Measuring Large Overlay Networks – The Overnet Example

Kendy Kutzner und Thomas Fuhrmann

System Architecture Group, Universität Karlsruhe (TH), 76128 Karlsruhe, Germany,
{kendy.kutzner|thomas.fuhrmann}@ira.uka.de

Abstract Peer-to-peer overlay networks have grown significantly in size and sophistication over the last years. Meanwhile, distributed hash tables (DHT) provide efficient means to create global scale overlay networks on top of which various applications can be built. Although filesharing still is the most prominent example, other applications are well conceivable. In order to rationally design such applications, it is important to know (and understand) the properties of the overlay networks *as seen from the respective application*.
This paper reports the results from a two week measurement of the entire Overnet network, the currently most widely deployed DHT-based overlay. We describe both, the design choices that made that measurement feasible and the results from the measurement itself. Besides the basic determination of network size, node availability and node distribution, we found unexpected results for the overlay latency distribution.

1 Description of Overnet and Kademlia

Overnet is a popular peer-to-peer filesharing network, currently used by more than 10^6 unique servants (i.e. server-clients). It is based on Kademlia [8], a general peer-to-peer routing algorithm that can be used to implement distributed hash tables (DHT). Its service is similar to that of other popular DHT routing algorithms like Chord [15], CAN [11], or Pastry [14]. In all these DHTs, participating nodes are assigned a unique identifier, usually a 128 or 160 bit long unsigned integer. In Overnet, the ID is 128 bits long, generated randomly (uniformly) during first node startup, and saved permantently for consecutive starts. These identifiers can be viewed as addresses in an overlay network that extends the functionality of the underlying network infrastructure [3]. Unlike classical networks, these overlays are organized in a way that *every* address maps to a node participating in the overlay network. Thus, these addresses can be used for hashing arbitrary keys to the network's nodes. Hence, the term distributed hash table.

With Kademlia, this address-to-node mapping assigns each address the closest node where closeness is defined by the XOR-metric, i. e. $d(a, b) = a \oplus b$. This metric has the benefit of symmetry, i. e. when node B is close to node A, then also node A is close to node B.

In order to provide routing in this address space, Kademlia nodes have to maintain overlay links to other Kademlia nodes. To this end, all nodes have so-called buckets each holding k different *(Kademlia-address, IP-transport-address)* pairs representing these overlay links. The assignment of pairs to the buckets is based on Kademlia-address prefixes: For each (shortest) prefix that is *not* a prefix of the respective node's address, there is one bucket. As an example, a node with ID 110100 would maintain a bucket for nodes whose addresses are starting with 0, 10, 111, 1100, 11011 and 110101. (Clearly, some of these buckets will be empty, since typically for large prefixes no appropriate overlay links are available.) For routing, a message then simply is forwarded to one of the peers from the longest prefix bucket.

Overnet employs this to create a filesharing network. We do not address any of the aspects associated with that particular use, but rather study the properties of the Overnet network resulting from this general overlay mechanism. Our results can therefore be expected to reflect fundamental aspects of such large overlay networks.

This paper is structured as follows: After discussing related work in section 2, we describe our measurement setup in section 3. We especially focus on general issues that are likely to occur with future peer-to-peer protocols, too. Section 4 summarizes our measurement results. Besides the determination of the fundamental parameters of the Overnet network, we report some astonishing findings which we cannot fully explain at the moment. In section 5, we conclude with an outlook to future work.

2 Related Work

Although there already is a multitude of peer-to-peer measurement studies, they rarely explore the global view of a large scale overlay network as seen from inside. Many studies are concerned with query patterns (user perspective) and Internet traffic properties (provider perspective), whereas we are interested in the global overlay's performance with respect to node availability and message routing delays (application perspective). E.g. RTT distributions *on the overlay layer* are an important parameter when judging the feasibility of certain peer-to-peer applications, like instant messaging, distributed filesystems, etc. To the best of our knowledge, no sufficiently significant measurement study of this property exists, so far.

