Understanding Availability

Ranjita Bhagwan[1], Stefan Savage[2], and Geoffrey M. Voelker[2]

[1] Department of Electrical and Computer Engineering
[2] Department of Computer Science and Engineering
University of California, San Diego

Abstract. This paper addresses a simple, yet fundamental question in the design of peer-to-peer systems: What does it mean when we say "availability" and how does this understanding impact the engineering of practical systems? We argue that existing measurements and models do not capture the complex time-varying nature of availability in today's peer-to-peer environments. Further, we show that unforeseen methodological shortcomings have dramatically biased previous analyses of this phenomenon. As the basis of our study, we empirically characterize the availability of a large peer-to-peer system over a period of 7 days, analyze the dependence of the underlying availability distributions, measure host turnover in the system, and discuss how these results may affect the design of high-availability peer-to-peer services.

1 Introduction

Inevitably, real systems stop working. At some point, disks fail, hosts crash, networks partition, software miscalculates, administrators misconfigure or users misuse. Consequently, the principal challenge in designing highly available systems is to tolerate each failure as it occurs and recover from its effects. However, engineering such systems efficiently requires the designer to make informed decisions about the availability of individual system components.

Webster's dictionary defines *availability* as "the quality of being present or ready for immediate use". However, this seemingly simple definition can conceal tremendous complexity. In traditional data storage systems, the components of interest are devices like disks, SCSI interfaces, and NVRAM buffers, each of which have well-understood statistical failure properties that are usually assumed fail-stop and independent (e.g., redundant disk arrays [5]). In peer-to-peer storage systems, however, the component of interest is the host, whose availability is poorly understood by comparison.

While the failure of individual hardware components can still compromise the availability of a host, a peer-to-peer system designer must also anticipate transient software failures, partial or total communication interruption, and users who join and leave the system independently of their own volition. Moreover, these components can be time varying. For example, a peer-to-peer system may replicate some file F on n machines at time t. However, by time $t + k$ some m machines may be turned off as their owners go to work, returning at some later time. The availability of the hosts is therefore dependent on time of day, and hence, the availability of the file F is a function of time.

Another issue is whether the availability of a host is dependent on the availability of another host, or, whether two host availabilities are interdependent. This issue is important since many peer-to-peer systems [4, 12] are designed on the assumption that a random selection of hosts in a P2P network do not all fail together at the same time.

Consequently, host availability is not well modeled as a single stationary distribution, but instead is a combination of a number of time-varying functions, ranging from the most transient (e.g., packet loss) to the most permanent (e.g., disk crash). Traditionally, distributed systems have assumed that transient failures are short enough to be transparently masked and only the long-term components of availability require explicit system engineering. In peer-to-peer systems, though, this abstraction is grossly insufficient. A new "intermittent" component of availability is introduced by users periodically leaving and joining the system again at a later time. Moreover, the set of hosts that comprise the system is continuously changing, as new hosts arrive the system and existing hosts depart it permanently on a daily basis. A peer-to-peer system designed on this substrate will need to incorporate arriving hosts into it without much overhead, while being able to provide all the functionality it promises to provide in the face of regular departures.

We were motivated to study peer-to-peer host availability in part to shape the design and evaluation of a highly available, wide-area peer-to-peer storage system [15]. A primary goal of the system is to provide efficient, highly available file storage even when the system is comprised of hosts with relatively poor and highly variable availability. Even so, our results can apply to any peer-to-peer system constructed from a similar collection of hosts.

The remainder of this paper examines these issues empirically by characterizing host availability in a large deployed peer-to-peer file sharing system over a 7 day period. We make four principal contributions in this work: First, we show that a minor methodological limitation of previous availability measurements has lead to underestimated P2P host availability [3, 13, 14]. Second, we show that host availability in peer-to-peer systems is a complex enough metric to warrant specification by more than just a fractional value between 0 and 1. Thirdly, we show that, for the purposes of storage placement, the availability of hosts is dependent on time-of-day, but is roughly independent of the availability of other hosts. Finally, we measure the system-wide dynamics of the P2P network by calculating the rate at which new hosts arrive the system for the first time, and existing ones depart. We conclude with a summary of our findings.

