mDHT: Multicast-augmented DHT Architecture for High Availability and Immunity to Churn

Jae Woo Lee*, Henning Schulzrinne*, Wolfgang Kellerer† and Zoran Despotovic†
* *Department of Computer Science, Columbia University, New York, USA*
{jae,hgs}@cs.columbia.edu
† *DoCoMo Communications Laboratories Europe, Munich, Germany*
{kellerer,despotovic}@docomolab-euro.com

***Abstract*—This paper presents *mDHT*, a novel architectural enhancement to DHT using multicast service discovery. In mDHT, a group of host computers in a subnet participate in a DHT overlay as a single node. A query is routed from subnet to subnet until it reaches the final destination subnet, where it is resolved among the hosts using link-local multicast. Under a reasonable deployment assumption, mDHT offers many benefits over standard DHTs, such as locality, easy bootstrapping, high availability, and near imperviousness to node churn.**

***Index Terms*—mDHT, DHT, multicast, churn, Zeroconf, mDNS, p2p, overlay.**

I. Introduction

We have witnessed two significant advances in peer-to-peer (p2p) networking technology in recent years, driven by the consumers' desire to interconnect at the two opposite ends of networking scale. On the global scale, distributed hash tables (DHTs) [1]–[3] solved the scalability problem of the Internet-wide overlay networks. DHTs impose certain structures into the overlay topologies in order to achieve logarithmic-time lookup of a resource in the network. On the local scale, Zero Configuration Networking (Zeroconf) [4] all but eliminated the need to configure applications and devices to discover and talk to each other in a same subnet. Zeroconf implements service discovery by exchanging link-local multicast packets.

This paper describes an architectural enhancement to DHT using Zeroconf multicast, which we call *mDHT*. In mDHT, the meaning of a "node" is changed from an individual host computer to an entire subnet, i.e., an entire subnet participates in a DHT overlay as a single node. A lookup query is routed from subnet to subnet in mDHT until it reaches the subnet for which the query is destined. The query is then resolved within the subnet using multicast.

Our mDHT architecture can be applied to any existing DHT system. Under a reasonable deployment assumption (the validity of which we reinforce with a measurement of an existing p2p system), mDHT offers numerous benefits including locality, load balancing, easy bootstrapping, high availability, and near imperviousness to node churn.

The rest of this paper is organized as follows. We give background information on hierarchical DHTs and Zeroconf technology in Section II. We delve into mDHT architecture in Section III, starting with an overview and then describing each aspect of mDHT architecture in detail. In Section IV, we discuss the benefits of mDHT, and analyze our deployment assumption using a measurement of a real-world p2p system. We conclude and discuss future work in Section V.

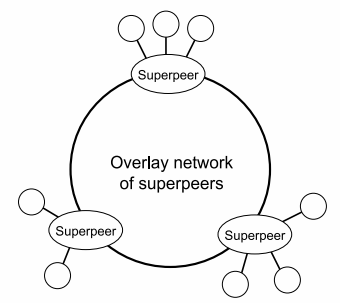
II. Background

A. Evolution of P2P Architecture

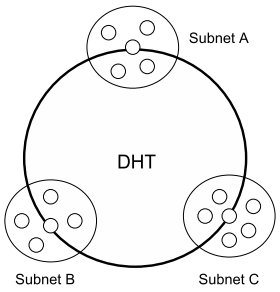
The p2p systems architecture has evolved continuously in order to accommodate the demand of ever-increasing scale of p2p overlay networks. The scalability limit of flat unstructured p2p systems such as the early version of Gnutella, which performed content lookup by flooding the network, inspired the development of structured p2p systems based on DHTs. A DHT network is characterized by an efficient algorithm to map an arbitrary string to a particular node in the network and to produce a routing path of a bounded number of hops from any node to that node. The mapping is deterministic and results in a balanced distribution of the strings among the participating nodes. This enables efficient lookup of distributed data items when each data item is associated with a string (a file name, for example) and the item is stored in the node to which the associated string maps.

Another way to overcome the scalability limit of flat unstructured systems was to introduce hierarchy. In a two-tier hierarchical organization used in the later versions of Gnutella [5], the core overlay network is formed not by every participating node, but by a selected subset called *superpeers*. Each non-superpeer maintains a connection to a superpeer who will act as a gateway to the services offered by the overlay. Figure 1(a) illustrates this arrangement.