The measurement study that comes closest to ours probed node availability in Overnet [1]. However, at that time Overnet seems to have been significantly smaller than today. (Only about 40 000 concurrent nodes were discovered there. We saw $4-5$ times as many.) Moreover, the design of that measurement did not aim at gathering the global picture of the network. (They probed only a small subset of the discovered nodes.) Especially, the authors did not measure application layer latency. [7] address the problem of overlay-layer latency. But their measurement included only 25 nodes. [9], too, mentioned that problem, but do not provide any measurements.

[4] reviews many measurements of interarrival times, transfer times, burst sizes and burst lengths. The authors found some evidence for heavy-tailed distributions in burst sizes, but less evidence for such a distribution in interarrival times. That result is somehow expected in the light of the well-studied self-similarity in the Internet. Our finding of a power-law RTT distribution on the overlay layer, however, is unexpected from these earlier Internet results. It seems to be in fact a genuine overlay effect.

3 Measurement Setup

The aim of our measurement was twofold: On the one hand, we wanted to get an up-to-date picture of a large overlay-network. On the other hand, we wanted to gain experience in measuring such large networks in general. We focused our measurements on the distribution of the overlay nodes in the underlying network and the round-trip-times (RTT) as experienced by the overlay application. The content (=files) users are sharing with Overnet were outside the scope of this study.

Such a measurement study requires a large base of overlay nodes capable of constantly probing the network. We used PlanetLab [2,10] to host the probing instances. This gave us more than 300 points in the Internet from which we could probe the Overnet network. Special care had to be taken to not degrade PlanetLab's network connectivity by our measurement traffic. Conversely, we had to take precautions to keep measurement artifacts caused by other PlanetLab users as small as possible.

3.1 The probing application

For our measurements we implemented a very reduced version of an Overnet servant. As described above, the search for nodes close to a given key is an elementary operation in Overnet/Kademlia. We used exactly this functionality to probe the network for nodes and at same time measure the RTT to these nodes. Hence, our measurement application could only send messages of type *search*, which Overnet servants use to discover other peers. The respective answer, a *search_next* message, then contains pointers to other servants in the network. These pointers are *(Kademlia-address, IP-transport-address)* pairs which could be used to iteratively search for more servants in the network.

The probing application maintained a ring buffer of all known Overnet clients. It continiously sent a *search* message to each of them. Every received *search_next* message was parsed and the sender of that message was marked alive. The round trip time (RTT) between sending and receiving was recorded. All previously unknown Overnet Clients from the received message were added to the ring buffer. During this process, addresses from special purpose IP ranges ([13,12,5]) were excluded. Also sites which previously complained about traffic from PlanetLab nodes were not added. The full source code of the application is available from the authors on request.

Our probing application employs two complementary probing modes. It queries either for

- the address of the queried node itself. This makes a node respond with a list of all its neighbors in the overlay. This mode ensures that all nodes have a chance to be found.

or

- a randomly generated bit string. Although the queried node is probably not very good at answering that query (it is not close in Kademlia terms), only this mode of querying ensures that the whole overlay address space will be covered. Moreover, this keeps the different probing applications from following the same trail through the network and thereby allows a simple parallelization of the measurements.

The queries were tailored so that answers fit in a 1400 octet UDP message: The above mentioned *(Kademlia-address, IP-transport-address)*-pair is encoded in 23 octets, and every probe sent queried for 60 such pairs. The Overnet message transporting these pairs has an encoded size of 19 octets.

A local Overnet node was continiously running the original Overnet P2P file sharing program. It's transport address (i. e. IP-address and port number) was hard-coded in the probing application and it was used as bootstrapping node. Even all probing applications started from the same bootstrapping node, by querying for random bit strings, their paths into the Overnet network quickly diverged.

3.2 Flow Control and Congestion Avoidance with UDP

Overnet uses UDP as transport protocol, i.e. there is no inherent flow and congestion control mechanism. An interesting problem was thus how to do flow and congestion control without modifying the given Overnet protocol. To this end, we reverted to two simple mechanisms:

1. Probes were sent either after a fixed timeout t or immediately after receiving a response to the previous probe. This ensured a rate bound by the timeout and the RTT value. The timeout was set to one second.
2. Results were constantly sent back to our central data collection facility using a TCP connection. If this connection stalled due to congestion on the PlanetLab link (a frequent event), we stopped further probing until this reporting traffic was flowing again.