2 Related Work

Saroiu et al. [13] and Chu et al. [3] have characterized host availability in the Gnutella [6] peer-to-peer network by actively probing TCP/IP addresses gathered using a Gnutella crawler. Sen and Wang [14] described a similar study using passive measurement of flow-level data from multiple routers across a large tier-1 ISP backbone. Finally, Long et al. [8] measured workstation availability in the Internet using an active probing methodology. Unfortunately, all of these approaches rely on IP addresses to uniquely identify individual hosts over time. This assumption was likely accurate for Long's 1995 study, but modern dynamic address assignment protocols such as DHCP can easily cause the same host to be counted multiple times and thereby underestimate host availability.

Moreover, the growth in the use of NAT boxes can affect the correctness of a TCP/IP address-probing technique.

Weatherspoon et al. [16] analyzed the impact of failure on peer-to-peer block storage systems using a model based on disk mean time-to-failure. In a separate paper, the authors also address the issue of independence between host failures. They mention the need to quantify dependence between failures at a coarse-grained level, i.e., failures due to network disconnectivity, OS version, etc. [17] However, these efforts do not capture the complexity of the peer-to-peer environment – particularly the user-controlled transient outages that dominate host availability in most real systems. The dependence of host availabilities on time and whether there is any interdependence between host availabilities stemming from user behavior have yet to be studied in detail.

3 Experimental Methodology

To study host availability in peer-to-peer systems, we actively measure the availability of hosts in the Overnet file-sharing network [11]. In this section, we describe Overnet and our reasons for choosing it over other popular systems. We then describe our experimental methodology for periodically identifying hosts and subsequently probing them.

3.1 Overnet

The Overnet peer-to-peer file-sharing system is based on Kademlia [9] and is structured on a distributed hash table (DHT). Overnet has no hierarchy, i.e., all hosts have identical functionality. When a new client joins Overnet, it randomly generates an ID for itself. This is the client's ID, and it remains unchanged on subsequent joins and leaves of the client until the user deletes the file containing the client's preferences. For lookup and routing purposes, each host maintains a list of neighbors and their IP addresses. The details of this list can be found in [9].

We measure the Overnet system to model host availability for two reasons:

- Overnet users are identified by immutable IDs, enabling us to track hosts by ID rather than by IP address. Using IDs eliminates the problem of host aliasing via DHCP and NATs.
- Host availability studies need to use a sufficiently widely-deployed peer-to-peer network for measurements to be valid and acceptable. To our knowledge, Overnet is the only widely deployed DHT-based peer-to-peer network.

Note that Overnet is not an open-source system or protocol. As a result, to perform our availability measurements we had to reverse engineer various aspects of the Overnet protocol. Other popular open-source systems would have been more convenient to measure, but they do not meet the requirements of our study (in particular, identifying hosts by unique ID).

3.2 Methodology

Our measurement infrastructure consists of two components; the *crawler* and the *prober*. The crawler provides us with a global view of host membership in the system. The prober allows us to get detailed and fine-grained information on individual host behavior.

Crawler: The purpose of the crawler is to collect a snapshot of the IDs of the active hosts in the network at a particular point in time. It does so by repeatedly requesting 50 randomly generated IDs. These requests lead to the discovery of some number of hosts. The crawler repeats the process by sending requests for the same 50 IDs to all the newly discovered hosts. Thus the crawler uses a recursive algorithm to discover the IDs of hosts in the network by performing lookups to as many hosts as it can find. The crawler runs once every 4 hours to minimize impact on the system as it locates these hosts.

