MDHT: A Hierarchical Name Resolution Service for Information-centric Networks

Matteo D'Ambrosio
Telecom Italia
Torino
Italy
matteo.dambrosio@telecomitalia.it

Christian Dannewitz
University of Paderborn
Paderborn
Germany
cdannewitz@upb.de

Holger Karl
University of Paderborn
Paderborn
Germany
hkarl@upb.de

Vinicio Vercellone
Telecom Italia
Torino
Italy
vinicio.vercellone@telecomitalia.it

ABSTRACT

Information-centric network architectures are an increasingly important approach for future Internet architectures. Several approaches are based on a non-hierarchical identifier (ID) namespace that requires some kind of global Name Resolution Service (NRS) to translate the object IDs into network addresses. Building a world-wide NRS for such a namespace with 10^{15} expected IDs is challenging because of requirements such as low latency, efficient network utilization, and anycast routing. In this paper, we present an NRS called Multi-level Distributed Hash Table (MDHT). It provides name-based anycast routing, can support constant hop resolution, and fulfills the afore mentioned requirements. A scalability assessment shows that our system can scale to the Internet level, managing 10^{15} objects with today's storage technology and 1/10th of today's DNS nodes. The evaluation indicates that a non-hierarchical namespace can be adopted on a global scale, opening up several design alternatives for information-centric network architectures.

Categories and Subject Descriptors

C.2 [**Computer Systems Organization**]: Computer-Communication Networks

General Terms

Design

Keywords

Information-centric, name resolution, name-based routing, flat namespace, hierarchical DHT

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
ICN'11, August 19, 2011, Toronto, Ontario, Canada.
Copyright 2011 ACM 978-1-4503-0801-4/11/08 ...$10.00.

1. INTRODUCTION

Information-Centric Networking (ICN) is an important networking paradigm that has the potential to solve several problems of today's Internet architecture, including inefficient resource utilization, problems caused by flash crowds, Distributed Denial of Service (DDoS) attacks, and inadequate security. Several ICN approaches have been proposed for the Future Internet, e.g., the Data-Oriented Network Architecture (DONA) [10], Content-Centric Networking (CCN) [9], and the Network of Information (NetInf) [1]. ICN approaches put the information at the center of the architecture and shift the communication model from the node-centric paradigm that focuses on conversations to the information-centric paradigm that focuses on information dissemination.

Efficient information dissemination should make use of any available data source. To do so, most ICN architectures are in some way based on an identifier/locator split. Information is identified with a location-independent identifier (ID) that cannot be used for forwarding/routing purposes with current schemes like IP. As a result, ICN architectures have to solve the problem how to retrieve data based on these location-independent IDs. In general, there are two major solutions to this problem. First, performing *name-based routing* on these IDs. Second, performing a *name resolution* step first that resolves the information ID into a locator, which can be used by some other forwarding scheme (e.g. IP) to retrieve the information. In this paper, we focus on a Name Resolution Service (NRS) for ICN architectures.

NRS systems like the Domain Name System (DNS) or common Distributed Hash Table (DHT) systems do not fulfill our requirements (Section 2) as discussed in Section 3. We propose a distributed NRS with integrated name-based routing functionality called *Multi-level Distributed Hash Table (MDHT)* (Section 4). The MDHT system provides a nested, hierarchical DHT architecture for scalable distributed name resolution and efficient anycast routing of flat IDs. If the IDs include some additional structure like owner/publisher information, the MDHT system can use this structure to perform name aggregation on the global level to simplify global deployment and further improve scalability. The hierarchical architecture can reflect the existing network topology, topologically embedding the NRS for efficient data dissemination

in the Internet. The MDHT system is applicable to any system that requires a distributed dictionary generally or an NRS specifically, like several ICN approaches. In the following, we use NetInf as a concrete example as our work is based on NetInf-related work [4]; however, MDHT is not restricted to NetInf.

2. REQUIREMENTS

An ICN NRS has to handle global data requests based on non-hierarchical IDs efficiently and with low latency. It might return either a (list of) locator(s) for the requested object or the requested data object itself.

