BOOTSTRAPPING LOCALITY-AWARE P2P NETWORKS

Curt Cramer, Kendy Kutzner, and Thomas Fuhrmann
Institut für Telematik, Universität Karlsruhe (TH), Germany
{curt.cramer|kendy.kutzner|thomas.fuhrmann}@ira.uka.de

***Abstract*— Bootstrapping is a vital core functionality required by every peer-to-peer (P2P) overlay network. Nodes intending to participate in such an overlay network initially have to find at least one node that is already part of this network. While structured P2P networks (e.g. Distributed Hash Tables, DHTs) define rules about how to proceed after this point, unstructured P2P networks continue using bootstrapping techniques until they are sufficiently connected.**

In this paper, we compare solutions applicable to the bootstrapping problem. Measurements of an existing system, the Gnutella web caches, highlight the inefficiency of this particular approach. Improved bootstrapping mechanisms could also incorporate locality-awareness into the process. We propose an advanced mechanism by which the overlay topology is – to some extent – matched with the underlying topology. Thereby, the performance of the overall system can be vastly improved.

I. INTRODUCTION

Overlay networks are virtual networks layered on top of an existing network (“underlay”) like the global Internet. Only a subset of all underlay nodes participates in the overlay. Since overlay networks typically organize themselves in a P2P manner, no centralized control or knowledge about them is available. In order to join an overlay, a node needs to learn the network layer address of at least one node currently participating in the overlay network (*“bootstrapping”*). Obviously, this has to be done by some out-of-band means, external to the overlay. In a second phase, the newly attached node may begin with *topology refinement*, i.e. locally optimize its position in the overlay. This can be done by means internal or external to the overlay.

It is commonly assumed or acknowledged [1], [2] that overlays require a so-called *bootstrapping service* for solving this initial start-up problem. However, so far little effort has been made to thoroughly understand and efficiently solve this problem.

Current research focuses on overlay networks which provide object lookup services, so-called “distributed object location and routing” [3]. I.e., such a network provides operations for storing data objects under a given key and for retrieving an object designated by a key. These overlay networks can be classified as being either unstructured like Gnutella [4] and Freenet [5], or structured, like Chord [6], CAN [7], and Pastry [1]. Unstructured overlays have random graph topologies and cannot offer object lookup guarantees. Since their topologies are very flexible, they may be freely adapted to the underlying network topology to optimize the performance of message routing.

Structured overlays – so-called “Distributed Hash Tables” (DHTs) [6], [7], [1] – impose rigid topologies on the participating nodes and create a well-defined mapping from object keys to these nodes. Thereby, they can offer strong guarantees on lookup efficiency in terms of overlay hops. However, DHTs are often very limited in their ability to adapt to the underlying topology [8]. This leads to comparatively high round trip times within the overlay and unnecessarily increases the load imposed onto the underlay.

Recent research in the overlay network community focuses on improving the performance of P2P networks by incorporating locality-awareness (see e.g. [9], [8] for an extensive list of publications in this field) into their protocols. We believe that the construction of efficient network structures should already be addressed in the bootstrapping phase.

This paper is structured as follows: Section II gives a survey of proposed or currently employed mechanisms for constructing bootstrapping services. Measurements of a popular and currently deployed bootstrapping system, the Gnutella web caches, show that state-of-the-art systems are far from being optimal.

In section III, we present a new way for supporting topology refinement by bootstrapping. Its key idea is to use the overlay network itself to support the refinement phase. Measurement results demonstrate the viability of our approach. Finally, section IV concludes with an outlook on future work.

II. APPROACHES TO SOLVE THE BOOTSTRAPPING PROBLEM

There is a number of straightforward solutions for bootstrapping an overlay network. In this section, we briefly highlight their respective benefits and limitations. We present results of a five-week measurement study of the Gnutella web cache system, which is the bootstrapping service of the Gnutella file sharing network. These results are supplemented with results from an 18-day measurement study of the Freenet network.

A. Static Overlay Nodes – Bootstrapping Servers

In original P2P networks like Gnutella [4], initial bootstrapping was solved by placing static nodes (i.e. servers) in the overlay. P2P client software contained hard-coded DNS names of the bootstrapping nodes. These bootstrapping nodes, called “pong caches” [4] in Gnutella, collected topological information about and addresses of other nodes participating in the overlay. On request by a newly joining node, the pong caches returned a list of addresses of nodes seen recently in the overlay. The new node then tried to establish overlay connections to the nodes in this list.