Prober: The purpose of the prober is to periodically probe a set of hosts to determine whether they are available in the system or not at that particular time. It uses a random subset of host IDs discovered by the crawler and probes them every 20 minutes. We chose only a subset of hosts to probe because the overhead of probing hosts limits the frequency at which we can cycle through them. The prober determines the availability of a host with ID I by performing a lookup for I. The lookup succeeds only if the host with ID I responds to the probes. So a successful lookup implies an available host running an Overnet peer.

All our probes look exactly like normal Overnet protocol traffic. This is in contrast to previous measurement studies of peer-to-peer networks [3, 13] that use TCP SYN packets. This strategy has two main advantages. First, it eliminates the problem of IP address aliasing due to the use of DHCP, NAT, and multiple users using the same machine. Second, due to Overnet's lookup procedure, we do not repeatedly send probes to hosts that have been unavailable for long periods of time, thus keeping our experiments non-intrusive.

4 Results

In this section, we present the results of our measurements and the inferences that we can draw from them. First, we summarize the data obtained from the crawler and the prober. Then, we show the effects of aliasing on measured availability. Next, we show how the distribution of host availability varies depending on the time over which it is calculated. We then characterize time-of-day effects on host availability and characterize host availability interdependence. Finally, we measure global host membership turnover in terms of host arrivals and departures in Overnet.

4.1 Experiment summary

The crawler ran every four hours for a period of 15 days, from January 14 through January 28, 2003[3]. Each pass of the crawler yielded approximately 40,000 host IDs,

[3] There was a disruption on of January 21 for roughly 24 hours due to storage problems.

while in a single day, or six passes of the crawler, between 70,000 and 90,000 host IDs were seen

Out of the roughly 84,000 IDs that the crawler gathered on the first day, We chose 2,400 at random for the prober to trace at fine-grained time intervals. . It probed these hosts every 20 minutes for 7 days, from January 15 to January 21, 2003. Out of the 2,400 hosts, 1,468 responded at least once to the probes.

4.2 Aliasing effects

Although only 1,468 unique hosts responded to the prober, a total of 5,867 unique IP addresses responded to the prober. This results in a unique host ID to IP address ratio of approximately 1:4. Clearly, host IP address aliasing is a significant issue in deployed peer-to-peer systems such as Overnet. The aliasing effects could be due to various reasons. Most likely, the main cause is the use of DHCP. It is common that, when a host leaves the system and joins it at a later time, it does so with a different IP address. NATs also introduce aliasing into the network by making use of private IP addresses for hosts behind them. Another possible cause of aliasing is multiple users using the same machine; they will have different unique IDs, but the same IP address.

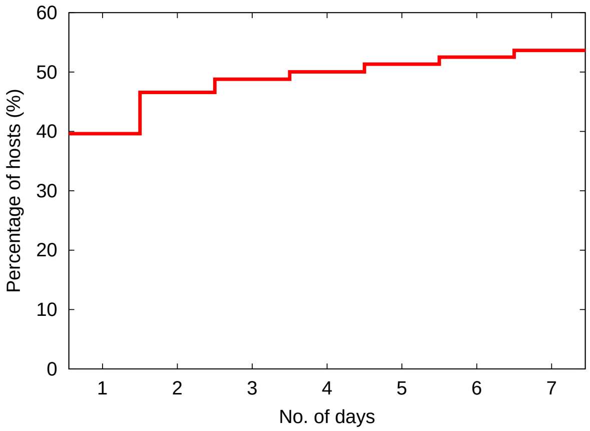

Fig. 1. Percentage of hosts that have more than one IP address across different periods of time.

Figure 1 provides more insight into the nature of aliasing. It shows the percentage of hosts that have more than one IP address over varying periods of time. For example, even over just one day, almost 40% of all probed hosts use more than one IP address. This number increases to 50% after 4 days. These results show that measuring host availability by probing hosts using IP addresses can be very misleading. In fact, 32% of all probed hosts used five or more IP addresses, and 12% used 10 or more! These numbers will only get larger with longer periods of probing. So, probing by IP address does not accurately capture the availability characteristics of the hosts. IP address probing would consider each IP address a new host, thus greatly overestimating the number of hosts in the system and underestimating their availability.

