Presence-Based Availability and P2P Systems

Richard J. Dunn, John Zahorjan, Steven D. Gribble, Henry M. Levy
University of Washington
E-mail: {rdunn,zahorjan,gribble,levy}@cs.washington.edu

Abstract

The availability of a P2P service is a function of the individual peers' availabilities, and it is often desirable to estimate how available a particular P2P service will be given the availability of its peers. Prior work in this area has widely used the fraction of time the average peer is available as the basis for this estimate. We show here that this approach has serious drawbacks. We develop a different measure, which we call presence-based *availability, which takes into account the availability of the individual peers.*

Using traces of live P2P systems taken from the literature, we demonstrate that presence-based availability is a more reliable indicator of potential performance than prior methods. We show that our metrics successfully estimate the availability of a P2P file-sharing system. Then, using presence-based measures to make a better estimate of a parameter in a highly-available system, we achieve a 75% decrease in resource usage relative to an existing technique relying on traditional metrics.

1. Introduction

Peer-to-peer file-sharing systems have had enormous impact, dramatically affecting Internet traffic [21] and inspiring researchers to consider peer-to-peer architectures in other problem domains. Recently, P2P systems have been proposed as a novel approach to providing distributed storage services [17, 14, 18, 13, 12]. By harnessing a large collection of individually managed machines, P2P storage systems have the potential to provide a scalable, highly-available storage service while eliminating centralized management.

In contrast to a dedicated centralized system, a P2P system must contend with the fact that its constituent components have a large degree of autonomy. Specifically, the machines that choose to participate in a P2P system are not administered centrally, and may exhibit high heterogeneity in performance, trustworthiness, and availability [20]. In this paper, we focus on the issue of availability.

It is useful in what follows to distinguish among three different measures of availability, independently of the approach used to compute them. These are:

- *peer availability* - the extent to which a single peer contributes to the P2P service, based on the times at which it is online and willing to participate.
- *workload availability* - the average of the peer availabilities across all peers. This is often used as a coarse measure of the difficulty of hosting available P2P services on that peer set.
- *service availability* - the extent to which the P2P system is able to satisfy client requests. Because individual peers, and so the resources they provide, are often unavailable, P2P systems typically employ some form of replication to achieve high service availability.

Peer and workload availabilities are determined by user behavior, and for that reason can be viewed as characteristics of a user population. Service availability is a reflection of the replication strategies of the P2P system, as well as the behavior of the peer population.

In studying P2P availability, many researchers have based their measures of peer and workload availability on the metric traditionally used for managed servers: the fraction of time the server is online. We refer to this as a *time-based* approach. While appropriate for managed systems, where individual server availability tends to be very high and server nodes are distinct from client nodes, time-based availability has shortcomings in P2P systems, where neither assumption applies. When comparing among different peer sets, the time-based measure of workload availability is often used to imply that one set of peers is more available than another, irrespective of service availability. In Section 2, we show that this definition of availability has serious drawbacks that make it unsuitable for this purpose.

In Section 3 we present our approach: *presence-based* availability. Presence-based availability defines the availability of an individual peer in proportion to the number of other clients online at the same time. We then extend the presence-based peer availability measure in a natural way to define workload and service availabilities. These measures address the shortcomings of the time-based measures, which assume independent uptimes and mask the unequal distribution of availability among peers.

We evaluate our work in Section 4 by demonstrating how the presence-based view of availability can help improve P2P system design and operation. Using existing traces of peer availability, we show that our service availability metric provides a good prediction of measured service availability in a simulation of a P2P file-sharing system. We also consider a system design issue: the amount of redundancy required to achieve a specified service availability level. We replicate an earlier study that used time-based service availability metrics, and show that they overestimate the necessary level of redundancy. By using presence-based availability metrics to obtain a better estimate, we are able to achieve the same level of target availability while reducing resource consumption by 75%.

Finally, we survey related work and conclude in Sections 5 and 6.

2. Time-Based Availability

We begin this section by reviewing the time-based availability measures widely used in prior work. We then illustrate a number of shortcomings they have through a set of examples.

2.1 Definitions

The existing, time-based measure of peer availability is simply the fraction of time that a peer is connected to the P2P system [5, 6, 11, 16, 20, 22, 7]. The corresponding workload availability measure is the average peer availability. Both are easily computed from trace data. Let $\mathcal{P}$ be the set of all peers observed, and $\mathtt{P} \equiv |\mathcal{P}|$ the number of peers. Let z_p be the number of connection sessions observed for peer $p \in \mathcal{P}$, $t_{p,k}$ be the length of the kth such session, and $\mathtt{T}$ the length of the trace interval. Then the availability of peer p, U_p, is

$$\mathtt{U_p} \equiv \frac{\sum_{\mathtt{k=1}}^{\mathtt{z_p}} \mathtt{t_{p,k}}}{\mathtt{T}} \tag{1}$$

and the workload availability, $\bar{\mathtt{U}}$, is:

$$\bar{\mathtt{U}} \equiv \frac{1}{\mathtt{P}} \sum_{\mathtt{p} \in \mathcal{P}} \mathtt{U_p} \tag{2}$$

Note that these availabilities are simply averages over time, and that $\bar{\mathtt{U}}$ is cheap to compute, requiring time $O(\sum_{p\in P} z_p)$.

Because service availability is influenced by the design of the P2P system, defining it requires that one make an assumption about that design. Most importantly, service availability must characterize the measures taken by the P2P system to overcome the intermittent availability of the peers. This usually involves making redundant copies of a single object. Typically, a version of "k-of-n" redundancy scheme is used, in which n copies (or, in the case of erasure coding, partial copies) of the object exist, and a peer must retrieve k of these to obtain the complete object.