The current number of world-wide "data pieces" including sensor data, etc., is approximately on the order of 10^{16} [7], while the number of indexed Web pages is on the order of 10^{10} in the main search engines[1]. We expect that a significant fraction of the digital universe might be registered in an information-centric network. Hence, the system has to be extremely scalable to support a large number of data objects as well as up to 10^9 users. Moreover, the NRS has to support frequently created/disappearing copies of those data objects. The *scope* of such copies, i.e., where a copy is made available, should be controllable. For example, a user or organization might want to make a data copy known and accessible only within the own local network but not outside this network.

To support efficient data dissemination, an ICN NRS should be able to make use of any available data copy. This reduces traffic in the network by selecting the most suitable (set of) copy(s) available. If a requested data copy is available in a site *close* to the requester with respect to some network metric, the NRS should be able to propose it for retrieval (we call this property *content locality*). In that case, if the two endpoints are in the same network domain, the resolution path and the data path should also be contained in that domain (*local resolution* and *forwarding locality*). As a consequence, the inter-domain traffic would be minimized.

For a real-world system, deployability is also important. Deployment is simplified if the system can be deployed "from the edges", i.e., parts of the NRS can be deployed and used without immediately requiring a full deployment. A system that fulfills the requirements as discussed above, especially a system that reduces inter-provider traffic, offers important incentives for providers to deploy such an NRS in their networks. Giving providers *administrative autonomy* of their NRS subsystem further simplifies deployment.

3. RELATED WORK

Several ICN architectures like NetInf use a non-hierarchical namespace [5] to achieve, among others, name persistence. The need for persistent names is also emphasized by the spreading of persistent naming schemes like the Digital Object Identifier (DOI). However, with a non-hierarchical namespace, we cannot use approaches that require hierarchical names for aggregation, like CCN in routing tables or DNS for name resolution. Like DNS, our system uses a hierarchical architecture. However, we do not use hierarchy to perform aggregation based on hierarchical names as done in DNS. Instead, MDHT uses the hierarchy to minimize inter-domain traffic (request *and* data traffic), to reduce latency, and to benefit from locality in request patterns as discussed later on.

Structured DHT systems like CHORD [15] and Kademlia [11] represent a scalable solution for handling flat namespaces. Approaches like CoDoNS [14] use a DHT to build an NRS. However, those *flat* DHT systems do not support efficient data dissemination. They are not topologically embedded, making the locality requirements hard to fulfill. Constant hop DHTs like Structured Superpeers [13] have the same problem but could be used as subsystems in MDHT to reduce latency.

Hierarchical DHTs [2] can fulfill those requirements. Approaches like Cyclone [3], the Generic Hierarchical DHT Framework [8], Hierarchical Rings [12], as well as Canon [6] provide topological embedding. However, none of the hierarchical DHT systems that we are aware of fulfills the combination of our requirements, especially when considering deployability.

4. Multi-level Distributed Hash Table

The following Section 4.1 describes the general MDHT system. Section 4.2 then discusses global name resolution and introduces an optimization for structured names, the Resolution Exchange (REX) system.

4.1 General MDHT Description

4.1.1 Overview

The MDHT system uses a two-step process to retrieve data objects based on IDs. In the first phase (*resolution phase*), the ID is resolved into a list of locators that point to copies of the desired data object. In the second phase (*data forwarding phase*), a set of locators is selected from the list based on configurable parameters (e.g. network conditions). Subsequently, the data object is delivered from the source(s) to the requester.

The MDHT system is independent of the underlying transport and forwarding/routing layer. For example, data transfer can be done via traditional topology-based routing (e.g., OSPF, ISIS, BGP) to ensure efficient data transmission via the topologically best path.

For the resolution phase, the list of object locators is stored in so called *Dictionary Record (D-REC)* entries, containing the ID–locator bindings as well as potentially related metadata. The MDHT dictionary storing the D-REC entries consists of multiple interconnected DHT systems, called *DHT areas*. The DHT areas are arranged in a *nested, hierarchical* structure as shown in Figure 1, building a *DHT tree* structure. Multiple connected DHT areas of the same provider are called *NetInf domain*. The nested arrangement reflects the underlying network topology. Thereby, it minimizes the routing stretch inefficiencies of common DHT systems resulting from missing topological embedding.