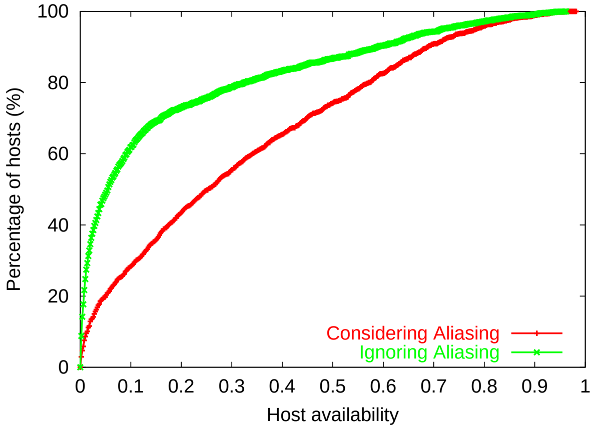

Fig. 2. Host availability derived using unique host ID probes vs. IP address probes.

To evaluate the implications of IP address aliasing on modeling host availability, we derive the host availability distributions for both probing techniques. Figure 2 shows the cumulative distribution of host availability calculated over seven days. We calculate host availability for each host by dividing the number of probes that a host responds to by the total number of probes. The darker line shows the accurate distribution obtained by using host IDs to identify individual hosts, while the lighter line shows the distribution calculated by using the first IP address that was seen for each host ID; the lighter curve is reminiscent of the availability curve in the popular Gnutella study [13]. There is a significant difference between the two curves, with the IP address-based calculation greatly underestimating host availability.

The difference in these distributions demonstrates the extent to which using IP addresses to identify hosts is inaccurate. For example, the darker curve shows that 50% of hosts have availability 0.3 or less, but, if we had used the IP address-based probing denoted by the lighter curve, we would conclude that 50% of all hosts have availability 0.07 or less. Using IP addresses would thus understimate availability by a factor of four. If we had used the IP address methodology to parameterize the design of a highly available storage system, we would make more replicas of files than required to compensate for the apparently low availability and waste storage space. For example, in one model of peer-to-peer file availability [2], the number of file replicas required to maintain a 99% file availability given a mean host availability of 0.07 is five times the storage overhead compared to the number of replicas required given a mean host availability of 0.3.

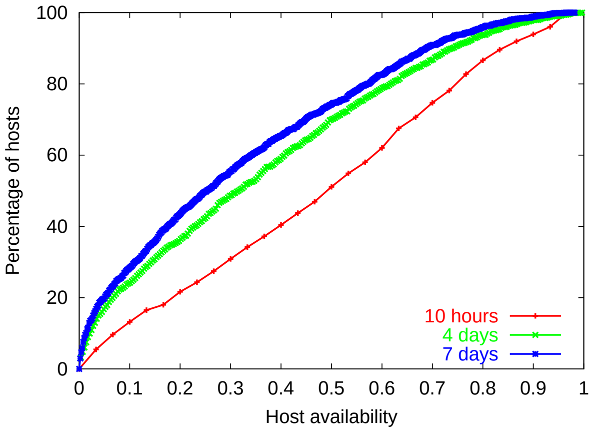

Fig. 3. The dynamic nature of the availability distribution. It varies with the time period over which availability is calculated.