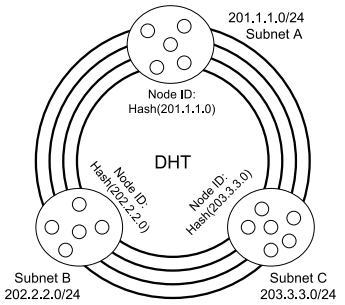
The idea of hierarchical overlay can be applied to DHTs as well as unstructured networks, and the depth of hierarchy can be more than just two levels. Many system proposals exhibit complexities that go well beyond that of the simple two-tier superpeer architecture [6]–[8]. In addition to the obvious advantage of reduced overlay size, hierarchical designs can offer various other advantages, such as taking account of physical network and node heterogeneity, better resilience to churn, administrative control and autonomy, and more efficient caching and load balancing strategies. These advantages apply both to DHTs and unstructured networks, but they are especially beneficial to DHTs, since the rigid overlay structures of DHTs make it harder to incorporate those considerations into flat overlays. Those hierarchical designs that are more



(a) *Two-tier superpeer architecture. The overlay network of superpeers can be an unstructured network or a DHT.*



(b) *A multicast-based superpeer architecture. (This is not our mDHT.) A superpeer is a single point of failure in a subnet.*



(c) *Our mDHT architecture. A subnet is a node in a DHT. The node IDs are chosen by hashing the subnet IP addresses.*

Fig. 1.

complex than the simple two-tier superpeer architecture tend to focus on maximizing performance in one or two of those areas. The simple two-tier architecture, however, still provides some benefits in all those areas compared to the flat overlay design. Moreover, the simplicity of the two-tier architecture is an important advantage over other more complex hierarchical designs, when it comes to developing, deploying and maintaining a large scale overlay network. For these reasons, the simple two-tier superpeer architecture of Figure 1(a) is often the preferred choice for p2p networks on the Internet [9], [10].

B. Zeroconf: Local Service Discovery Using Multicast

When multiple IP-enabled devices are physically connected with one another, Zeroconf makes it possible for one device to use the services provided by another without requiring the user to configure the devices manually. For example, when a user connects two computers either directly using an Ethernet crossover cable or via an Ethernet switch, he will be able to accomplish his file-transfer task by simply starting up the appropriate Zeroconf-enabled applications at both ends. The applications use Zeroconf technology to *discover* each other, without the user having to furnish them with the connection information such as IP addresses and port numbers.

Zeroconf performs local service discovery by exchanging DNS packets via link-local multicast, the details of which are described in a pair of specifications, DNS-based Service Discovery (DNS-SD) [11] and Multicast DNS (mDNS) [12]. DNS-SD defines a set of naming rules for certain DNS record types that it uses for advertising and discovering services. PTR records are used to enumerate service instances of a given service type. A service instance name is mapped to a host name and a port number using a SRV record. If a service instance has more information to advertise than the host name and port number, the additional information is carried in a TXT record.

The DNS records are stored in a collection of mDNS daemons, which are limited-functionality DNS servers running on each host in a local subnet. The mDNS daemons collectively manage a special top-level domain, ".local.", which is used for names that are meaningful only in a local subnet. The queries and answers are sent via link-local multicast using UDP port 5353 instead of 53, the conventional port for DNS. An application can advertise a network service to the subnet by creating appropriate DNS records and depositing them into the mDNS daemon running on the same host. The mDNS daemon will then respond with these records when it hears a multicast query for a matching service. Creating DNS records and storing them with mDNS are usually done by invoking API calls in a DNS-SD/mDNS client library implementation.

Zeroconf technology is widespread today. Bonjour, Apple [13]'s Zeroconf implementation, is an integral part of Mac OS X operating system. Bonjour is also installed on a large fraction of computers running Windows, thanks to the popularity of iTunes—Apple's music playing application—which installs Bonjour for Windows as part of its installation process. For UNIX-like platforms, there is a mature open-source implementation of Zeroconf called Avahi [14], which comes preinstalled in a number of major Linux distributions such as Ubuntu [15].