Service availability is thus the probability over the trace interval that k of the n servers are available, averaged over all possible sets of size n. Past work has widely used the average peer availability $\bar{\mathtt{U}}$ as the probability that any one peer is available. This leads to a straightforward expression for service availability, $\mathtt{U}^{<\mathtt{k,n}>}$, as a function of k and n [7]:

$$\mathtt{U}^{<\mathtt{k,n}>} = \sum_{\mathtt{j=k}}^{\mathtt{n}} \binom{n}{j} \bar{\mathtt{U}}^{\mathtt{j}}(\mathtt{1}-\bar{\mathtt{U}})^{\mathtt{n-j}} \tag{3}$$

2.2 Distributional Shortcomings

While the time-based service availability defined in Equation 3 is simple, it ignores all distributional information. Thus, it is insensitive to any correlations among the uptimes of peers. It is also insensitive to the possibly unequal distribution of uptimes across the peers, depending only on the total. Both of these factors can strongly influence the effectiveness of a P2P system, which ultimately depends on the *simultaneous* connectivity of two or more peers.

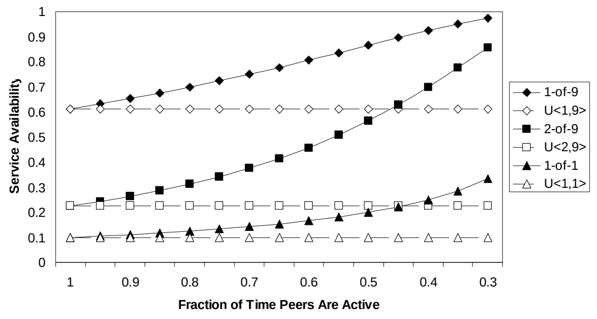


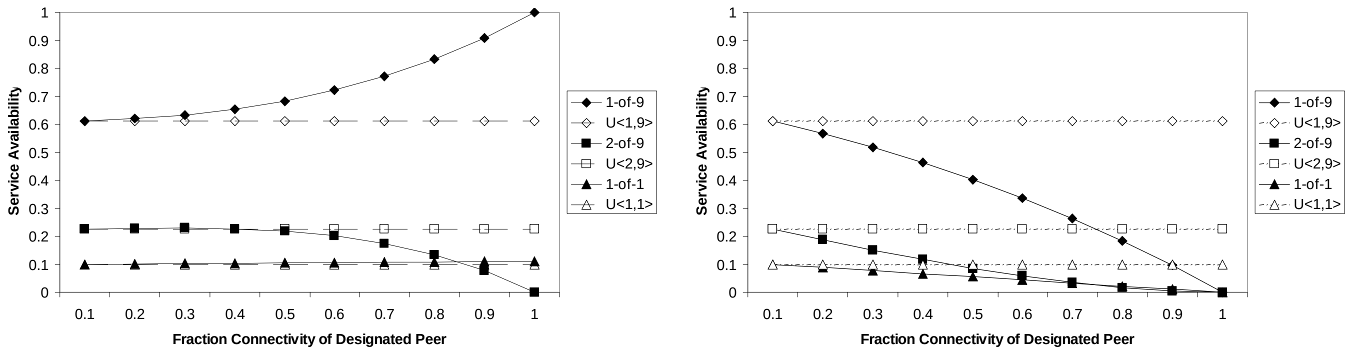
Figure 1. The effect of temporal affinity on availability. *All peers are restricted to be active only during a specific fraction of each day, given by the X-axis. For example, $x = 0.5$ might represent the case when peers are only active from noon until midnight, and no peers are active from midnight until noon.*

We use a simple model to demonstrate each of these factors. In this model, we start with 10 identical peers, each connected 10% of the time, with uptimes chosen at random from the measurement interval. We calculate the actual service availability, as well as $\mathtt{U}^{<\mathtt{k,n}>}$, using a set of values for k and n under a "k-of-n" redundancy scheme. By varying the interval in which uptimes are chosen and, separately, the total uptime of peers, we demonstrate the insensitivity of $\mathtt{U}^{<\mathtt{k,n}>}$ to the two factors described above.

Temporal affinity: Figure 1 shows the effect of temporal affinity, where peers prefer certain times of day for their connected activity (for example, because of diurnal effects). To demonstrate this, we modify the above model so that rather than choose uptimes at random throughout the trace, all activity is restricted to a fraction of the measurement interval indicated on the X-axis. At the left end, users connect with equal probability at any time of day. At the right end, there are only about 8 hours in each day when users choose to connect. Although $\mathtt{U}^{<\mathtt{k,n}>}$ is constant across these scenarios, the probability of obtaining service is strongly affected, with the probability increasing as peers are more likely to be up at the same time. While this model is somewhat unrealistic (it is doubtful that peers would strictly avoid certain times of day), it suffices to show that temporal affinity is a factor in P2P performance, and measurements of actual P2P systems show patterns of user connections that result in this kind of concentration. $\mathtt{U}^{<\mathtt{k,n}>}$ is unable to distinguish among systems with differing time of day preferences.

The distribution of uptimes across peers: Figure 2 shows the effect of unequal activity across peers. We again modify the above model to change the fraction of time that peers are connected. At the left of each graph, all peers are connected exactly 10% of the measurement interval. The X-axis values indicate the connected activity fraction for a single, designated peer, increasing to the right. The activity fractions of the other peers are reduced to hold $\bar{\mathtt{U}}$ constant at 10%. We show service availability to the designated peer and to the other peers in separate graphs.