The latter made our probing application TCP-friendly, but only on the probing site local links. Rate limitation for the probed Overnet servants didn't have to be considered in particular, since we probed an extremely large set of nodes from only 300 measurement nodes. Thus the rate there was at least three orders of magnitude below that at the probing sites.

Nevertheless, the probing application performed an emergency halt as soon as one of the following conditions became true:

– The number of nodes that responded is smaller than 10 percent of the probes sent.
– All known nodes have been tried at least once, regardless of whether they answered or not.
– The application ran out of credits. For every probe received the credit was set to c_{max}. For every probe sent, the credit was decreased by one. During deployment, we initialized the credit value to 500 and used a c_{max} of 100.
– The application ran for more than one hour.

The motivation behind these stop rules is that we wanted to prevent the probing application to overload the local network when our security measures descibed above failed to work. One scenario where this could occur is a local admistrator installing a firewall to block the probing application during runtime of the prober. Without these additional measures the prober would run forever.

3.3 Deployment

As stated above, we used PlanetLab as basis for our measurements. During our preparations we encountered a few minor issues that could be solved by software upgrades in the respective operating systems, but that we nevertheless believe worth reporting here, since they shed light on potential problems that can occur with future similar measurement studies.

We started our development with Linux 2.4.26/i386. Its protocol stack keeps a table with IP destinations. This table is, e.g., used to cache routing decisions and to store results of path MTU discovery. Since our application uses many different destination addresses, this table filled more quickly than Linux could garbage collect this table. Thus the maximum probing rate we achieved with the above platform was only about 300 different IP destinations per second. (Meanwhile, kernel 2.6.7 solved this issue.)

A similar limitation was encountered at the faculty firewall whose connection tracking mechanism did not clean its table quick enough for our measurements. Unlike with the Linux kernel problem, here, the firewall machine just crashed and had to be disabled for further measurements. Although our actual measurements deployed with PlanetLab did not use such a high rate as we did in the initial trials from within the faculty network, the problems came up there, too. At the University of Waterloo, Canada, e.g., the large number of destination addresses triggered a security alarm, and we had to exclude this site from our experiment.

From this experience we conclude that for further deployment of peer-to-peer techniques, firewall and intrusion detection system manufacturers need to modify their software, in order to be able to cope with peer-to-peer traffic. Note that we consider traffic with a large number of different IP destinations an inherent characteristic of peer-to-peer systems, not an artifact of our measurements. Moreover, even 300 messages per second is not a high rate.

4 Results

The measurement described in the previous section was run continiously for two weeks in July 2004. During that time the system was able to perform 482 912 462 RTT measurements and collect 2 943 223 different Kademlia addresses (with 9 549 043 different IP addresses) of the participating nodes (The difference between these numbers is explained below in section 4.2). Typically, between 200 000 and 265 000 Overnet nodes were concurrently present worldwide.

4.1 Performance of the probing application

Figure 1 shows the number of successfully received reports plotted over the rank of the respective PlanetLab node. Most nodes received almost 2 million reports each, corresponding to a rate of about 1.6 reports per second. Not all PlanetLab nodes were available all the time, hence the differences in figure 1. Remarkably, about 30 nodes only have ca. 20 000 reports. These nodes are part of the *Internet2* [6] and have limited connectivity to the rest of the Internet, so that they weren't able to collect as many reports as the rest of PlanetLab.