C. Multicast-based Superpeer Architecture

Figure 1(b) illustrates a possible hybrid of Zeroconf and superpeer-based DHT. A node in a subnet is elected as a superpeer and participates in the DHT. The other nodes in the subnet learn the identity of the superpeer through the superpeer's Zeroconf service announcement, and therefore are able to access the DHT through the superpeer.

At first glance, this architecture seems to offer a reasonable alternative to the regular superpeer architecture of Figure 1(a) if we assume that, on average, a significant number of nodes are found in a single subnet. It is unclear, however, that the use of multicast provides much benefit at all. Moreover, a major weakness of superpeer architectures is still present: the superpeer is a single point of failure among the nodes attached to that superpeer. We will not consider this model any further. It is presented here as a conceptual bridge leading to our mDHT architecture, and to ensure that the reader does *not* conjure up this model as his or her mental image of mDHT.

III. mDHT Architecture

A. Overview

Figure 1(c) illustrates our mDHT architecture. An entire subnet, not an individual host, becomes a "node" and participates in a DHT. A node identifier (node ID) must be assigned to a subnet as a whole, so it cannot be based on an IP address of any individual host. Figure 1(c) shows one way to assign node IDs on subnet level: node IDs are chosen by hashing subnet IP addresses. Other methods of ID assignment can be used as long as a single ID is assigned to an entire subnet and that ID is propagated to all participating hosts in the subnet.

Messages are routed in the same way they are routed in regular DHTs. A query is routed among the nodes until it reaches its destination according to the particular DHT algorithm used in the overlay. In mDHT, a node is a subnet. Once a query reaches the destination subnet, the query is resolved among the hosts in the subnet using link-local multicast.

Our mDHT can be applied to any DHT since its operation depends only on the generic facilities common to all DHTs such as routing tables or node identifiers. Nevertheless, it is often helpful to use a specific DHT when referring to a part of data structure or a maintenance procedure, since terminology varies across DHTs. In those cases, we use Chord [1]. Also, when we use the term *node*, we mean a logical entity that is given a node ID, which is a host computer in a regular DHT, but a subnet in mDHT. We use the term *host* to refer to individual computers.

B. Routing Table

We explained that a query is routed in mDHT just as it is routed in a regular DHT, passing through intermediate nodes and finally reaching the node to which it is destined. Once the query arrives in the destination subnet, it is resolved among the hosts in the subnet using multicast. But how does the message travel from a node to another when a node is a subnet, not a host? Message transfer is still based on TCP/IP networking, and there is no such thing as sending a message to a subnet.

Suppose a subnet *A* is a node in a mDHT overlay and, among the hosts in the subnet *A*, three hosts *a1*, *a2*, *a3* have joined the overlay. (The three hosts share a single node ID, *hash(A)* for example, as explained in Section III-A.) Suppose a subnet *B* is also a node, with hosts *b1*, *b2*, *b3* participating. Imagine that *a1* has issued a query, and the DHT algorithm has determined that the node *B* is the next hop. In order for *a1* to send a message to the subnet *B*, it needs to know at least one specific host in that subnet. Let's assume that *a1* knows that *b1* and *b2* reside in *B*. The host *a1* randomly picks one of the two hosts in *B*, say *b2*, and sends the message. If *B* is the final destination for the message, *b2* will switch to multicast to resolve the query in its subnet. If not, it will find the next hop, say *C*, and proceed in the same way as *a1* did before. (This is assuming that recursive query routing is used in the DHT; if iterative routing is used, *b2* will tell *a1* where the next hop is and *a1* will repeat the procedure.)

Each host in a DHT carries a routing table that has a list of nodes. In Chord, a routing table entry (called a *finger*) points to a host node, and it consists of the node's ID, IP address and port number. In mDHT, a node is a subnet, and the routing table entry for a node needs to include a *host set*, the IP addresses and port numbers of the participating hosts in the subnet. For example, a mDHT implementation based on Chord may redefine the finger as a collection of the following information: SHA-1 hash of a subnet IP address as a node ID, the subnet IP address, and a host set of maximum 8 IP addresses and port numbers.

In our example of *a1* sending a message to *B*, *a1* randomly picked one host from the host set for *B*. When iterative query routing is used, there is another option. The same message can be sent to multiple hosts in the host set. In the example, *a1* can send the query to both *b1* and *b2*, and take the faster response. This will shorten the overall lookup latency by reducing timeout delays from failed hosts. It will also help maintain the host sets, as unresponsive hosts can be removed from them. This mechanism is similar to sending parallel queries to multiple adjacent nodes, available in some DHTs such as Kademlia [3]. The difference is that, in mDHT, parallel queries are sent to a *single node*.

