addition, the routers can receive information about network changes as soft-state updates and reselect neighboring nodes, which increases robustness [21]. However, it is not always possible to create shortcut paths through different Internet service providers.

3.2 Scenario 2: Entry Migration

Another scenario for quickly answering queries regarding routing information for highly accessed FQDNs is to delete the routing information for entries that have a high access frequency from the initial state router and to add entries that have a high request frequency to the router. As shown in Figure 4(b), with this 'migrating entry' scenario, the routing information for node/resource '$\star$' is moved from router X to router Y when the number of requests exceeds a threshold. If it is moved, notification is sent to the routers in the same ring as X, to upper layer router W, to router Y's upper layer router (Z), and to the routers in the same ring as Y.

Migrating an entry from a router with a high access frequency to one with a high request frequency enables the routing information for the entry to be deleted from the initial router. This results in better utilization of the router memories. In addition, routing information is stored in a router with a high request frequency or in one close to where there are many requests, which speeds up query response. However, overhead is increased because other routers have to be notified of the change.

3.3 Scenario 3: Combination

The third scenario for quickly answering queries regarding routing information for highly accessed FQDNs is a combination of scenarios 1 and 2. The algorithm uses two parameters.

1. $P_a(i)$ is the number of accesses to an item P. It is the sum of the number of accesses from the router i that has the routing information for P and from the routers that have the same TLD as i.
2. $P_r(j)$ is the number of requests to an item P. It is the sum of the number of requests from the router j that is on the path from the source to P and from the routers that have the same TLD of j.

The algorithm has three steps.

1. If $P_r(j) > P_a(i)$, add a shortcut path from j to i
2. If $P_r(j) > P_a(i)$ over a time period t, move item P to j
3. Repeat (1) and (2) depending on the request/access frequency

4. EVALUATION

We evaluate the static placement of routing tables and the adaptive placement (reconstruction) of routing tables in accordance with the access frequency. We first distribute the database using the HLAM and HD algorithms, which are also referred to here as the 'name-based' and 'hash-based' algorithms. We use the FQDN database for January 2010 from the Internet Systems Consortium (ISC) [11], which has approximately 732 million entries. To shorten the simulation time, only 1% of the entries (7.3 million) are used. The size of the TCAM in a router is also reduced, to 1.8 Mbit $\times$ 1 TCAM = 1.8 Mbit compared to 18 Mbit $\times$ 10 TCAMs = 18 Mbit used in our previous work.

4.1 Simulation Settings

We compare static placement, which does not reconstruct the routing tables, with adaptive placement, which reconstructs the routing tables in accordance with the request/access frequency of the destination FQDN. The evaluation metric is the average number of hops between the source and the destination. Since the average

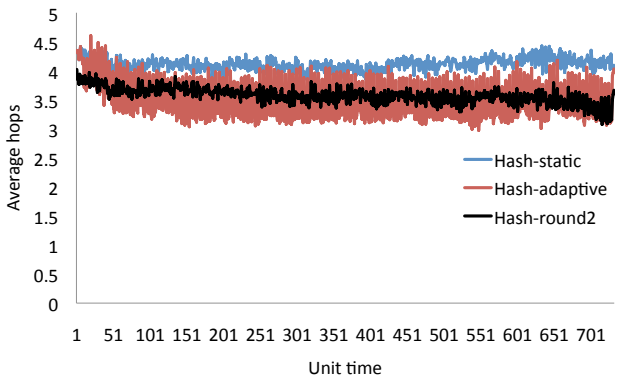


Figure 5: Average number of hops (hash-based)

number of hops in a general DHT ring with n nodes is $O(\log n)$, we estimate it as $\log n$. The number of hops between a router in the lower level and one in the upper level is set to 1, on the basis of the single-connection intra-group structure described by Zoels et al. [22]. This is considered to be a desirable structure in a hierarchical DHT system, in which multiple peers are connected to a super peer. Furthermore, it is assumed that the nodes in each lower level make a ring of their own.

There are two rules for selecting the source/destination pairs.

1. Determine the destination first. The destination has a Zipfian distribution with the index factor 1 [3]. In other words, of N = 732,740 pairs, the 2nd most popular destination has half the number of accesses as the most popular destination. In addition, it is assumed that general (g) TLDs are accessed more often than country-code (cc) TLDs. The top three destinations are selected from the pool of gTLDs.