Once the designated peer reaches about 50% connectivity, further increases have a strong effect on the service availability to the other peers: if only a single peer



(a) (b)

Figure 2. Effects of the distribution of uptimes across peers. *The connectivity of a designated peer increases along the X-axis. The connectivities of all other peers are reduced to hold* $\bar{\mathtt{U}}$ *constant at 10%. (a) Change in service availability as seen by standard peers, and (b) Change in service availability as seen by designated peer.*

is required, service availability increases, because the designated peer is very likely to be available. If two peers are required, service probability drops, as all the peers except the designated one have very low connectivities. Service availability to the designated peer suffers as its own connectivity increases for the same reasons. Clearly, the distribution of activity across peers can have a significant effect on the overall system. However, $\mathtt{U}^{<k,n>}$ is unable to distinguish among systems that differ in the localization of connectivity to subsets of its peers.

2.3 Addressing These Problems

To help account for these effects, we introduce in the next section a new class of availability measures. The basic idea behind the new measures is quite simple: rather than considering only the average behavior over the entire time interval of the trace, define availability of a peer from the point of view of one or more observers. For example, if over some measurement interval whenever peer A is up, peer B is also up, A views B's availability as 100%, even if B is up only 5% of the entire interval. That is, B's *presence*, as viewed by A, is 100%. This change accounts for the two shortcomings described above, since it considers availability as it relates to other peers rather than independently.

In general, we are suggesting the use of additional information to obtain a better estimate of availability. This requires that each peer's availability be gathered, rather than an average, but this is standard practice even though only the average is usually quoted. It may also require additional computation, and this must trade off with the value the additional information yields. We discuss the two aspects of this tradeoff in Sections 3 and 4.

3. Presence-Based Availability

In this section we introduce new, *presence-based* measures of availability. To simplify the presentation, we first outline measures for a "1-of-1" system (the service is available when a single, designated peer is available). We the extend these definitions to consider a general "k-of-n" system, and show that the measures can be computed efficiently.

3.1 1-of-1 Presence-Based Availability

We use $\mathtt{A}^{<1>}_{\{p\}}$ to denote the presence-based availability of peer p with $k = 1$. Intuitively, it is the uptime of p weighted by the number of peers online during each moment of uptime, and normalized by the average number of peers up during the entire measurement interval. The larger the number of peers online, the greater the probability that the peer necessary for use to receive service is online.

Let T be the length of the measurement interval. Let $o_p(t)$, $0 \le t \le \mathtt{T}$, be the "online function" for p, having value 1 at times t when p is online, and 0 otherwise. Let $N_{\mathcal{P}}(t)$ be the number of peers online at time t. Then

$$\mathtt{A}^{<1>}_{\{p\}} \equiv \frac{\int_0^{\mathtt{T}} \mathtt{o_p(t)(N_{\mathcal{P}}(t)-1)dt}}{\int_0^{\mathtt{T}} \mathtt{(N_{\mathcal{P}}(t)-o_p(t))dt}} = \frac{\int_0^{\mathtt{T}} \mathtt{o_p(t)(N_{\mathcal{P}}(t)-1)dt}}{\mathtt{T(P\bar{U}-U_p)}}$$

$\mathtt{A}^{<1>}_{\{p\}}$ is clearly between 0 and 1, reaching its minimum when p is online only at times when no other peer is online, and reaching its maximum when p is online only when all other peers are also online.

We use $\bar{\mathtt{A}}^{<1,1>}$ to denote the presence-based workload availability with $k, n = 1$. Like the time-based measure, we define workload availability in this case to be the average peer availability:

$$\bar{\mathtt{A}}^{<1,1>} \equiv \frac{1}{\mathtt{P}} \sum_{\mathtt{p}\in\mathcal{P}} \mathtt{A}^{<1>}_{\{p\}}$$

In this simplest case, for the service to be available a single, designated peer must be online. We present two variants of service availability that differ in how that designated peer is chosen: uniformly at random, or in proportion to the uptime of each peer. We denote the measure for the former as $A^{<1,1>}_{random}$ and for the latter as $A^{<1,1>}_{time}$.

Beginning with the purely random selection, we have:

$$\mathtt{A}^{<1,1>}_{\mathtt{random}} \equiv \sum_{\mathtt{p}\in\mathcal{P}} \frac{1}{\mathtt{P}} \mathtt{A}^{<1>}_{\{p\}} = \bar{\mathtt{A}}^{<1,1>}$$

That is, each peer is equally likely to be the one required (since it is chosen at random). Then, the availability of a specific peer, as seen from the vantage of the other peers, is precisely what is measured by $\mathtt{A}_{\{\mathtt{p}\}}^{<1>}$. Thus, service availability reduces to workload availability in this case.

A second way to define service availability (in the 1-of-1 case) is to assume that each peer p is the one needed in proportion to amount of time p participates in the P2P system. This variant might be applied to P2P storage systems, for instance, for which it has been observed that the most active peers were also the most likely to have copies of the largest number of objects [16]. In this case we have:

$$\mathtt{A}_{\mathtt{time}}^{<1,1>} \equiv \sum_{\mathtt{p}\in\mathcal{P}} \frac{\mathtt{U_p}}{\mathtt{P\bar{U}}} \mathtt{A}_{\{\mathtt{p}\}}^{<1>}$$

since a peer p has probability $\frac{U_p}{\mathtt{P\bar{U}}}$ of being the one needed. If so its availability as seen by other nodes is exactly $\mathtt{A}_{\{\mathtt{p}\}}^{<1>}$.

3.2 k-of-n Presence-Based Availability

To extend from 1-of-1 peer availability to k-of-n availability requires two changes: considering sets of peers, rather than individual ones, and considering the possibility that only a subset of that set may be needed.