Each DHT area represents a network on a different topological level, e.g., the *Autonomous System (AS)* level and the *Point of Presence (POP)* level. The MDHT hierarchy can freely be adapted to the existing network requirements and topology. On the lowest level are the *Access Nodes (ANs)* where *client/host* nodes are attached to the MDHT system, from the clients' point of view somewhat similar to a local DNS server. These ANs use a local Hash Table (HT). Each DHT area can use its own DHT mechanism. For example, classical DHT algorithms like Pastry and Chord with

[1] http://www.worldwidewebsize.com/

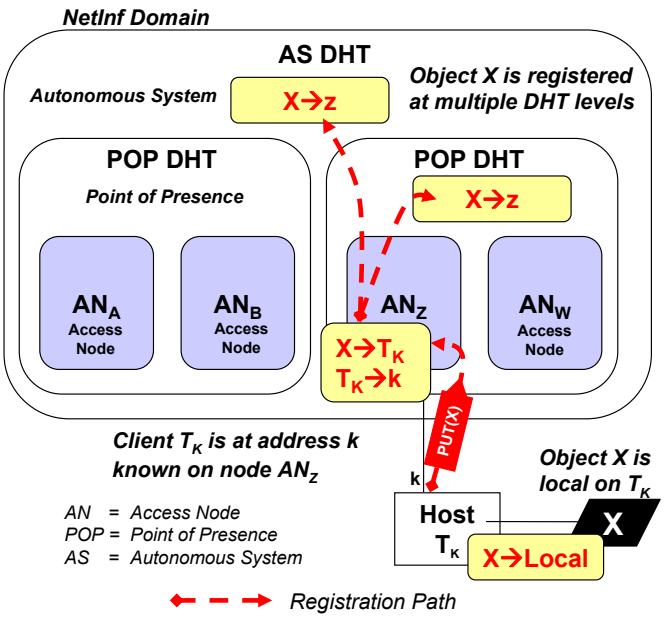

Figure 1: Register object X into an intra-domain MDHT system with areas on three levels: AN, POP, AS

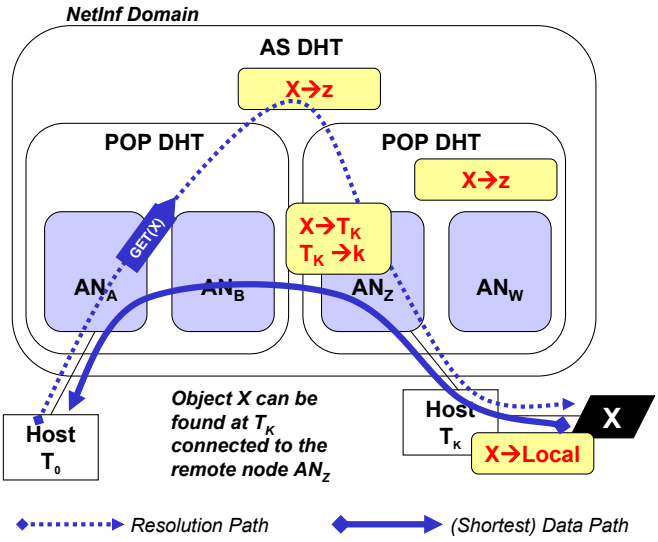

Figure 2: Resolving and retrieving object X

$\mathcal{O}(\log n)$ routing steps can be used. However, since the number of nodes in an infrastructure network is small and stable compared to client nodes in a typical Peer-to-Peer (P2P) overlay network, we expect that most intra-domain DHTs can be operated with $\mathcal{O}(1)$ DHTs.