Afterwards, nodes periodically flooded so-called *ping* messages in their overlay neighborhood to learn about other nodes to which they could connect to. All nodes receiving ping messages, i.e. all nodes up to a certain hop limit ("horizon"), answered by sending a *pong* message back into the direction of the original *ping*. (It have been these pong messages that were collected by the pong caches for helping the subsequently joining nodes.) Since this approach relies on flooding, its scalability is limited. Measurement studies [10] have shown that as much as 55% of the whole traffic generated in a Gnutella network resulted from the exchange of ping-pong messages.

This bootstrapping method is very simple. It relies on the passive collection of topology data without any rating or refinement. Moreover, this method lacks scalability [2] and requires at least some administrative overhead, since it relies on centralized components. Although more bootstrapping nodes could be added to scale the system, their addresses would have to be manually entered into the DNS. These addresses are returned to DNS clients as a list or in a round-robin fashion. The original CAN paper [7] proposed that a list of bootstrap nodes should be resolved from a DNS name. These nodes manage lists of CAN nodes of which they return a random subset upon request.

This use – or rather misuse – of DNS meanwhile has some tradition in solving problems like node mobility and service discovery in distributed systems.

B. Out-of-Band Address Caches

The Gnutella community implemented another bootstrapping system, the Gnutella web caches [11], to overcome the limitations of the pong cache mechanism. E.g., without further modifications, it is not possible to provide query parameters other than a name to the DNS.

Nodes in the P2P overlay actively report suitable nodes (e.g. with long uptimes) to HTTP-based caches. Joining nodes contact the caches in order to get a list of IP addresses. The caches themselves are accessed via URLs.

The key problem here is the web caches' content getting outdated very quickly. Measurements of P2P file sharing networks have shown that the rate of nodes joining and leaving the network is quite high [12], [13]. In order to detail these results, we recently have performed a long-term measurement study of the Gnutella web caches. The caches were monitored over a period of 5 weeks in intervals of 10 minutes. A similar study by a group at the Georgia Institute of Technology [14] with a coarser measurement interval of 30 minutes largely agrees with our observations.

We measured the cumulative availability of distinct *(IP address, TCP port)* pairs in the cache directories. Fig. 1 shows a double-log plot of the results. Out of 664,723 distinct nodes, a very large fraction (63.1%) was available for less than or equal to 30 minutes. Only 32 nodes were available for more than a week. The average session duration is 80.96 minutes.

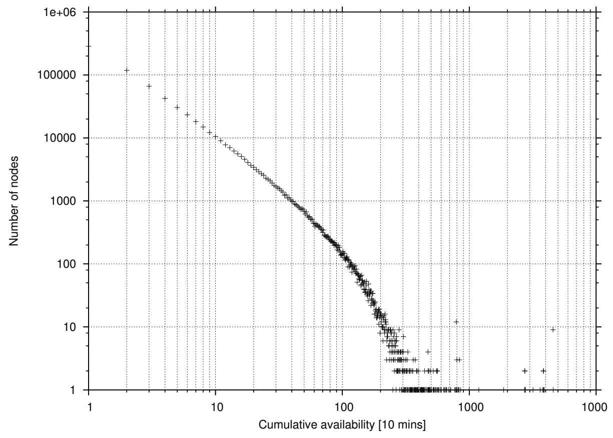

Fig. 1. Measured availability of nodes in the Gnutella web caches

To complement these observations, we measured the duration of connections between nodes in the Freenet network. Fig. 2 shows the durations of 7,979,265 TCP sessions as seen from a Freenet node running in our lab over an 18-day period. The mean session duration here is 34.66 seconds, and 99.7% of all sessions last for less than 30 minutes. Only 0.11% of all connections stay available for more than one hour. Both the plot for Gnutella and for Freenet exhibit a typical power-law structure, where a large fraction of nodes stays in the overlay for only very short time intervals.