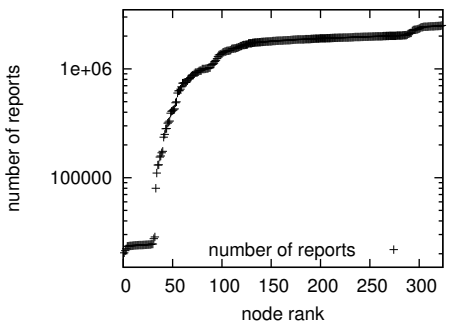

Figure 1. Rank plot of reports per PlanetLab node

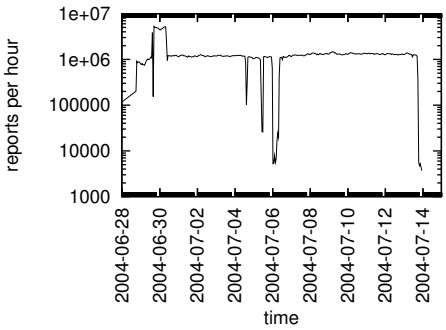

Figure 2. Reports per hour

As stated above, the whole experiment ran for 14 days at the beginning of July 2004. Figure 2 shows the number of reports the entire measurement system received per hour. Typically, the system ran very smoothly at its design rate of 1 million reports per hour. Taking the mentioned number of an average of almost 250 000 Overnet node, this means that each Overnet servant was probed on average every 15 minutes.

In the initial phase, before July 1st, we experimented with different rate limits and adjusted them in accordance with feedback from the PlanetLab community. Except for one site (Waterloo, Canada) all sites were happy with the traffic the experiment created. (As explained above, the problem was not the bandwidth

usage of this experiment but its address cache usage in the firewalls and intrusion detection systems.)

Arround July 5th, we experienced local network problems which kept us from storing the reports in our central data collection facility. Hence the seemingly lower rate.

4.2 Usage patterns

While the above analysis was concerned with the performance of our measurement application, the collected data reveils also temporal effects caused by the Overnet users.

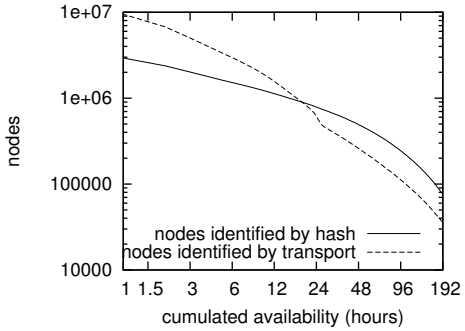

Figure 3. Cumulated Node Availability

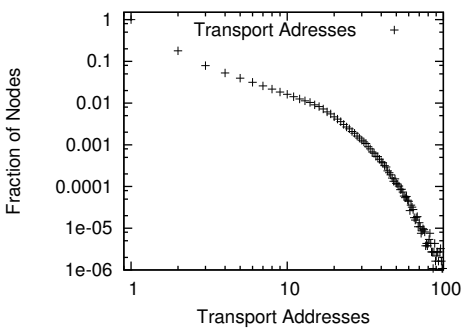

Figure 4. Transport Addresses seen per Hash Value

Figure 3 shows the availability of servants. In this double-logarithmic plot, the number of nodes is plotted against the nodes' observed cumulated availability. According to our measurement scheme we cannot determine uninterrupted uptime, but only availability defined as time-interval during which we did not discover an outage. To prevent packet loss from spoiling the result, we define a node available if we receive at least one response within an hour of measurement. We chose one hour as sampling interval because the probing application were restarted after that time.

The plot shows two lines. The dashed lines corresponds to the transport addresses (i. e. IP-address and port number pairs), the normal line to the Kademlia overlay addresses. This distinction is necessary because of the use of dynamic IP addresses. I.e. an Overnet node may come and go, each time bearing a different transport address. Accordingly the transport address line is skewed against the line showing the availability of overlay addresses. Even though a transport connection breaks, e.g. by a modem hang-up, the node may be constantly available through an immediate re-dial. [1] calls the differences created by distinct ways to identify a node the *aliasing effect*.

This skew effect that is caused by the typically home users with dial-up connectivity to the Internet is also clearly demonstrated by the curve's elbow at 24 hours which is caused by those DSL providers who cut connections after 24 hours. Nodes behind such an access cannot have uptimes beyond 24 hours. Thus the transport address uptime actually is the combination of a 24h limited distribution and an unlimited distribution.