Let $\mathcal{S}$ be a set of peers whose availability we would like to compute. We use $\mathtt{A}_{\mathcal{S}}^{<\mathtt{k}>}$ to denote the presence-based availability of that set, under the assumption that any k members of it are sufficient to consider the set available. To compute $\mathtt{A}_{\mathcal{S}}^{<\mathtt{k}>}$, we must first extend the definition of the online function $o_p(t)$ (Section 3.1). Let $\mathcal{O}_{\mathcal{S}}(t)$ have value 1 if all peers in set $\mathcal{S}$ are available at time t and 0 otherwise. We can compute $\mathcal{O}_{\mathcal{S}}(t)$ using a simple dynamic program. Then:

$$\mathtt{A}_{\mathcal{S}}^{<\mathtt{k}>} \equiv \frac{\int_0^{\mathtt{T}} \mathcal{O}_{\mathcal{S}}^{<\mathtt{k}>}(\mathtt{t})\mathtt{N}_{(\mathcal{P}-\mathcal{S})}(\mathtt{t})\mathtt{dt}}{\int_0^{\mathtt{T}} \mathtt{N}_{(\mathcal{P}-\mathcal{S})}(\mathtt{t})\mathtt{dt}} = \frac{\int_0^{\mathtt{T}} \mathcal{O}_{\mathcal{S}}^{<\mathtt{k}>}(\mathtt{t})\mathtt{N}_{(\mathcal{P}-\mathcal{S})}(\mathtt{t})\mathtt{dt}}{\mathtt{T}(\mathtt{P\bar{U}} - \sum_{\mathtt{s}\in\mathcal{S}} \mathtt{U_s})} \tag{4}$$

where $N_{(\mathcal{P}-\mathcal{S})}(t)$ is the number of peers not in set $\mathcal{S}$ online at time t.

To define workload availability, $\bar{\mathtt{A}}^{<\mathtt{k,n}>}$, we average the peer availabilities of all sets $\mathcal{S} \in \mathcal{P}$ of size n:

$$\bar{\mathtt{A}}^{<\mathtt{k,n}>} \equiv \frac{1}{\mathtt{C(P,n)}} \sum_{\mathcal{S}\in\mathcal{F}_{\mathtt{n}}(\mathcal{P})} \mathtt{A}_{\mathcal{S}}^{<\mathtt{k}>} \tag{5}$$

where $\mathcal{F}_n(\mathcal{P})$ is the set of all size n subsets of $\mathcal{P}$, and $C(\mathtt{P}, n) \equiv \frac{\mathtt{P}!}{(\mathtt{P}-n)!\; n!}$ is the number of such subsets.

As in the 1-of-1 case, we distinguish between two kinds of k-of-n service availability. In the first, the specific set of n peers from which k are required is chosen at random from among all such sets of that size. This type of service availability, $A_{random}^{<k,n>}$, is given by:

$$\mathtt{A}_{\mathtt{random}}^{<\mathtt{k,n}>} \equiv \sum_{\mathcal{S}\in\mathcal{F}_{\mathtt{n}}(\mathcal{P})} \frac{1}{\mathtt{C(P,n)}} \mathtt{A}_{\mathcal{S}}^{<\mathtt{k}>} = \bar{\mathtt{A}}^{<\mathtt{k,n}>} \tag{6}$$

The second choice is to weight server sets containing peers who are connected a large fraction of the time higher than those containing only peers who are occasionally connected. For this case we choose to assign weights that are proportional to the product of the fraction of time each peer in the server set is connected. Doing that, time-weighted service availability is:

$$\mathtt{A}_{\mathtt{time}}^{<\mathtt{k,n}>} \equiv \sum_{\mathcal{S}\in\mathcal{F}_{\mathtt{n}}(\mathcal{P})} \frac{\prod_{\mathtt{i}=1}^{\mathtt{n}} \mathtt{U}_{\mathtt{s_i}}}{\mathtt{G}} \mathtt{A}_{\mathcal{S}}^{<\mathtt{k}>} \tag{7}$$

Here, we use s_i to mean the i^{th} member of $\mathcal{S}$, and $G = \sum_{\mathcal{S}\in\mathcal{F}_n(\mathcal{P})} \prod_{j=1}^{n} \mathtt{U}_{s_j}$.

3.3 Computing presence-based availability

While the idea of presence-based availability provides a nice conceptual framework, it poses a practical problem. As shown in Equations 5-7, these measures involve essentially enumerating all the subsets of $\mathcal{P}$ of size n. Because there are $C(\mathtt{P}, n)$ such subsets, it would appear that computing these availability measures is computationally infeasible. However, there exist polynomial time procedures to compute them exactly. Due to space constraints, we do not fully outline the derivation of these algorithms, but simply present the results.

For $A_{random}^{<k,n>}$, first note that:

$$\mathtt{A}_{\mathtt{random}}^{<0,\mathtt{n}>} = 1$$

Now, let $P_Q(j)$ be the fraction of time over the entire trace that exactly j peers are available. Then we can iteratively define $A_{random}^{<k,n>}$ as:

$$A_{random}^{<k+1,n>} = A_{random}^{<k,n>} - \frac{1}{C(\mathtt{P}-1,n)} \frac{1}{\bar{N}_{\mathcal{P}}} \sum_{j=k+1}^{\mathtt{P}-n+k} j \begin{pmatrix} j-1 \\ k \end{pmatrix} \begin{pmatrix} \mathtt{P}-j \\ n-k \end{pmatrix} P_Q(j) \tag{8}$$

Use of this equation requires calculating $P_Q(j)$. Assuming that the peer session start and stop times are in sorted order in the trace, these can be computed in time $O(\mathtt{Z}^*)$, where $\mathtt{Z}^* \equiv \sum_{p\in\mathcal{P}} z_p$. This is identical to the time required to compute $\bar{\mathtt{U}}$ (Section 2.1).