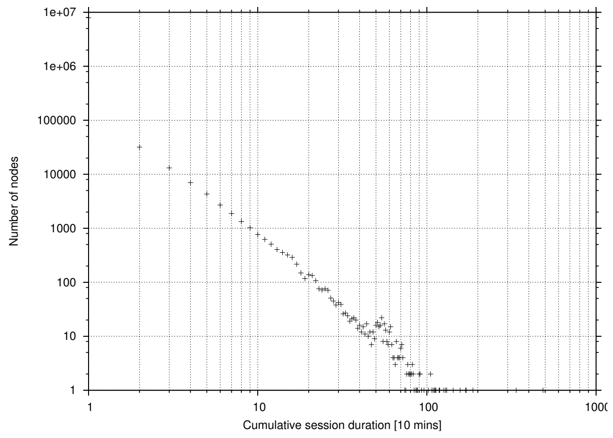

Fig. 2. Measured availability of nodes (i.e. session durations) in Freenet

According to [14] and our observations, the percentage of nodes listed in Gnutella web caches effectively online and accepting overlay connections can be as low as 16%. Because of this low success rate, nodes wishing to join the overlay typically need to make several attempts before they can successfully connect to the network. This lengthens the bootstrapping time to several minutes.

Even worse, this system again requires the client software to contain some hard-coded URLs for finding the node caches. If the caches determined by this small set of URLs are unavailable, the client will not be able to connect to the network at all.

C. Random Address Probing

For large-scale overlay networks, random address probing could be suitable for bootstrapping. A node wishing to enter the overlay randomly draws an IP address from the global (or local) address space. It then tries to establish a transport connection to this IP address using a *well-known* port. In case the connection cannot be established because the node does not exist or does not participate in the overlay, another address has to be tried.

The failure probability when randomly selecting addresses is dependent on the distribution of overlay nodes in the probed address range. (Our evaluations in section III show that these distributions significantly vary between different overlays.) Nodes behind NATs or firewalls typically cannot accept incoming connections. Hence, they must not be considered as potential entry points for bootstrapping. They will, however, be able to use bootstrapping to connect to other nodes.

Since these techniques are known to be successful for malicious purposes like viruses and worms, they must be applied with care to avoid false alerts.

D. Employing Network Layer Mechanisms

One can argue that – ultimately – the discovery of overlay nodes while bootstrapping should be based on the topological structure of the underlying network. I.e., if there are overlay nodes in the same network segment, it is both convenient and highly efficient to use network layer mechanisms to connect to these nodes, at least for bootstrapping.

In a multicast-capable network, e.g. in a local area network (LAN), a multicast group can be established for bootstrapping purposes. Each node participating in the overlay has to join the group. Newly joining nodes then can query the multicast group for IP addresses of nodes already participating in the overlay. Once a node has received these addresses, it can establish overlay connections to these nodes. In order to handle both densely and sparsely populated network domains without risking an explosion of the accompanying traffic, the search should be performed using e.g. the expanding ring search technique.

Often, multicast traffic is not routed beyond the local network domain. This bootstrapping technique therefore must be complemented by alternative approaches.

Anycast, too, can be used to acquire the IP address of a potential peer. However, this limits the requesting nodes' freedom of choice, since node selection is done at the network layer alone, not at the overlay layer. Anycast can therefore only be used as an initial step and should be followed by an overlay-specific refinement phase. Nevertheless, with the advent of IPv6 and its anycast facilities [15], this might be able to supersede the current "workaround" solutions.

Bootstrapping distributed systems has also been studied in the area of service discovery for IP networks (see e.g. [16] for an overview of different service discovery protocols). Mostly, however, the approaches discussed in that field require manual administration and centralized servers, e.g. for storing service descriptions. Both aspects contradict the notion of self-organization in P2P systems. We will therefore not continue to discuss these approaches here.

III. Incorporating Locality-Awareness into Bootstrapping

In this section, we propose a bootstrapping service that supports P2P overlay networks in discovering nodes that are nearby in terms of the underlay network topology. Specifically, we address the topology refinement phase of the bootstrapping process, since this phase is crucial for the performance of a P2P system. A P2P system may use this mechanism immediately after finding one initial node that already is in the overlay. We propose to implement this initial step by one of the methods described in section II. If a node is to participate in several P2P systems, all these systems could use the same bootstrapping service. Then, it suffices to find one initial node for bootstrapping all of its overlays from that node.

The key idea of our approach is to publish locality information as a directory in the overlay network itself. (For simplicity, we assume a DHT in the following.) Each node hashes a reasonable part of its IP address, e.g. its network mask or an arbitrary k-bit prefix of its IP address, to a key and publishes its IP address under this key in the DHT. The DHT has to be able to store multiple values under the same key. As a result, a newly joining node can look up the exact IP addresses sharing a prefix with its own IP address. Such nodes are likely to be nearby in terms of the network topology. With long prefixes, storage and maintenance costs are reasonable.