2. The rules for selecting the source depend on whether the destination is a gTLD or a ccTLD. To bias the origin of the source, three popular sources are selected to have a distribution of 40%, 20%, and 10% when the destination is a gTLD. The remaining 30% are selected randomly. When the destination is a ccTLD, 90% of the sources are selected to have the same TLD to impart a locality characteristic.

The number of hops is calculated for static and adaptive placement using the selected pairs. The condition $P_r(j) > P_a(i)$ is checked after 5 time unit have passed. Here, 1,000 communications (i.e., source requests for destination FQDN) occur in each unit time. Therefore, if $P_r(j) > P_a(i)$ holds after 5,000 communications, a shortcut path is established. Time t is when $P_r(j) > P_a(i)$ holds even after the shortcut path is established. In other words, if there is an attempt to create a shortcut when one already exists, the source and destination pair is considered to be very popular, resulting in entry migration.

4.2 Simulation Results

Figure 5 shows the average number of 1,000 hops in each unit time for hash-based static, adaptive, and adaptive-round 2. 'Adaptive round 2' uses the content of the forwarding tables from the 'adaptive' and the same source-destination pairs are used in the evaluation. An average of approximately 4.1 hops per unit time was maintained with static placement while the average dropped to about 3.4 hops after 730 time units with adaptive placement. This may seem as a small amount of reduction but shows a potential in developing a mapping algorithm to reach the destination with the shortest path possible.

Figure 6 shows the number of requests for the three different path types with the hash-based algorithm. 'Regular path' means the source reached the destination using the path given in the initial state. 'Shortcut path' and 'Moved path' are self-explanatory. When the number of communications using the shortcut path increased, the number using the 'regular path' decreased. In addition, more source and destination pairs used the 'shortcut path' initially, but as these popular destinations migrated, more pairs used the 'moved path'. In Figure 6(b), the number of requests using each path are stable, maintaining the values of Figure 6(a). The results shown in this paper are only example cases.

The result of name-based is not shown due to the small difference in the numbers which is thought to be caused by the relatively small database used in the simulation. Since the name-based algorithm distributes the database in accordance with the name's ASCII, it creates a less balanced distribution of the database, so more routers are needed for storage than with the hash-based algorithm.

5. CONCLUSION

We have investigated the feasibility of using name-based routing, particularly routing using FQDNs, as a first step towards resource-based routing. We investigated the mapping of the logical topology to the physical topology and described an algorithm for reconstructing routing tables on the basis of the differences in access frequency between objects. Simulation of the proposed algorithm using an FQDN database showed that using such reconstructed tables reduces the number of hops by approximately 20%, demonstrating that this approach is feasible. Future work includes developing an algorithm to handle end nodes with mobility and generalizing name-based routing to resource-based routing.

Acknowledgment

This research was supported by the National Institute of Information and Communications Technology (NICT) of Japan.

6. REFERENCES

[1] Abilene Network, `http://www.internet2.edu/network/`.

[2] T. Berners-Lee, R. Fielding, and L. Masinter. RFC 3986: Uniform Resource Identifier (URI), Jan 2005.

[3] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web Caching and Zipf-like Distributions: Evidence and Implications. In *Proc. of IEEE INFOCOM*, volume 1, pages 126–134, Mar 1999.