Figure 4 shows the number of transport addresses seen per Kademlia address. Most nodes have static IP addresses, but others appeared with more than 100 different addresses within our measurement period.

Cum grano salis, the up-time is power-law distributed. 50% of the nodes are available for up to (more than) 6 hours. 18% of the nodes were available for more than 48 hours. It is unclear whether there is a cut-off on the order of one or two weeks. (This could not be decided by a two week measurement.)

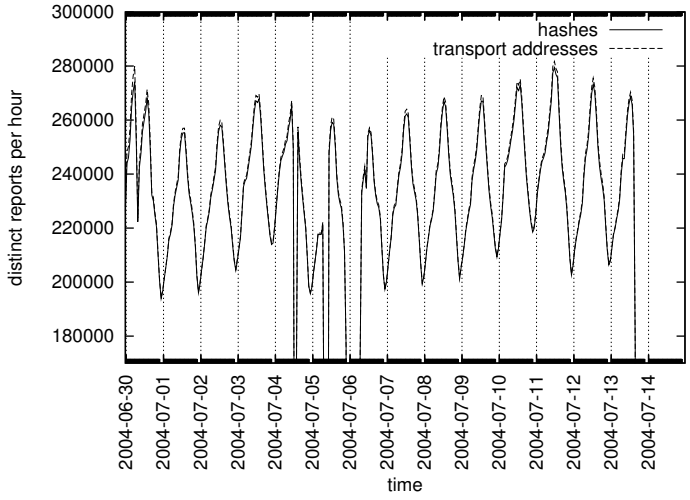

Figure 5. Number of Hash Values and Transport Addresses seen per hour

Figure 5 shows the number of distinct nodes available at a given hour (time in GMT). There is a strong diurnal pattern with respect to both the transport and overlay addresses, but no significant difference between the two types of addresses. The above mentionend aliasing effect is not visible here because in any given hour, a node is likely to be online with a single transport address.

Figure 6 explains why this pattern is visible even without correction for local timezones. The figure shows that most Overnet nodes are located in Asia. Note the log-scale of the plot. IP addresses were mapped to time zones in two steps. First we used a snapshot of BGP data from `www.routeviews.org` to map the address to an AS number. To map AS numbers to time zones, we utilized the

longitude column of `http://netgeo.caida.org/aslatlong.txt`. Of course this method is far from perfect, but it gives an rough overview of node distribution. One source of error is that all addresses are mapped to the headquarter of their origin AS. Furthermore the routing table used did not contain entries for all prefixes. But in summary we were only unable to locate 796136 (8.3%) of all addresses.

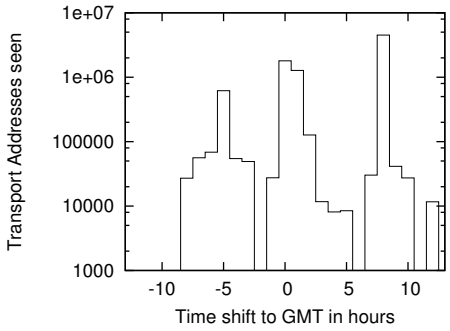

Figure 6. IP Addresses per time zone

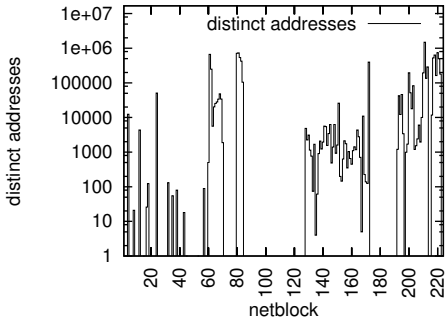

Figure 7. Number of distinct transport addresses per /8 netblock

4.3 Overnet node distribution

During the entire measurement over 9 million distinct transport (=underlay) addresses were encountered. The difference between the number of overlay and underlay addresses is again due to the presence of machines with dynamic IP addresses.