We can derive an approach to computing $A_{time}^{<k,n>}$ in a manner similar to the analysis of $A_{random}^{<k,n>}$ just given. The result is somewhat more expensive, however, because assigning distinct weights to the server sets means that we must consider not just how many peers are connected at any moment ($P_Q(j)$)but the exact identities of those peers. Though we do not outline the procedure here, suffice it to say that a weighted version of $P_Q(j)$, updated with each trace event, allows us to compute $A_{time}^{<k,n>}$ in time $O(\mathtt{Z}^* n^2)$.

3.4 Summary

In this section, we derived a new class of availability measures that we call *presence-based* availability. These measures take the perspective of individual peers attempting to gain service from other peers in the system, and are specifically designed to take into account the shortcomings with time-based availability outlined in Section 2. Despite incorporating more information than time-based measures, they remain efficient to compute.

4. Evaluation

The metrics developed in the previous sections were specifically designed to handle the problems noted with $\mathtt{U}^{<\mathtt{k,n}>}$, namely its insensitivity to temporal affinity and variations in peer activity. However, these metrics are useful only if they correctly reflect the performance of actual systems. In this section, we validate the usefulness of our metrics by evaluating them on data drawn from traces of existing P2P systems, and show that they yield more accurate and reliable measures and insights than those based on $\mathtt{U}^{<\mathtt{k},\mathtt{n}>}$.

4.1 Trace Data

In order to evaluate our metrics, we require measurements of availability in existing systems. We obtained two traces previously described in the literature:

Kazaa trace. This trace of the Kazaa [1] P2P file-sharing system was described in [16]. The trace was collected at the University of Washington (UW) over the course of 203 days between May 28 and December 17, 2002. This trace recorded only file transfers, using the times when a peer is actively transferring as a proxy for availability, and thus is a lower bound on the true availability in the system. In this paper we are interested only in validating our metric against a real system, rather than analyzing the specific properties of Kazaa users at UW, so using a lower bound of true availability is acceptable. Because of the sheer length of the trace, we consider select pieces chosen from the university's summer and fall quarters (and which are thus comprised of two different user populations).

Overnet trace. The second trace is of the Overnet [2] P2P file-sharing system, as described in [5]. This trace was collected at the University of California, San Diego over the course of 7 days between January 15 and 21, 2003. The availability data is the result of actively monitoring 1,468 hosts chosen at random from a set of 84,000 hosts present in the Overnet system on January 14, 2003. Hosts were probed at a granularity of 20 minutes. The probe used an application-level lookup, and thus hosts must be running the Overnet system to be considered available in the trace.

Table 1 names and summarizes the properties of these traces.

4.2 Service Availability Metrics as a Predictor of Performance

In this section, we evaluate our service availability metrics by comparing them to the measured service availability of an existing P2P file-sharing system. We show that our metrics agree with the measured values far better than traditional measures based on $\bar{\mathtt{U}}$. As our target system, we chose to reimplement the Kazaa simulator described in [16], and leverage it as a model of an existing P2P system. This simulator uses the measured peer availability as well as separate trace of object requests (again described in [16]). When a peer issues an object request, it is satisfied either from another peer (if one with that object is available) or from an artificial peer which holds every object and is always available. The object then becomes available on the requesting peer. The simulator is an idealization of the Kazaa system: all peers are aware of each other and are not limited in bandwidth or storage capabilities.

With these properties in mind, we can create an empirical measure of service availability for the Kazaa simulator which we can compare with our metrics of service availability. First, we give some definitions. A peer is *up* at time t if it is actively connected to the system and available for requests. A peer is *live* at time t if there exists times t_1 and t_2 such that $t_1 < t < t_2$ and the peer is up at t_1 and t_2 (i.e. the peer has entered the system but not yet left completely). Let $S(n)$ be the set of requests such that if $r \in S(n)$, when request r is made exactly n peers currently holding copies of the object sought by r are live. Finally, let $O(r)$ be 1 if a peer with the object requested by r is up when r is made, and 0 otherwise. (Though in the simulator the peer obtains the object from an artificial source, we consider this situation a service unavailability.) Our measure of service availability, which we modified the simulator to compute, is then:

$$\tilde{\mathtt{A}}(\mathtt{n}) \equiv \frac{\sum_{\mathtt{r}\in\mathtt{S(n)}} \mathtt{O(r)}}{|\mathtt{S(n)}|} \tag{9}$$

This is the measured probability that request r is satisfied given that n copies exist at the time.

The values of this empirical measure can be directly compared to those computed by our service availability metrics. We consider the case $k = 1$, as Kazaa allows only whole-file replication. In Kazaa, it has been noted that the number of requests (and correspondingly, the number of objects) from a given peer are correlated with the availability of that peer [16]. Thus, our $A_{time}^{<k,n>}$ metric captures most closely the dynamics of this system.

Figure 3 shows the measured service availability of our Kazaa simulation over a subset of the traces in Table 1. These traces cover two different time periods (summer and fall quarters) and are of differing lengths (1 to 4 weeks). We can see that there is higher service availability in the fall quarter versus the summer quarter, and higher availability in the shorter time periods versus the longer ones.

Figure 4(a) and (b) shows the results of evaluating both $A_{time}^{<k,n>}$ and $\mathtt{U}^{<\mathtt{k},\mathtt{n}>}$ for $k = 1$ and varying n over the same time intervals as above. We can see that the $A_{time}^{<k,n>}$ metric closely matches Figure 3 in predicted value, and the relative orderings of the traces match. $\mathtt{U}^{<\mathtt{k},\mathtt{n}>}$, on the other hand, significantly underestimates the measured availability and, in the case of Kazaa-S1 and Kazaa-F1A, does not match the same relative ordering. These results indicate that $A_{time}^{<k,n>}$ is a better predictor of actual performance than $\mathtt{U}^{<\mathtt{k},\mathtt{n}>}$. (We also carried out this comparison between the four intervals Kazaa-F1A through D. While the results are substantially the same, space limitations prevent us from including those results.)