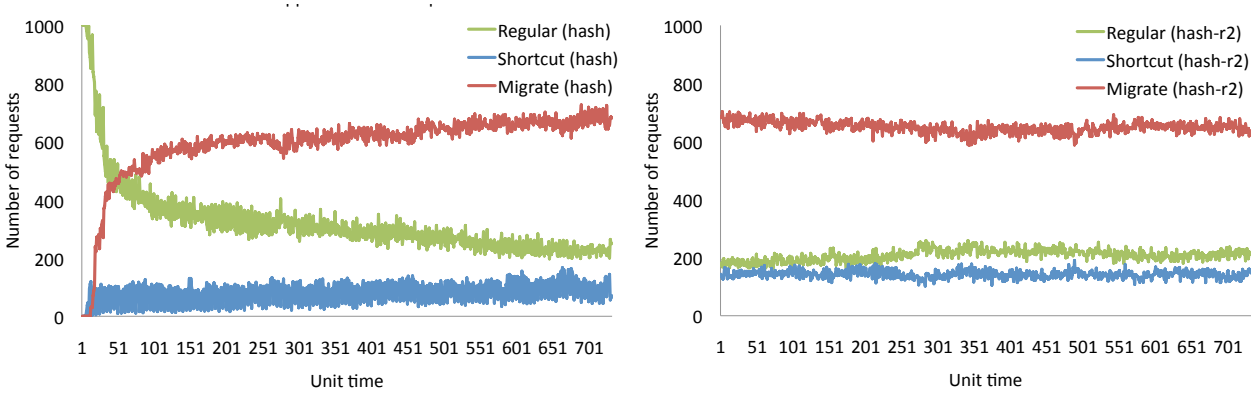
[4] A. Carzaniga, M. J. Rutherford, and A. L. Wolf. A Routing Scheme for Content-Based Networking. In *Proc. of IEEE INFOCOM*, Mar 2004.

[5] A. Datta, R. Schmidt, and K. Aberer. Query-load Balancing in Structured Overlays. In *Proc. of the 7th IEEE CCGRID*, pages 453–460, May 2007.

[6] Digital Object Identifier (DOI), `http://www.doi.org/`.

[7] W. Doeringer, G. Karjoth, and M. Nassehi. Routing on longest-matching prefixes. *IEEE/ACM Transactions on Networking*, 4:86–97, Feb 1996.

[8] B. Hariri, S. Shirmohammadi, and M. R. Pakravan. LOADER: A Location-Aware Distributed Virtual Environment Architecture. In *Proc. of the IEEE VECIMS*, pages 97–101, July 2008.



(a) Round 1

(b) Round 2

Figure 6: Number of requests via regular, shortcut, and migrated paths

[9] H. Hwang, S. Ata, and M. Murata. A Feasibility Evaluation on Name-based Routing. In *Proc.of IEEE IPOM*, pages 130–142, Oct 2009.

[10] H. Hwang, S. Ata, and M. Murata. The Impact of FQDN Database Updates on Name-based Routing Architecture. In *to be presented at 5th IFIP/IEEE BcN*, Apr 2010.

[11] Internet Systems Consortium (ISC), `http://www.isc.org/index.pl?/ops/ds/`.

[12] C. Kozierok. *The TCP/IP Guide: A Comprehensive, Illustrated Internet Protocols Reference*. No Starch Press, 2005.

[13] Life Sciences Identifiers (LSID), `http://lsids.sourceforge.net/`.

[14] T. Qiu, G. Chen, M. Ye, E. Chan, and B. y. Zhao. Towards Location-aware Topology in both Unstructured and Structured P2P Systems. In *Proc. of International Conference on Parallel Processing*, pages 30–37, Sep 2007.

[15] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A Scalable Content-Addressable Network. In *Proc. of the SIGCOMM*, pages 161–172, Aug 2001.

[16] D. Reed and D. McAlpin. Extensible Resource Identifier (XRI) Syntax V2.0. `http://www.oasis-open.org/committees/download.php/15376#_Toc117301832`.

[17] A. Rowstron and P. Druschel. Pastry: Scalable, Distributed Object Location and Routing for Large-scale Peer-to-peer Systems. *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, 11:329–350, Nov 2001.

[18] S. Serbu, S. Bianchi, P. Kropf, and P. Felber. Dynamic Load Sharing in Peer-to-Peer Systems: When Some Peers Are More Equal than Others. *IEEE Internet Computing*, 11(4):53–61, July 2007.

[19] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. *Proceedings of the SIGCOMM*, 31(4):149–160, Aug 2001.

[20] K. Tesink and R. Fox. RFC 4152: A Uniform Resource Name (URN), August 2005.

[21] Z. Xu, C. Tang, and Z. Zhang. Building Topology-aware Overlays using Global Soft-state. In *Proceedings of International Conference on Distributed Computing Systems*, volume 23, pages 500–508, May 2003.

[22] S. Zoels, Z. Despotovic, and W. Kellerer. On Hierarchical DHT Systems–An Analytical Approach for Optimal Designs. *Computer Communications*, 31(3):576–590, 2008.