Each MDHT node participates in its own DHT area and, typically, in some or all higher DHT areas. This means that higher DHT areas are built by aggregating the MDHT nodes of the DHT areas below into a single, larger DHT area, leading to the name "*nested* hierarchical". Each MDHT node can freely choose the number of DHT areas that it participates in. This hierarchical, area-based approach makes deployment of the MDHT system easy as it can grow "from the edges" of the Internet, i.e., the MDHT system can be deployed in small networks first, which can subsequently be interconnected to build a larger system.

Routing and forwarding requests in the MDHT system happens in two separate ways: *Intra-area* routing/forwarding, i.e., within an DHT area, is performed via the respective routing/forwarding mechanism chosen by the MDHT-area provider, e.g. via Chord. *Inter-area* routing/forwarding, i.e., between MDHT areas, is performed via the MDHT nodes that are part of both respective levels. This is done by handing the request to the forwarding process of the next higher level within the same physical node. If a node is not part of the next higher level, the request is forwarded to the next node that participates in both levels.

4.1.2 Data registration and retrieval

The MDHT interface offers two main primitives:

- *PUT(ID, metadata)* is used to register bindings in the network dictionary, i.e., the ID is made public and bound to a set of locators or metadata.
- *GET(ID)* may return a (list of) locator(s) where the data object can be retrieved or the object itself. In addition, related metadata bound to the specified ID can be returned.

To make a new object known and accessible for other users, it has to be *registered* (via *PUT*) in the MDHT system (Figure 1). Two types of bindings can be registered in D-RECs: *location bindings*, which map object/node IDs to network locations, and *indirection bindings*, which map IDs to other IDs. First, the user's device ID T_K is registered in the MDHT system. On the user's AN, T_K is mapped to its local address k, which can be private. Thereafter, the data object can be registered in the user's AN to make it accessible for local users connected to the same AN. Registering a new object creates a new D-REC storing an indirection binding which maps the object ID X into the user device ID T_K. Second, the registration request is propagated up the MDHT tree so that a new binding for object X is recorded in the upper DHT areas along the path from leaf (AN) to root (AS DHT). On all levels (except the top level *REX*, see Section 4.2), the hash(ID) is used as DHT key to store the binding. The used hash function depends on the implemented DHT system.

Note that, in the upper DHT areas, the object ID is mapped to the address z of Access Node AN_Z, where X can be reached via the address k of host T_K. This binding scheme allows to keep host addresses private and reduces firewall issues as responses are received from the initially queried Access Node. In addition, this indirection scheme has good mobility support for mobile users and objects as the Access Node can perform a role similar to a mobile IP "home agent", redirecting requests to the new location of node T_K (details about mobility are, however, out of scope of this paper).

To provide control of ID registrations, the publisher can specify the *scope* of the registration for each ID, i.e., up to which level registration takes place. The scope limits the propagation of the respective D-REC entries within the tree. For example, a publication can be restricted to the local company network.

Figure 2 shows a user (host T_0) *requesting* a data ob-

ject by ID X. The request is first processed in the Access Node AN_A. When resolution in the AN fails because the ID is unknown (i.e., no data copy is registered in this area yet), the request is propagated to the next higher DHT level until a hit is found or the resolution fails on the highest level. By starting the resolution process in the AN and propagating the request higher only if required, the routing stretch is minimized because nodes in lower DHT areas are generally closer (in terms of *no. hops* and *hop latency*) to each other than nodes in higher areas. At the same time, traffic is kept local as this approach always selects a copy from the "closest" area and prevents inter-area traffic whenever possible/desired. This approach also ensures that the MDHT system fulfills the *Content Locality* property: if a local copy is available close to a source, resolution and routing can happen locally. This enables anycast strategies and locality-aware content distribution policies. Given that content is likely to have local access patterns, we also profit from smaller DHT structures.

4.1.3 Copies and caching

When a user resolves and downloads a published data object, a new copy is stored on his local machine. In addition, new copies can be created by caching popular data objects in the MDHT nodes (*in-network caching*). In both cases, the new copy can be made available for other users by simply registering new bindings to the new copies' locators in the MDHT system. Again, the scope of each published copy can be limited, e.g., to limit the load on the user's node or on a caching server.