Figure 7 shows the distribution of the encountered IP addresses over the IP address space. (Addresses were binned into /8 prefixes.) One can clearly see the large currently unassigned chunks between 89/8 and 126/8, as well as between 173/8 and 190/8. The usage of the remaining net-blocks shows huge deviations. (Note the logarithmic axis of ordinates.) The most densely populated regions remain only about one order of magnitude behind the theoretical maximum (upper edge of the plot box). Typically, the regions bear between 100 and 100 000 Overnet nodes.

4.4 Round-trip times

Figure 8 depicts the distribution of experienced round-trip times, and figure 9 the CCDF thereof. The resolution for the binning of values was 10 milliseconds. Note that these are application-level round-trip times, i.e. they include all processing done in the underlying operating systems as well as the time the application

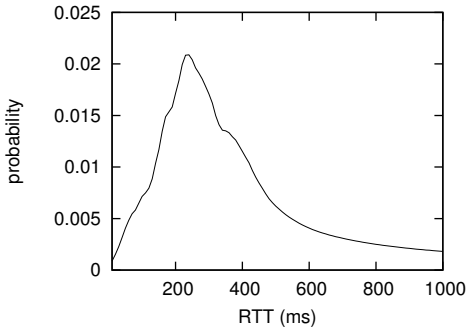

Figure 8. Distribution of RTTs

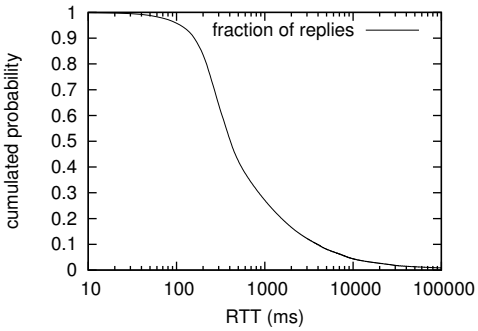

Figure 9. CCDF of RTT distribution

needs to create the reply. The peak in this graph is at arround 250 ms, which appears to be a typical round trip time between two Overnet applications across the Internet.

Half of all replies are received within 410 milliseconds. But it took 8.71 seconds until 95 percent of the replies arrived. 1.15 percent of the replies arrived after one minute and later. A few valid replies even arrived after more than one hour! Such extremes are typical for power-law distributions. The double-logarithmic plot of the results (figure 10) in fact suggests that round-trip times above 250 milliseconds follow a power-law distribution, with no cut-off being visible after two weeks of measurement. Only a longer measurement could reveal where the cut-off appears.

This result is astonishing since we do not know of any obvious reason that would cause a rather sharp on-set of a pretty exact power-law distribution that spans over at least five orders of magnitude. We can only speculate about over-dimensioned application layer message queues, processes delayed by swapping the system memory, and scheduling peculiarities.

If analyzed in detail, the RTT measurement provides more surprises. E.g., there is a distinct dip in figure 8 at arround 320 milliseconds for which we do not have an explanation at the moment. The same holds for the three small bumps between 2.5 and 50 seconds in figure 10. We can again only speculate about interference of the Overnet application's message queue with the underlying operating system.

5 Conclusion and Outlook

In this paper, we presented a measurement study of the Overnet network. With the help of more than 300 PlanetLab nodes, we probed every hour 1 million Overnet servants for a period of two weeks. We described both, the design of our measurement application and the problems we encountered when setting up the measurement. These problems are rooted in the fact that although we measured at a very low rate (1.6 messages per second for the probing nodes, 4 messages