4.3 Host availability

We calculated host availability in Figure 2 over seven days, the entire period of our active probe measurements. However, the period of time over which host availability is calculated can change the distribution. To determine the extent of this effect, we varied the time period over which we calculate host availability. Figure 3 shows the results of this experiment. Over a period of 10 hours, the distribution curve is slightly concave, while for a period of 4 days, the distribution curve becomes convex. Over 7 days, the convexity of the curve increases. And we suspect that with longer periods of measurement this will only increase. Put differently, the distribution curve moves more and more to the left as the period over which availability is calculated increases. This stems from the fact that we are probing the same hosts over the entire period, and the longer the period of time, the greater the chances of a host being unavailable. We are continuing our probing measurements for longer periods of time to validate this hypothesis.

The implication is that, when using an availability distribution to characterize hosts in the system or simply using a fractional value to reflect mean host availability in models of system behavior (e.g., [4]), one needs to also specify the period of time over which the availability measurement holds. Also, the fact that host availability decreases over longer periods of time motivates the need for periodic file refreshes (redistributions, or re-insertions) in the system.

4.4 Time-of-day effects

Next, we characterize the effect of time-of-day on host availability. To do this we need to see how the host availability pattern varies with local time, where local time is based on the geographic location of each host. To calculate local time for each host, we use CAIDA's Netgeo tool [10] to determine the longitude of the host using its current IP address at the time at which it was probed. We then map host longitude to a local time zone.

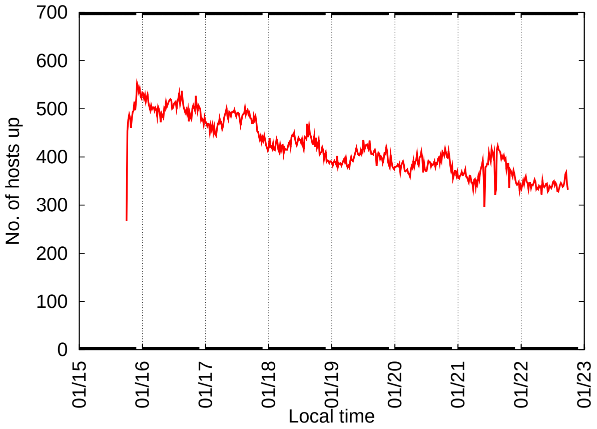

Fig. 4. Diurnal patterns in number of available hosts.

Figure 4 shows the number of available hosts as a function of local time at the hosts' geographic location. The ticks on the x-axis correspond to midnight on the days that are labeled, and this applies to all following time-series graphs. As with other studies of peer-to-peer systems, the graph shows a diurnal pattern [3]. The difference between the maximum and minimum number of available hosts in a single day is roughly 100.

We also found that on average, there were 9413 host joins and leaves per day, or 6.4 joins and leaves per host per day. This figure is considerable, given that the number of hosts that were probed and responded to probes were only 1468. In a system such as CFS, which actively replicates objects immediately when it learns of a host join or leave, this could cause a large amount of overhead in terms of the amount of data transferred between hosts.

The other feature to notice in this graph is the steady decrease in the total number of hosts that are available over subsequent days, which was reflected in our availability distribution measurement. Although limited by a short trace duration, the trend indicates a decay of about 32 hosts per day. The fact that there is a steady decay in the number of hosts with time indicates that in a system such as Oceanstore [7], frequent and periodic file refreshes are required to maintain high file availability.

4.5 Host availability interdependence

The diurnal pattern indicates that availability varies with time-of-day. At non-peak hours a number of hosts become unavailable. Most structured peer-to-peer storage systems, e.g., [4, 9], assume that this happens with very low probability, failing which, objects stored in the system could be lost forever. To our knowledge, this is the first study to investigate the extent that this assumption holds.

We characterize the dependence between every host pair using conditional probabilities. Consider two hosts X and Y. We need to determine the conditional probability of Y being available given that X is available for a given time-of-day t. Call this value P(Y=1/X=1). If this is equal to the probability that Y is available whether or not X is available, or P(Y=1), then X and Y are independent. If independent, then the availability of X at time t does not imply anything about the availability of Y at that time.