C. Host Set Maintenance

The host sets in the routing tables need to be periodically updated. The authoritative list of active hosts in a subnet comes from the hosts in the subnet themselves. Each host monitors the multicast announcements sent by other hosts as they join and leave the overlay, and keeps track of the list of active hosts in the subnet. Zeroconf API makes this easy.

This list of active hosts in a subnet is propagated to all the routing table entries that point to this subnet using a combination of push and pull methods. Every host periodically refreshes its own routing table entries by contacting one of the hosts for an up-to-date list of active hosts. This can be incorporated into the regular DHT maintenance procedures such as Chord's FIX_FINGERS(). An updated list can also be pushed, on a join or leave event in a subnet, onto the neighboring nodes such as Chord's successor and predecessor.

D. Host Join and Leave

When there is no other participating host in a subnet, host join and leave in mDHT follow the same procedures of regular DHTs.

When a subnet already contains one or more participating hosts, joining and leaving the mDHT overlay from that subnet is almost trivial. A newly joining host simply makes a multicast announcement. The other hosts, which are monitoring the multicast announcements, add the new host into their lists of active hosts, which will eventually find their way to the routing tables of other nodes (Section III-C). A leaving host also makes a multicast announcement, telling the other hosts to remove it from their lists of active hosts. In addition, a host may need to transfer data when joining or leaving, according to the data replication policy in place (Section III-E).

E. Data Replication in Subnet

When DHT is used as a data storage and lookup facility, a data item is mapped to a particular node by the DHT algorithm. In the case of mDHT, where a node is a subnet, there is a question of which host (or which set of hosts) will store a data item mapped to the subnet.

The simplest strategy is to have one host store a particular data item, most likely the host that happened to receive the initial message for depositing the data item into the DHT. If a lookup request for that data item arrives at a different host in the subnet, it will issue a multicast query, to which the host owning the data will respond. When the host owning the data item leaves the overlay, it must transfer the data item to another participating host. This strategy does not provide any shield against host failures at mDHT level. However, the data replication schemes of regular DHTs such as Chord's successor-list can still be used.

On the opposite end, another strategy is to replicate all data items fully within the subnet. When a host receives a new data item, it is immediately propagated to all participating hosts in the subnet. This results in a high data injection cost, but the lookup latency caused by the multicast query is eliminated. A host failure is not a problem as long as there is another participating host in the subnet. On the other hand, a newly joining host needs to copy all existing data from a neighbor.

The optimal strategy is likely to be somewhere in the middle. A simple, yet reasonable approach might be to try to replicate data in a fixed number of hosts. Obviously the number represents a maximum since there may not be as many participating hosts in the subnet. Another possible strategy is to start with a low number of replicas and increase the number per data item as the node receives more and more lookup queries for the item.

IV. Discussion

A. Benefits of mDHT

1) Immunity to Churn: High rate of churn—the continuous process of hosts joining and leaving an overlay—has been a difficult problem in DHT design. In mDHT, as long as there are other participating hosts in a subnet, hosts joining and leaving in that subnet has *no effect* on the DHT structure. The subnet remains as the same node in the DHT as individual hosts come and go, and there is no need to fix anything in the DHT structure.

Contrast this with the superpeer architecture we saw earlier in Figure 1(a). A superpeer represents a single point of failure among the nodes attached to it. If a superpeer leaves the overlay, presumably one of the non-superpeers can step up to become a new superpeer to hold the same group together, but even then, the new superpeer needs to be repositioned in the DHT overlay because its node ID is different from that of the previous superpeer.

2) High Availability: Many DHTs use data replication and parallel queries to increase the availability of data items stored in a DHT as a whole. Achieving high availability of a specific node, however, is not so straightforward. Our mDHT, on the other hand, can increase the availability of a node simply by adding more hosts to the subnet. This ability to strengthen a specific node would be particularly useful when it is used with a DHT that also provides administrative autonomy and controlled data placement, such as SkipNet [16].