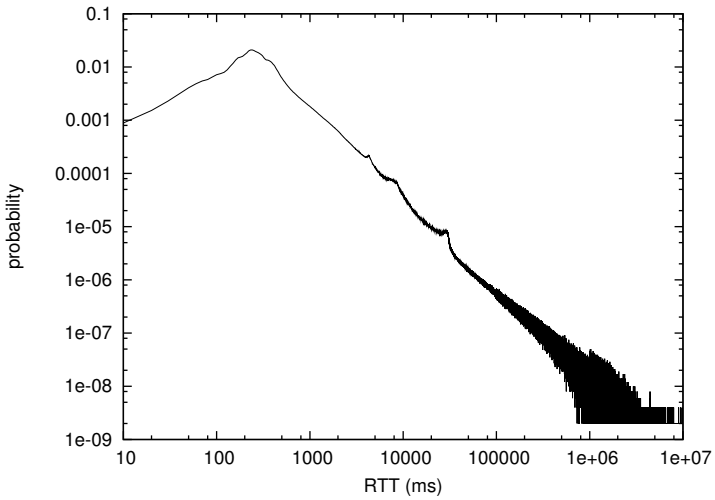

Figure 10. Distribution of RTTs

per hour for the probed nodes), we communicated with a considerable number of destinations. Since we expect such communication patterns to become more and more common with the spread of more sophisticated peer-to-peer applications, we recommend addressing this problem in operating systems, as well as in firewall and intrusion detection software.

Concerning the measurement results, we determined the fundamental parameters of the Overnet network: network size, node availability, and node distribution. Unlike most measurement studies in the field of peer-to-peer networking, we successfully documented the global view on two weeks in the life of a heavily used file-sharing network. Moreover, we collected more than a total of 480 million RTT measurements from hundreds of measurement sites worldwide. Currently, we are still analyzing that data to localize Overnet nodes with the help of network tomography methods. This analysis is complicated by the fact that our approach was to capture the actual overlay RTTs, not the Internet RTTs typically measured by the *ping* tool. Besides that, we are still investigating the unexpected peculiarities that we found in the RTT distribution.

References

1. Ranjita Bhagwan, Stefan Savage, and Geoffrey Voelker. Understanding availability. In *Proceedings of the 2003 International Workshop on Peer-to-Peer Systems (IPTPS)*, Berkeley, CA, February 2003.
2. Brent Chun, David Culler, Timothy Roscoe, Andy Bavier, Larry Peterson, Mike Wawrzoniak, and Mic Bowman. PlanetLab: An Overlay Testbed for Broad-Coverage Services. *ACM SIGCOMM Computer Communication Review*, 33(3):00–00, July 2003.

3. Curt Cramer and Thomas Fuhrmann. On the Fundamental Communication Abstraction Supplied by P2P Overlay Networks. *European Transactions on Telecommunications*. To be published.
4. Allen B. Downey. Evidence for long-tailed distributions in the internet. In *Proceedings of the First ACM SIGCOMM Workshop on Internet Measurement*, pages 229–241. ACM Press, 2001.
5. C. Huitema. An anycast prefix for 6to4 relay routers. RFC 3068, Internet Engineering Task Force, June 2001.
6. Internet2 Consortium. Internet2 homepage, 2004. `http://www.internet2.edu`.
7. Karthik Lakshminarayanan and Venkata N. Padmanabhan. Some findings on the network performance of broadband hosts. In *Proceedings of the 2003 ACM SIGCOMM conference on Internet measurement*, pages 45–50. ACM Press, 2003.
8. Petar Maymounkov and David Mazières. Kademlia: A Peer-to-Peer Information System Based on the XOR Metric. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 53–65. Springer-Verlag, 2002.
9. Hirokazu Miura and Miki Yamamoto. Content routing with network support using passive measurement in content distribution networks. *IEICE Transactions on Communications, Special Issue on Content Delivery Networks*, E86-B(6):1805–1811, June 2003.
10. PlanetLab Consortium. Planetlab homepage, 2004. `http://www.planet-lab.org`.
11. Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A Scalable Content-Addressable Network. In *Proceedings of the SIGCOMM 2001 conference*, pages 161–172. ACM Press, 2001.
12. Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, and E. Lear. Address allocation for private internets. RFC 1918, Internet Engineering Task Force, February 1996.
13. J. F. Reynolds and J. B. Postel. Assigned numbers. RFC 1700, Internet Engineering Task Force, October 1994.
14. 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, November 2001.
15. Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the SIGCOMM 2001 conference*, pages 149–160. ACM Press, 2001.