4.1.4 Binding schemes

MDHT supports multiple binding schemes, i.e., different ways how object IDs are (directly or indirectly) bound to their locations. Object IDs can be bound, e.g., to the (public) address of the server which holds a copy of the object or to the ID of that server. The binding scheme shown in Figure 1 keeps host addresses private at the edge and facilitates mobility of users and objects by means of indirection of objects to user devices and Access Nodes. In general, this kind of indirection is very powerful. It can be used to provide useful capabilities, like private address masking, support of mobility and multihoming, and redirecting traffic towards application servers, e.g., firewalls and accounting servers.

4.1.5 Locator selection

The MDHT system can support three different ways to select appropriate locators from the list of locators for a certain ID. In the *Requester-controlled mode* the MDHT system returns the list of (public) locators to the requester, the requester locally selects a suitable one, and the requester triggers the download of the data. The requester has full control over the process, which is similar to today's DNS system. In the *MDHT-controlled mode*, the MDHT system performs the locator selection and also triggers and handles the data transfer from the source(s) to the requester, i.e., the resolution is transparent to the requester. This is the case shown in Figure 2. Because the MDHT system is embedded in the underlying network infrastructure, it can use its knowledge about the network topology and current network status to select the best locator(s). This integration of resolution and data routing phase also reduces the overall routing stretch. In the *Hybrid mode*, MDHT returns a ranked locator list, based on its network knowledge. The requester can choose the desired locator(s) based on the ranking and other factors, and downloads the data. This mode combines best locator selection based on network knowledge with full requester control.

4.2 Global Name Resolution: REX

Global Name Resolution on the highest MDHT level must be highly scalable because of the large number of globally available data objects. If the namespace is flat without any structure, the top level can use a global DHT composed of all/most of the MDHT nodes from the lower levels. Thanks to the hierarchical MDHT structure and expected locality in the user requests (i.e., users request local content with higher probability), most requests will already be answered in lower levels. Hence, load on the top level will be limited. Our preliminary simulation results support this expectation.

However, using a global DHT at the top level is unfeasible, due to the *key placement* issue: for security and reliability, bindings generated by users of a given network domain should be managed on systems under the control of that domain's provider or on independent systems.

If the namespace is structured, e.g., names contain information about the publisher, global name resolution can be further improved by combining the MDHT system with our Resolution Exchange (REX) system. The REX system is based on independent REX points which offer global resolution services to client ISPs. The REX system, just as the DNS top level domains, can be managed by an independent third party. This trusted third party guarantees its clients the correct management of their resolution bindings.

REX is based on the idea that each owner/publisher has a primary resolution system where his IDs are stored (typically the NetInf domain of the object owner's ISP). In addition, the REX system performs aggregation based on the ID structure. Let us assume an ID structure $A{:}L$, where A is a prefix with some semantic (object owner in the NetInf case), and L is a label which unambiguously identifies the object in the scope of A. This structure can be used to aggregate all data objects of an owner into a single binding, i.e., the system only has to scale with the number of owners and not with the much larger number of data objects. The REX system only stores redirects to the primary resolution system. The A part of each ID can be mapped and redirected to the primary resolution system which is responsible for managing bindings for all IDs with that prefix. Although the REX system can be designed to handle and redirect the resolution traffic itself, we think of it more as an administrative entity. Instead of performing resolution itself, the REX system only manages registrations, updates, and aggregation of bindings on the global level. These aggregated bindings are then cached within intra-domain MDHT systems. By using these cached bindings, requests for IDs registered in other NetInf domains can (already on the levels below REX) be redirected to the appropriate primary resolution system in the remote domain via a topological inter-domain routing protocol. Hence, they do not have to be forwarded to the REX system.

When assuming up to 10^9 aggregated entries (equaling the expected number of users) with each 1 KB, this results in 1 TB of memory which requires to be distributed in the MDHT nodes of each NetInf (transit) domain. Hence, Tier 1 providers will typically cache all REX entries inside their

NetInf domain. Regional providers can choose to download only a subset of the bindings which are associated to close networks, and/or use a default route to upstream providers with a larger cache of REX bindings.