The shorter the prefix length employed, the worse the locality approximation will get. Moreover, shortening the prefix length will increase the retrieved number of addresses. Depending on the node density, this may lead to very large address lists. For both reasons, nodes should start the discovery with a long prefix which is successively shortened until a sufficient number of nodes was found or a lower limit on the prefix length is reached. This "bottom-up" approach resembles the expanding ring search technique mentioned before.

Alternatively, the directory can use a "top-down" approach. There, a node starts querying with a short prefix (e.g. 10.0.0.0/8) and the service returns all prefixes subsumed under the queried prefix (e.g. a list of 10.$*$.0.0/16 prefixes). The querying node then selects suitable prefixes from the resulting set to repeatedly query the service for longer prefixes until a suitable node address is found. (The service could also carry out the entire iterative process recursively on behalf of the querying node.)

The choice between "top-down" and "bottom-up" is governed by a trade-off between the latency of the DHT in case of negative responses and the DHT's ability to respond to popular requests in a scalable manner. The "bottom-up" approach avoids concentrating lookups but needs to wait for negative responses from the DHT. With the "top-down" approach, many lookups are made for the same short prefixes. If those e.g. are efficiently cached, the lookup procedure is speeded up. With both approaches, the hierarchical organization of the discovery process guarantees manageability of storage and maintenance overhead.

To experimentally evaluate the viability of our method, we used the result sets of our measurements as described in section II-B and performed simulations based on simple models derived from the measurement results.

For reference, we first assumed a simple model where nodes are uniformly distributed over the full address range. A random sample of 664,723 uniformly distributed IPv4 addresses was generated. For each address, the prefix length for discovering at least one additional address with the same prefix – starting with a length of 32 bits, which is successively reduced until a node is found – was calculated. Fig. 3 shows the absolute frequencies of the resulting *minimum prefix lengths*. Additionally, we also determined the total number of distinguishable prefixes for a given length, i.e. the *cumulative distribution function of the prefix lengths*. The results are also shown in fig. 3.

One can see that under the assumption of uniform node distribution, the lookup procedure can be expected to find nodes for prefixes in the range 18 to 21.

Subsequently, we checked the model assumptions against the measurements made in the Gnutella file sharing network. There, the same evaluations were made, this time for a sample of 664,723 nodes participating in that network. Fig. 4 depicts the results. This clearly shows that the uniform distribution assumption made above does not hold. In the Gnutella network, nodes tend towards being situated in networks with longer prefixes. Most of the nodes belong to networks with prefix lengths between 24 and 28 bits. Thus, even with a bitwise reduction of the prefix length (or expansion respectively), the lookup procedure would terminate after at most five steps. If the lookup steps combine pairs, triples, or quadruples of bits, the process could be accelerated further. Thus, we conclude that our proposed mechanism has a high probability of success with low overhead.

For a further confirmation of that positive result, we evaluated a second sample of 19,235 distinct IP addresses in the Freenet network. The results are shown in fig. 5. Although the distribution's spread is wider than in the case of Gnutella, the general trend of nodes being distributed much more favorably than in the uniform distribution model is confirmed.

We believe that the observed distribution is a general property of P2P networks and the shift in the Freenet measurement is mainly due to its smaller sample size. To further test this hypothesis, we studied the effects of sample size on the results of our evaluation. To this end, we reduced the size of the Gnutella sample (664,723) to that of the Freenet sample (19,235). As can be seen from the resulting fig. 6, the distribution also is shifted towards shorter prefix lengths. From this, we conclude that for more nodes participating in the network, the distribution and hence the probability of finding a nearby node would again be increased to the value observed in our original Gnutella measurement. This evidence for a