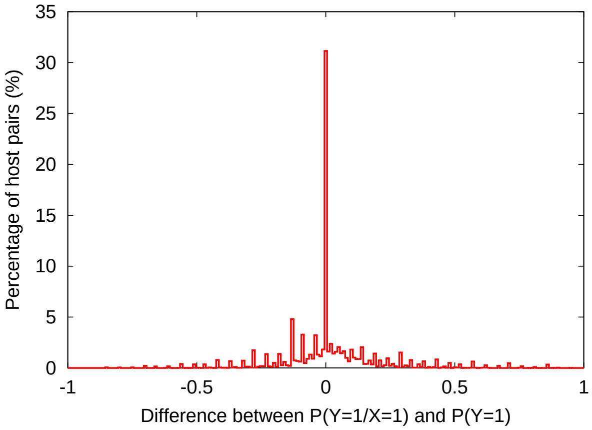

Fig. 5. Probability density function of the difference between P(Y=1/X=1) and P(Y=1).

We calculated P(Y=1/X=1) and P(Y=1) for every host pair from our empirical data for each hour in the trace. Figure 5 shows the probability density function of the difference between these two values. The graph shows that more than 30% of all pairs have 0 difference. Further, 80% of all host pairs lie between +0.2 and -0.2, indicating that there is significant independence between host pairs. So if we were to pick a small subset of hosts randomly, it is highly unlikely that the availabilities of all of them are strongly dependent on each other, even though each may show a strong correlation with time of day. For example, in CFS, the size of this subset is 6, while in Kademlia, it is 20. The probability of all these hosts failing together would be very low.

4.6 Arrivals and departures

Host turnover is important for peer-to-peer systems that rely upon long-term host membership. For example, archival peer-to-peer storage systems like Oceanstore use a high degree of redundancy to mask host failures and departures over time. The rate of host turnover fundamentally determines the rate at which the system must refresh and restore the redundancy in the system to maintain file availability [2, 16], and the overhead that this process entails.

To characterize host membership turnover in Overnet, we would like to determine the rate at which new hosts enter the system for the first time (*arrive*) and the rate at which existing hosts leave the system permanently (*depart*). Note the distinction with host joins and leaves, which refer to intermittent disconnections of hosts from the system. We estimate arrival and departure rates in Overnet using the 15-day crawler trace of active hosts. We consider the first occurrence of a host ID in the trace as an arriving host, and the last occurrence of a host ID as a departing host.

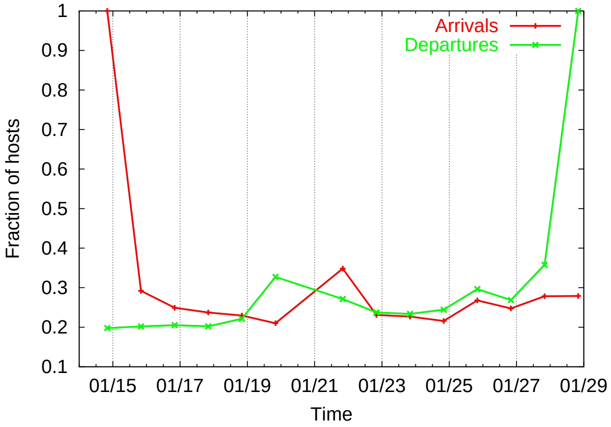

Fig. 6. New host arrivals and existing host departures in Overnet as a fraction of all hosts in the system (approximately 85,000 during this period). The high values at the beginning and end of the period are artifacts of starting and ending the trace.

Figure 6 shows host arrivals and departures as a fraction of the number of active hosts in the system for each day in the trace. For perspective, during this period the crawler found that roughly 85,000 hosts in Overnet were active each day. From the graph, we see that Overnet has a significant degree of turnover. Each day, new hosts never seen before in the trace comprise over 20% of the system (or roughly 17,000 hosts/day). At the same time, existing hosts are departing at roughly the same rate. As a result, the overall size of Overnet stayed constant over our trace period. Since our trace is only 15 days, though, these results only capture short-term turnover. We are continuing our trace to capture long-term turnover as well.