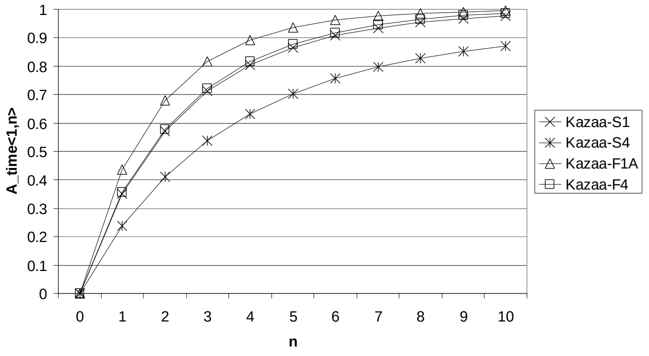
4.3 Service Availability Metrics as a Design Parameter

We also wished to determine the extent to which our metrics could be used to guide design decisions in building highly-available systems. The Total Recall system [7] strives to maintain high availability of files in a managed P2P file system. They achieve this through a 3-step process:

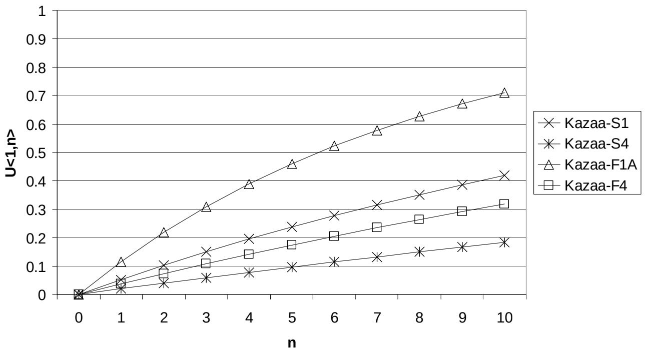
1. A centralized monitor records the availability of peers, using traditional time-based metrics.
2. From the recorded availabilities, the system computes a *replication factor* for individual files. The replica-

Trace Name	Start Date	Length (Days)	# of Peers	$\bar{\mathtt{U}}$	$\bar{\mathtt{A}}^{<1,1>}$
Kazaa-S1	July 26, 2002	7	1596	0.0528	0.0331
Kazaa-S4	July 26, 2002	28	3242	0.0202	0.0254
Kazaa-F1A	October 11, 2002	7	4641	0.1162	0.1132
Kazaa-F1B	October 18, 2002	7	4941	0.1128	0.1149
Kazaa-F1C	October 25, 2002	7	4595	0.0951	0.0874
Kazaa-F1D	November 1, 2002	7	4602	0.0934	0.0893
Kazaa-F4	October 11, 2002	28	8557	0.0376	0.1017
Overnet	January 15, 2003	7	1469	0.3104	0.3156

Table 1. Characteristics of traces used in Section 4. *The first 7 traces are taken from [16], the last from [5].*



(a)



(b)

Figure 4. Calculated values of (a) $A_{time}^{<k,n>}$ and (b) $\mathtt{U}^{<\mathtt{k},\mathtt{n}>}$ (for $k = 1$) over traces

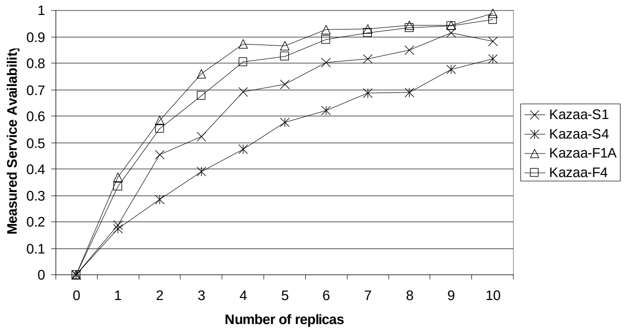


Figure 3. Measured service availability of Kazaa over traces

tion factor determines how many copies of a file are maintained by the system.

3. When a peer holding a particular file leaves the system, a replication manager maintains the replication factor of the file by proactively copying it to a new peer, chosen at random among those peers available at the time the copy is made.

The Total Recall system relies on a measured value of $\bar{\mathtt{U}}$ to calculate a replication factor for files. While they consider both whole-file replication and erasure coding (the particular replication strategy is chosen by policy), we will consider only whole-file replication in this section. Given $\bar{\mathtt{U}}$ and a target availability of A, they calculate the number of copies c needed so that the probability of at least one peer being up to serve the file is at least A:

$$A = 1 - (1 - \bar{\mathtt{U}})^c$$

(Note that this equation is a simplified form of Equation 3, with $k = 1$, $c = n$, and $U^{<k,n>} = A$.) Solving for c:

$$c = \frac{\log(1 - A)}{\log(1 - \bar{\mathtt{U}})} \tag{10}$$

Because of its reliance on $\bar{\mathtt{U}}$, Equation 10 will tend to overestimate the number of copies needed, since it is assumed that each peer is available an equal proportion of the time. When availabilities are skewed, however, making a copy onto a highly available peer will result in a net increase in the availability of the object. Bhagwan, et al. note themselves (see [7], Figure 3) that the empirically measured availabilities of objects in their simulations are much higher than their target availabilities.