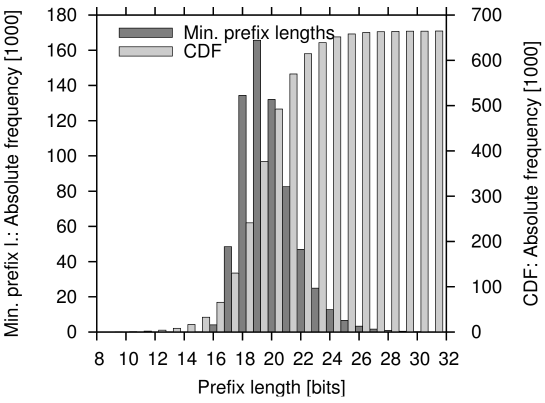

Fig. 3. Min. prefix lengths and CDF of prefix length, random addresses

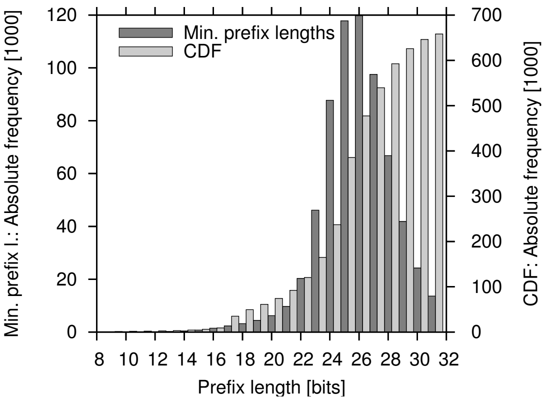

Fig. 4. Min. prefix lengths and CDF of prefix length, Gnutella

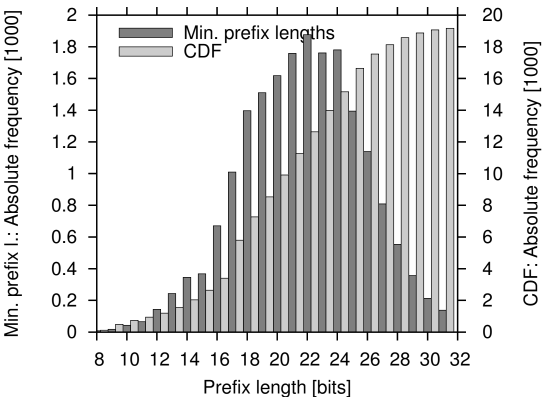

Fig. 5. Min. prefix lengths and CDF of prefix length, Freenet

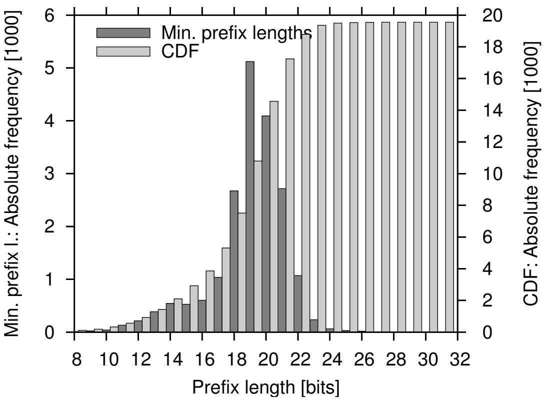

Fig. 6. Min. prefix lengths and CDF of prefix length, Gnutella (reduced sample size)

presumed general property of P2P networks is also supported by the observation that the popularity of P2P networks shows regional differences.

From all this, we conclude that our proposed mechanism can be quite efficiently exploited for a large class of P2P networks, both by supporting the observed trend to regional differences in the usage of different P2P networks, and by speeding up the bootstrapping process in general.

Our approach is related to one presented by Garces-Erice et al. [17]. They build structured overlays by grouping nodes according to their IP address prefixes as declared in the routing tables. Nodes preferably maintain overlay connections to other nodes in their respective groups. Their work differs from ours in two points: First, we do not rely on the availability of the address prefixes from some external source, like BGP routing tables. Second, our service is supplemental to any overlay and does not modify the topology formation algorithm itself. Thus, we believe that our approach provides sufficient flexibility to serve as a building block for the improvement of various types of overlay networks.

IV. Conclusions and Future Work

Although the need for a topology-aware bootstrapping mechanism for P2P networks is obvious, little research regarding this topic has been conducted so far. Most deployed P2P networks still neglect any relation to the topology of the underlying network when forming their overlay. On the other hand, proposals to overcome this nuisance often are not orthogonal to protocol functionality, therefore requiring major adaptations.

In this paper, we proposed an alternative approach, namely a bootstrapping service that supplements the topology refinement phase of P2P networks without entirely modifying the respective P2P protocol. Thereby, P2P systems can be engineered using this bootstrapping service for topology refinement as a building block only.