NetInf domain providers can also choose to directly interconnect their NetInf domains below the top level to further improve the topological embedding of the MDHT levels, e.g., at a regional, national, or continental level. This is similar to today's peering agreements between network providers for directly exchanging traffic. For example, some providers can build a joined DHT area at a certain MDHT level. If no sufficient trust relationship exists, providers can still interconnect their NetInf domains by locally caching the aggregated REX entries of the IDs registered in the respective other NetInf domain as described above. In both cases, shortcuts for efficient redirects between NetInf domains are created, reducing resolution load at the higher levels and improving resolution latency.

To behave as desired, all actors must have appropriate incentives that can be fostered in various ways depending on the business model. For example, assuming a global name-based routing system, transit providers have an inherent incentive to update their local REX entries because they sell a global interconnection service.

5. SCALABILITY AND NODE PERFORMANCE ANALYSIS

We give a preliminary and simplified assessment of the system scalability in terms of required number of world-wide MDHT nodes and the nodes' system requirements. For this assessment, we assume a world-wide MDHT system with four levels (AN, POP, AS, Global). As low latency is important, the binding records cannot be stored on a conventional hard drive. However, Solid State Disk (SSD) memory offers sufficiently fast access (15 μs). Current state-of-the-art SSD storage servers have 4 TB of memory (e.g. Tera Ramsan [16]). Since all nodes equally participate in the top DHT and in the nested levels of the hierarchy, they all roughly store the same amount of bindings. Assuming 10^{15} objects globally (Section 2) with binding records (D-RECs) of 1 KB, each MDHT node can store 10^9 binding records on each of the four MDHT levels.

Therefore, 10^6 MDHT nodes are required for a world-wide MDHT system with 10^{15} objects. This is only about 1/10th of the 12 million DNS nodes in today's Internet [17]. If all D-RECs are stored in a Balanced Binary Tree (BBT), up to 30 tree levels are necessary for a resolution lookup inside a single dictionary node. Hence, the storage access latency of a single GET operation is less than 500 μs (15 μs access time $\times$ 30 tree levels = 450 μs).

To evaluate the number of supported users, we assume that the number of GET requests significantly dominates the overall number of requests. With up to 2 million memory access operations/s supported by current SSD storage servers, an MDHT node is able to manage up to 66500 GET requests/s (when assuming 30 levels in the BBT). With an average of 2 GET requests/s per user and the worst case that all requests go up all 4 hierarchy levels, a single MDHT node can still handle more than 8300 users with a single storage unit. Therefore, 10^6 MDHT nodes can handle almost 10^{10} users. Further improvements can be obtained by replicating Access Nodes to exploit parallelism.

Another important issue of dynamic NRS systems is the bandwidth required to perform binding refreshes. We assume a certain level of aggregation of refresh messages (e.g., each packet of 1500 bytes contains on average 10 binding refreshes). Assuming 1% changing entries per day, 40 million entries would change on a single MDHT node, resulting in 4 million refresh messages or 6 GB. Therefore, a node would require $\approx 0,56$ Mbit/s of continuous refresh bandwidth to handle 1% changing entries per day. Even for 10% changing entries, only 5,6 Mbit/s refresh bandwidth would be required.

6. CONCLUSION AND FUTURE WORK

The MDHT system offers an efficient, scalable solution for name resolution in information-centric networks with non-hierarchical namespaces. Preliminary simulation results that we will present in our future work show a low average latency acceptable for real-world deployment. The low latency is a result of the hierarchical, topologically embedded architecture, and locally registered information that leverages locality in the user requests.

For now, MDHT has been considered mainly for global name resolution. However, the MDHT system is suited to play a much larger role as part of an overall ICN architecture, which we will investigate in our future work. For example, MDHT nodes could also perform in-network caching, multicast support, metadata management, security and data integrity checks, data transcoding, and efficient locator selection based on network-internal knowledge. This would allow an ICN network to inherently offer services now only available via P2P overlays (BitTorrent, Skype, etc.) and external Over-The-Top services (e.g., Akamai, Google).