We sought to determine whether our metrics might yield a better estimate of the number of copies necessary to maintain a target availability. Because Total Recall uses random selection of hosts *at the time a copy needs to be made*, peers with high availability will tend to be more likely to receive copies of files than peers with lower availabilities, simply because they are more likely to be available when a decision is made. Therefore, the $A_{time}^{<k,n>}$ metric should accurately reflect the dynamics of the system.

To validate whether $A_{time}^{<k,n>}$ produces a better estimation of the necessary replication factor, we reimplemented the Total Recall simulator described in [7]. For a subset of traces from Table 1, we calculated the number of copies predicted by Equation 10 for a target availability of 0.99, the value used in [7]. We also evaluated $A_{time}^{<k,n>}$ for $k = 1$ and numerous values of n, and chose the smallest value of n for which $A_{time}^{<k,n>} \geq 0.99$. We ran Total Recall simulations using both of these values, and measured the corresponding service availabilities. The results of these simulations are summarized in Table 2. C_{TR} represents the

Trace Name	$\bar{\text{U}}$	Target Avail	C_{TR}	Avail(C_{TR})	Xfers(C_{TR})	C_A	Avail(C_A)	Xfers(C_A)	Avail($C_A - 1$)
Kazaa-F1A	0.1162	0.9900	38	0.9994	153,311	9	0.9921	35,822	0.9873
Kazaa-F1B	0.1128	0.9900	39	0.9999	294,854	11	0.9899	82,369	0.9863
Kazaa-F1C	0.0951	0.9900	47	1.000†	324,376	11	0.9913	75,234	0.9879
Kazaa-F1D	0.0934	0.9900	47	1.000	241,385	10	0.9917	50,781	0.9869
Overnet	0.3104	0.9900	13	0.9999	22,513	6	0.9897	10,516	0.9785

Table 2. Results of executing Total Recall simulation on various traces. *The number of copies estimated by Equation 10 (C_{TR}) far outweighs that estimated by $A_{time}^{<k,n>}$ (C_A). Using C_A we are able to achieve the target availability with the benefit of far fewer transfers to maintain those copies. (†All measured availabilities use 4 significant digits. Values of 1.000 have been rounded.)*

number of copies predicted by Equation 10, and C_A represents the number predicted by $A_{time}^{<k,n>}$. Avail(x) represents the measured service availability given one of these values, and Xfers(x) represents the number of object copies made over the course of the simulation.

We first consider the measured service availability that the two estimates provide. Figure 5(a) shows the measured availability across 5 different traces using these values. We derive similar results to [7], in that we find a much higher measured availability than the target availability when using the number of copies predicted by Equation 10. $A_{time}^{<k,n>}$, however, predicts a far smaller number of copies, and yields a measured availability much closer to the target availability. Note also the precision with which $A_{time}^{<k,n>}$ estimates the number of copies. When we execute the simulations with a replication factor one less than our computed value, in all cases the measured availability is below the target.

The smaller number of copies predicted by our measure yields benefits in that fewer transfers are needed to maintain that number. This results in bandwidth savings throughout the system. Figure 5(b) shows the number of transfers required over the traces to maintain the number of copies estimated by the two measures. We can see that the savings is significant; on average, we use 75% fewer transfers to maintain the target availability. Note also that these numbers do not reflect the price of initially seeding the file copies: we save a similar amount of storage space in that case.

Though one might argue in the abstract that higher availability is better, recall that the target availability was a parameter in this system. There is a clear tradeoff between higher availability and higher bandwidth usage; which of these to optimize for should be a conscious decision. In that respect $A_{time}^{<k,n>}$ is superior: it allows for precise control in making this decision.

4.4 Summary

In this section, we used published traces of existing peer-to-peer file-sharing systems to validate the effectiveness of our availability measures. We demonstrated that our measures generate service availability values that correspond well to simulated Kazaa performance, outperforming the traditional measure of service availability. We also demonstrated the our metrics have a practical value: they predict the replication factor needed to maintain high availability in a P2P storage system better than those relying on traditional metrics.

5. Related work

Many system design and measurement studies have considered the problem of availability in the context of homogeneous, single-administration domains, such as Web servers [19], RAID systems [10], clusters [9], or groups of machines in a corporation [8]. These systems are usually measured in terms of the number of "nines of availability," ($\bar{\text{U}} = .9999$ corresponds to four "nines") and typically have such a high degree of availability that temporal affinity and variation in activity are not significant. $\bar{\text{U}}$ (and probabilistic combinations thereof) are reasonable measures of availability in this kind of system.

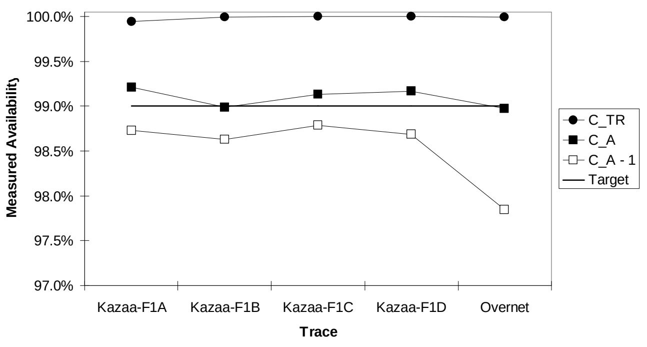
Several projects [5, 11, 20, 22] have traced the availabilities of peers in various P2P file-sharing systems, and noted low availability, high skew in uptimes, and a high turnover rate of peers. Each of these studies reports availability as a distribution of $\bar{\text{U}}$ over hosts, and thus it is difficult to compare the results generated by these studies, since each gathered a trace of different length. Studies of ad-hoc wireless networks and chat systems [3, 4, 15] have found similar behavior and availability distributions among users.