The development of our bootstrapping mechanism was motivated by a survey of methods addressing the bootstrapping phase of P2P networks. Our analysis, which was backed by measurements, has shown that ironically one of the most popular approaches, out-of-band caches, is inefficient. With our new, service-based approach, P2P networks can be enhanced to become topology-aware, thus saving resources of the underlying network infrastructure.

Up to now, the notion of locality used by us is solely based on proximity in the address space of the Internet. Although this notion is suitable to find peers in the same organization or in the same network region, a more general concept of locality is desirable. The quality of the distance approximation using network prefixes largely depends on the network type. E.g., in an environment consisting mainly of analog dial-up lines, locality in the address range yields high delays, which P2P users might perceive less favorable than locality defined directly in terms of end-to-end delay.

References

[1] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems", in *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms (Middleware) 2001*, Heidelberg, Germany, Nov. 2001.

[2] B. Wilcox-O'Hearn, "Experiences Deploying a Large-Scale Emergent Network", in *Proceedings of the First International Workshop on Peer-to-Peer System (IPTPS) 2002*, Cambridge, MA, USA, Mar. 2002, vol. 2429, pp. 104–110, Springer-Verlag Heidelberg, Lecture Notes in Computer Science.

[3] K. Hildrum, J. D. Kubiatowicz, S. Rao, and B. Y. Zhao, "Distributed Object Location in a Dynamic Network", in *Proceedings of the fourteenth annual ACM symposium on Parallel algorithms and architectures*. 2002, pp. 41–52, ACM Press.

[4] G. Kan, "Gnutella", in *Peer-to-Peer. Harnessing the Power of Disruptive Technologies*, A. Oram, Ed., pp. 94–122. O'Reilly, Sebastopol, CA, 2001.

[5] I. Clarke, T. W. Hong, S. G. Miller, O. Sandberg, and B. Wiley, "Protecting Free Expression Online with Freenet", *IEEE Internet Computing*, vol. 6, no. 1, pp. 40–49, 2002.

[6] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications", in *Proceedings of the SIGCOMM 2001 conference*. 2001, pp. 149–160, ACM Press.

[7] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A Scalable Content-Addressable Network", in *Proceedings of the SIGCOMM 2001 conference*. 2001, pp. 161–172, ACM Press.

[8] K. Gummadi, R. Gummadi, S. Gribble, S. Ratnasamy, S. Shenker, and I. Stoica, "The Impact of DHT Routing Geometry on Resilience and Proximity", in *Proceedings of the SIGCOMM 2003 conference*. 2003, pp. 381–394, ACM Press.

[9] X. Y. Zhang, Q. Zhang, Z. Zhang, G. Song, and W. Zhu, "A Construction of Locality-Aware Overlay Network: mOverlay and Its Performance", *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 1, pp. 18–28, Jan. 2004.

[10] M. Ripeanu, A. Iamnitchi, and I. Foster, "Mapping the Gnutella Network", *IEEE Internet Computing*, vol. 6, no. 1, pp. 50–57, Feb. 2002.

[11] H. Dämfpling, "Gnutella Web Caching System", July 2002, http://www.gnucleus.com/gwebcache/specs.html, accessed 11 March 2004.

[12] S. Saroiu, K. P. Gummadi, and S. D. Gribble, "Measuring and Analyzing the Characteristics of Napster and Gnutella hosts", *Multimedia Systems*, vol. 9, no. 2, pp. 170–184, Aug. 2003.

[13] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz, "Handling Churn in a DHT", Tech. Rep., Computer Science Division (EECS), University of California at Berkeley, Dec. 2003, Report No. UCB/CSD-03-1299.

[14] P. Karbhari, M. Ammar, A. Dhamdhere, H. Raj, G. Riley, and E. Zegura, "Bootstrapping in Gnutella: A Measurement Study", in *Proceedings of the PAM 2004 workshop*, Antibes Juan-les-Pins, France, Apr. 2004, Springer-Verlag.

[15] R. Hinden, "RFC 2373: IP Version 6 Addressing Architecture", July 1998.

[16] E. Guttman, "Service Location Protocol: Automatic Discovery of IP Network Services", *IEEE Internet Computing*, vol. 3, no. 4, pp. 71–80, July 1999.

[17] L. Garces-Erice, K. W. Ross, E. W. Biersack, P. A. Felber, and G. Urvoy-Keller, "Topology-Centric Look-Up Service", in *Proceedings of COST264/ACM Fifth International Workshop on Networked Group Communications (NGC)*, Munich, Germany, 2003.