5 Summary

In this paper we studied several characteristics of host availability in the Overnet peer-to-peer file sharing system, and discussed the implications of our findings on the design and operation of peer-to-peer systems. We found that IP address aliasing is a significant problem in these systems, and that measurements according to host IP address significantly underestimate peer-to-peer host availability. We also argue that availability is not well-modeled by a single-parameter distribution, but instead is a combination of two time-varying distributions: (1) short-term daily joins and leaves of individual hosts, and (2) long-term host arrivals and departures. In our Overnet trace, both behaviors significantly impact host availability. For a given set of hosts probed at a fine time granularity, each host joined and left the system 6.4 times a day on average. For a global crawl of all active hosts in the system at a coarser granularity, we also found that host turnover in the system is considerable: over 20% of the hosts in system arrive and depart every day. Peer-to-peer systems must take into account both sources of host unavailability to gracefully and efficiently provide highly available service.

References

1. R. Bhagwan, D. Moore, S. Savage, and G. M. Voelker. Replication strategies for highly available peer-to-peer storage. In *Proceedings of FuDiCo: Future directions in Distributed Computing*, June 2002.
2. R. Bhagwan, S. Savage, and G. M. Voelker. Replication strategies for highly available peer-to-peer systems. Technical Report CS2002-0726, University of California, San Diego, Nov 2002.
3. J. Chu, K. Labonte, and B. Levine. Availability and locality measurements of peer-to-peer file systems. In *Proceedings of ITCom: Scalability and Traffic Control in IP Networks*, July 2002.
4. F. Dabek, M. Kaashoek, D. Karger, R. Morris, and I. Stoica. Wide-area cooperative storage with cfs. In *proceedings of the 18th ACM Symposium on Operating System Principles (SOSP)* , 2001.
5. G. Gibson. *Redundant Disk Arrays: Reliable, Parallel Secondary Storage*. PhD thesis, University of California at Berkeley, 1990. Report UCB/CSD 91/613.
6. Gnutella homepage, http://www.gnutella.com.
7. J. Kubiatowicz, D. Bindel, Y. Chen, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao. Oceanstore: An architecture for global-scale persistent storage. In *Proceedings of ACM ASPLOS*, 2000.
8. D. Long, A. Muir, and R. Golding. A longitudinal study of internet host reliability. In *Proceedings of the Fourteenth Symposium on Reliable Distributed Systems*, September 1995.
9. P. Maymounkov and D. Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. In *Proceedings of the 1st International Workshop on Peer-to -Peer Systems (IPTPS'02)*, March 2002.
10. Netgeo - the internet geographic database, http://www.caida.org/tools/utilities/netgeo/.
11. Overnet website, http://www.overnet.com.
12. A. Rowstron and P. Druschel. Storage management and caching in past, a large-scale, persistent peer-to-peer storage utility. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles(SOSP '01)*, 2001.
13. S. Saroiu, P. K. Gummadi, and S. D. Gribble. A measurement study of peer-to-peer file sharing systems. In *Proceedings of MMCN*, 2002.
14. S. Sen and J. Wang. Analyzing peer-to-peer traffic over large networks. In *Proceedings of ACM SIGCOMM Internet Measurement Workshop*, November 2002.
15. Total recall website, http://ramp.ucsd.edu/projects/recall/.
16. H. Weatherspoon and J. Kubiatowicz. Erasure coding v/s replication: a quantitative approach. In *Proceedings of the First International Workshop on Peer-to-peer Systems*, 2002.
17. H. Weatherspoon, T. Moscovitz, and J. Kubiatowicz. Introspective failure analysis: Avoiding correlated failures in peer-to-peer systems. In *Proceedings of the International Workshop on Reliable Peer-to-peer Distributed Systems*, October 2002.