Bhagwan et al. [6, 7] proposed a model for estimating service availability in the face of a replication strategy, and proposed a method to determine the number of copies needed to maintain a desired level of service. However, they relied on traditional measures of availability, and we have shown that our metric does a far better job of estimating the number of copies necessary, resulting in improved resource usage.

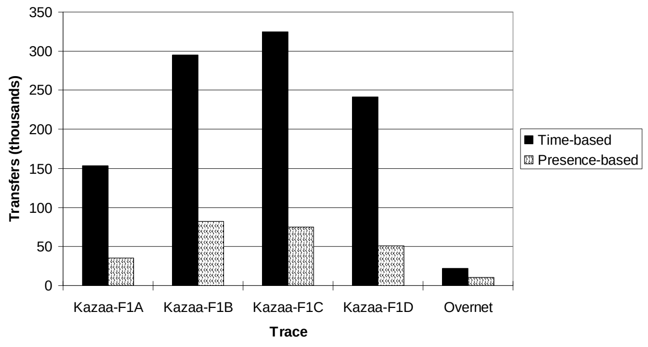
6. Conclusions

Several research projects have investigated the potential of peer-to-peer architectures in distributed storage systems. In this paper, we considered the question of how to calculate a meaningful measure of availability in these systems. We began by demonstrating that traditional availability measures based on notions of average uptime have serious flaws in a P2P environment. We then presented a new family of availability measures based on the idea of measuring availability from the peer's perspective.

We validated our new family of measures using data drawn from traces of real P2P systems. We showed that the estimates of service availability predicted by our metrics matched those measured in a P2P file-sharing system. We also demonstrated that our measures could be used to establish replication parameters in highly-available systems. We showed that the number of copies estimated by our



(a)



(b)

Figure 5. Performance in Total Recall using different replication factors. *(a) Measured availability, and (b) Number of transfers.*

measures was far lower than that of traditional measures, and while achieving the same target availability, we used 75% less bandwidth.

7. References

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

[2] http://www.edonkey2000.com/.

[3] A. Balachandran, G. M. Voelker, P. Bahl, and P. V. Rangan. Characterizing user behavior and network performance in a public wireless LAN. In *Proceedings of ACM SIGMETRICS*, Marina Del Rey, CA, June 2002.

[4] M. Balazinska and P. Castro. Characterizing mobility and network usage in a corporate wireless local-area network. In *Proceedings of iMobiSys*, San Francisco, CA, May 2003.

[5] R. Bhagwan, S. Savage, and G. Voelker. Understanding availability. In *Proceedings of the 2nd International Workshop on Peer-to-peer Systems*, Berkeley, CA, December 2002.

[6] R. Bhagwan, S. Savage, and G. M. Voelker. Replication strategies for highly available peer-to-peer storage systems. Technical Report CS2002-0726, University of California, San Diego, November 2002.

[7] R. Bhagwan, K. Tati, Y.-C. Cheng, S. Savage, and G. M. Voelker. Total Recall: System support for automated availability management. In *Proceedings of the 1st ACM/USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, San Francisco, CA, March 2004.

[8] W. J. Bolosky, J. R. Douceur, D. Ely, and M. Theimer. Feasibility of a serverless distributed file system deployed on an existing set of desktop PCs. In *Proceedings of SIGMETRICS*, Santa Clara, CA, June 2000.

[9] E. A. Brewer. Lessons from giant-scale services. *IEEE Internet Computing.*

[10] A. Brown and D. Patterson. Towards availability benchmarks: A case study of software RAID systems. In *Proceedings of the 2000 USENIX Annual Technical Conference*, San Diego, CA, June 2000.

[11] J. Chu, K. Labonte, and B. N. Levine. Availability and locality measurements of peer-to-peer file systems. In *Proceedings of ITCom: Scalability and Traffic Control in IP Networks II*, Boston, MA, July 2002.

[12] B. Cohen. Incentives build robustness in BitTorrent. http://bittorrent.com/bittorrentecon.pdf, May 2003.

[13] L. P. Cox and B. D. Noble. Samsara: Honor among thieves in peer-to-peer storage. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP)*, Lake George, NY, October 2003.

[14] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Wide-area cooperative storage with CFS. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP)*, Chateau Lake Louise, Banff, Canada, October 2001.

[15] C. Dewes, A. Wichmann, and A. Feldmann. An analysis of Internet chat systems. In *Proceedings of the 3rd ACM/SIGCOMM Internet Measurement Conference (IMC)*, Miami, FL, October 2003.

[16] K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, and J. Zahorjan. Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. In *Proceedings of the 19th ACM Symposium on Operating System Principles (SOSP)*, Lake George, NY, October 2003.

[17] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, 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 the 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, Cambridge, MA, November 2000.

[18] A. Muthitacharoen, R. Morris, T. Gil, and B. Chen. Ivy: A read/write peer-to-peer file system. In *Proceedings of the 5th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, Boston, Massachusetts, December 2002.

[19] D. Oppenheimer, A. Ganapathi, and D. Patterson. Why do Internet services fail, and what can be done about it? In *Proceedings of the 4th USENIX Symposium on Internet Technologies and Systems (USITS)*, Seattle, WA, March 2003.

[20] S. Saroiu, P. K. Gummadi, and S. D. Gribble. A measurement study of peer-to-peer file sharing systems. In *Proceedings of Multimedia Computing and Networking*, San Jose, CA, January 2002.

[21] S. Saroiu, K. P. Gummadi, R. J. Dunn, S. D. Gribble, and H. M. Levy. An analysis of Internet content delivery systems. In *Proceedings of the 5th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, Boston, MA, December 2002.

[22] S. Sen and J. Wang. Analyzing peer-to-peer traffic across large networks. In *Proceedings of the 2nd Internet Measurement Workshop (IMW)*, Marseille, France, November 2002.