3) Easy Bootstrapping: The use of multicast makes it easy for a new node to discover and join an mDHT overlay when there is another participating host already present in the subnet. This may reduce the load on the global bootstrapping servers in some systems, since only the first participating host in a subnet needs to contact a bootstrapping server.

4) Parallel Queries and Load Balancing on Single Node: As explained in Section III-B, the fact that a node is a subnet consisting of multiple hosts enables mDHT to send parallel queries to a single node, as opposed to Kademlia's parallel queries which are sent to multiple adjacent nodes.

Within a subnet in mDHT, the participating hosts naturally share the load, since a random subset of the hosts receive each query. This is again in contrast with the superpeer architecture of Figure 1(a), where a superpeer represents a single point of failure and possibly a bottleneck.

There is a subtle issue, however, when we consider load balancing among all the hosts in the whole overlay. The fact that a node in mDHT is a subnet, and therefore contains a varying number of hosts in it, may conceivably result in a load deviation worse than those of standard DHTs. We conjecture that a DHT load balancing strategy such as [17] will be as effective on mDHT as it is on standard DHTs. Analysis and simulation to support this conjecture is planned as a future work.

Another point to note is that DHT load balancing is not such a serious problem in practice, since the expectation is that most hosts will only consume a small fraction of the host's resources. A more important task is mitigating "hot spots" caused by exceedingly popular items. Standard DHTs use replication to deal with hot spots. Our mDHT can do the same. In addition, node redundancy can be increased when the DHT algorithm allows controlled data placement (Section IV-A2).

5) Awareness of Physical Proximity: A mDHT node represents a grouping of p2p participants in closest proximity to one another. A p2p application built on top of mDHT can take advantage of this locality property in various ways. For example, a file sharing application can reduce data traffic by having a host computer cache popular contents, not only for the purpose of repeated retrieval, but for making it available for the other hosts in the subnet. The cache inventories can then be exchanged via multicast among the hosts.

B. Analysis of Assumption

The claimed benefits of mDHT depend on the assumption that the majority of the subnets contain multiple hosts participating in the overlay network. In order to test this assumption on a real-world p2p network, we examined the IP addresses of 9582 Skype relay hosts, which were collected as part of an

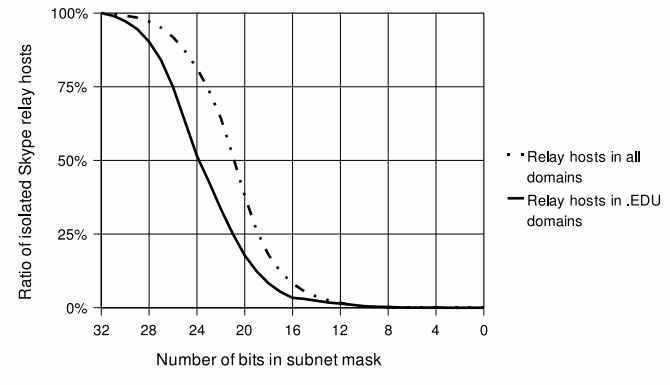


Fig. 2. *Ratio of the hosts that are participating alone in their subnets.*

experiment by Kho *et al.* [18], measuring Skype relay calls over a three-month period.

Since it is difficult to determine the subnet mask of an arbitrary IP address, we simply fix a hypothetical subnet mask of certain number of bits, and group those IP addresses that fall into the same subnets under the fixed subnet mask. Figure 2 shows the ratios of the *isolated hosts*, the hosts that are alone in their subnets, as we vary the subnet mask from 32 to 0 bits. (The 32-bit subnet mask is the vacuous one where each host IP address becomes its own subnet; and the 0-bit mask is the degenerate one where the whole Internet is a single subnet.) The dotted line plots a result for the whole data set of 9582 hosts, and the solid line for a subset of 2150 hosts that belong in the .EDU domain.

The result from the .EDU hosts (the solid line) supports our assumption. With 24-bit subnet mask (i.e., /24 subnet), half of the hosts have at least one other host in their subnets. Moreover, subnets in University campuses tend to be larger than /24. The 21-bit mask reduces the ratio of isolated hosts down to 25%.