7. REFERENCES

[1] B. Ahlgren, M. D'Ambrosio, C. Dannewitz, M. Marchisio, I. Marsh, B. Ohlman, K. Pentikousis, R. Rembarz, O. Strandberg, and V. Vercellone. Design considerations for a network of information. In *Proc. ReArch2008*, Dec. 2008.

[2] M. Artigas, P. Lopez, and A. Skarmeta. A comparative study of hierarchical DHT systems. In *IEEE Conference on Local Computer Networks (LCN)*, pages 325 –333, October 2007.

[3] M. S. Artigas, P. G. Lopez, J. P. Ahullo, and A. F. G. Skarmeta. Cyclone: A novel design schema for hierarchical DHTs. In *Proc. IEEE International Conference on Peer-to-Peer Computing*, pages 49–56, Washington, DC, USA, 2005. IEEE Computer Society.

[4] M. D'Ambrosio, P. Fasano, M. Marchisio, V. Vercellone, and M. Ullio. Providing data dissemination services in the future Internet. In *Proc. World Telecommunications Congress (WTC'08)*, New Orleans, LA, USA, Dec. 1-2, 2008. At IEEE Globecom 2008.

[5] C. Dannewitz, J. Golic, B. Ohlman, and B. Ahlgren. Secure naming for a network of information. In *Proc. 13th IEEE Global Internet Symposium*, San Diego, USA, March 2010.

[6] P. Ganesan, K. Gummadi, and H. Garcia-Molina. Canon in G major: Designing DHTs with hierarchical structure. In *Proc. Conference on Distributed*

Computing Systems (ICDCS'04), pages 263–272, Washington, DC, USA, 2004. IEEE Computer Society.

[7] J. F. Gantz, C. Chute, A. Manfrediz, S. Minton, D. Reinsel, W. Schlichting, and A. Toncheva. The diverse and exploding digital universe: An updated forecast of worldwide information growth through 2011. IDC White Paper, March 2008.

[8] L. Garces-Erice, E. W. Biersack, P. Felber, K. W. Ross, and G. Urvoy-Keller. Hierarchical peer-to-peer systems. *Parallel Processing Letters*, 13(4):643–657, December 2003.

[9] V. Jacobson, D. K. Smetters, J. D. Thornton, M. Plass, N. Briggs, and R. L. Braynard. Networking named content. In *Proc. 5th ACM International Conference on emerging Networking EXperiments and Technologies (ACM CoNEXT)*, Rome, Italy, December 2009.

[10] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica. A data-oriented (and beyond) network architecture. In *Proc. ACM SIGCOMM '07*, pages 181–192, New York, NY, USA, 2007. ACM Press.

[11] P. Maymounkov and D. Mazières. Kademlia: A peer-to-peer information system based on the xor metric. In *Proc. Workshop Peer-to-peer Systems*, pages 53–65, London, UK, 2002. Springer.

[12] A. Mislove and P. Druschel. Providing administrative control and autonomy in peer-to-peer overlays. In *Proc. 3rd Workshop on Peer-to-Peer Systems (IPTPS'04)*, February 2004.

[13] A. T. Mýzrak, Y. Cheng, V. Kumar, and S. Savage. Structured superpeers: Leveraging heterogeneity to provide constant-time lookup. In *WIAPP '03: Proc. 3rd IEEE Workshop on Internet Applications*, page 104, Washington, DC, USA, 2003. IEEE Computer Society.

[14] V. Ramasubramanian and E. G. Sirer. The design and implementation of a next generation name service for the Internet. In *SIGCOMM '04: Proc. 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 331–342, New York, NY, USA, 2004. ACM.

[15] I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, M. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup protocol for Internet applications. *Transactions on Networking*, 11(1):17–32, Feb. 2003.

[16] Texas Memory Systems. Web source: www.ramsan.com. Last checked: June 2011.

[17] TheMeasurementFactory. DNS survey. Web source: dns.measurement-factory.com/surveys/200810.html, October 2008. Last checked: June 2011.