It is harder to justify our assumption if we include not only the .EDU hosts, but all the hosts in the data set (the dotted line). It takes the 21-bit mask to achieve 50%, and the 19-bit mask to get down to 25%. It should be noted, however, that the majority of the IP addresses in this data set belong in the domains of residential ISPs [18], indicating that they are home computers. The access networks of residential ISPs are not likely to allow multicast between subscribers, so they represent hostile environments for mDHT. But we observe two encouraging trends that point to a future direction of residential networks that is more favorable to mDHT. First is the rise of home networks. Many households have multiple networked devices, and the use of a NAT router to form a home network is becoming commonplace. (The residential IP addresses in our data set do not include any host behind NAT, since Skype does not choose such a host as a relay.) Second, residential ISPs are increasingly concerned about the traffic generated by p2p applications, and they are looking for ways to reduce the traffic [19]. A residential ISP can reduce the traffic generated by a mDHT-based application simply by enabling multicast among the users in a neighborhood, so that they can share content cache as described in Section IV-A5.

V. Conclusion and Future Work

We presented mDHT, a novel hybrid architecture that augments DHT with multicast service discovery. Our mDHT shares some of the positive traits of the traditional two-level superpeer architecture, but we eliminate the single point of failure in a peer cluster. In addition, the redundancy within a node makes mDHT impervious to churn, and offers an easy way to increase node availability.

In the future, we plan to implement a prototype, and prove our conjecture on load balancing through simulation.

References

[1] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM, 2001, pp. 149–160.

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

[3] P. Maymounkov and D. Mazières, "Kademlia: A peer-to-peer information system based on the XOR metric," in *IPTPS*, 2002, pp. 53–65.

[4] Zero Configuration Networking. [Online]. Available: http://www.zeroconf.org/

[5] Gnutella protocol 0.6. [Online]. Available: http://rfc-gnutella.sourceforge.net/src/rfc-0_6-draft.html

[6] P. Ganesan, K. Gummadi, and H. Garcia-Molina, "Canon in G Major: Designing DHTs with hierarchical structure," in *ICDCS '04: Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04)*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 263–272.

[7] M. S. Artigas, P. G. Lopez, J. P. Ahullo, and A. F. G. Skarmeta, "Cyclone: A novel design schema for hierarchical DHTs," in *P2P '05: Proceedings of the Fifth IEEE International Conference on Peer-to-Peer Computing*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 49–56.

[8] L. Garces-Erice, E. W. Biersack, K. W. Ross, P. A. Felber, and G. Urvoy-Keller, "Hierarchical P2P systems," in *Proceedings of ACM/IFIP International Conference on Parallel and Distributed Computing (Euro-Par)*, Klagenfurt, Austria, 2003.

[9] LimeWire. [Online]. Available: http://www.limewire.com/

[10] D. Bryan, P. Matthews, E. Shim, and D. Willis. (2007) Concepts and Terminology for Peer to Peer SIP. Internet draft. [Online]. Available: http://tools.ietf.org/html/draft-ietf-p2psip-concepts-01

[11] S. Cheshire and M. Krochmal. (2006) DNS-based service discovery. Internet draft. [Online]. Available: http://files.dns-sd.org/draft-cheshire-dnsext-dns-sd.txt

[12] ——. (2006) Multicast DNS. Internet draft. [Online]. Available: http://files.multicastdns.org/draft-cheshire-dnsext-multicastdns.txt

[13] Apple Inc. [Online]. Available: http://www.apple.com/

[14] Avahi. [Online]. Available: http://avahi.org/

[15] Ubuntu. [Online]. Available: http://www.ubuntu.com/

[16] N. Harvey, M. B. Jones, S. Saroiu, M. Theimer, and A. Wolman, "Skipnet: A scalable overlay network with practical locality properties," in *In proceedings of the 4th USENIX Symposium on Internet Technologies and Systems (USITS '03)*, Seattle, WA, March 2003.

[17] J. W. Byers, J. Considine, and M. Mitzenmacher, "Simple load balancing for distributed hash tables," in *IPTPS*, 2003, pp. 80–87.

[18] W. Kho, S. A. Baset, and H. Schulzrinne, "Skype relay calls: Measurements and experiments," *Computer Communications Workshops, 2008. INFOCOM. IEEE Conference on*, pp. 1–6, April 2008.

[19] ISPs experimenting with new P2P controls. [Online]. Available: http://www.networkworld.com/news/2008/061908-nxtcomm-isp-p2p.html?page=1