

# Technical Report UCSC-CRL-95-16

## A longitudinal survey of Internet host reliability

*Darrell Long and Andrew Muir*

Computer and Information Sciences  
University of California  
Santa Cruz, CA 95064

*Richard Golding*

Storage Systems Project  
Hewlett-Packard Laboratories  
Palo Alto, CA 94304

### Abstract

*An accurate estimate of host reliability is important for correct analysis of many fault-tolerance and replication mechanisms. In a previous study, we estimated host system reliability by querying a large number of hosts to find how long they had been functioning, estimating the mean time-to-failure (MTTF) and availability from those measures, and in turn deriving an estimate of the mean time-to-repair (MTTR). However, this approach had a bias towards more reliable hosts that could result in overestimating MTTR and underestimating availability. To address this bias we have conducted a second experiment, using a fault-tolerant replicated monitoring tool. This tool directly measures TTF, TTR, and availability by polling many sites frequently from several locations. We find that these more accurate results generally confirm and improve our earlier estimates. We also find that failure and repair are unlikely to follow Poisson processes.*

### 1 Introduction

Accurate analyses of fault-tolerance and replication mechanisms depend on an accurate model of the reliability of the systems that make them up. The overall reliability of a replication protocol, for example, depends on the probability that some fraction of the replica sites are functioning when data must be read or written.

There are several important measures used to quantify system reliability, including *time-to-failure* (TTF), *time-to-repair* (TTR), *availability*, and *reliability*. Throughout this study, “failure” is defined in a distributed-environment sense; that is, as an inability to access a host. The term encompasses both hardware and software faults attributable to the host, and can include power failures and scheduled downtime. It can

also be caused by off-site communications failures, ranging from temporary routing failures to problems with the physical communications links. We have not attempted to characterize the causes of failure, though it seems that most failures are brief and are probably caused by software faults or voluntary reboots.

*Time-to-failure* and *time-to-repair* are the distributions of how long a system is available for use, and how long it takes to bring the system back to normal operation after a failure has occurred. These distributions are often summarized by their means (MTTF and MTTR).

Many studies have assumed that failure and repair followed Poisson processes, that is, that the distributions for TTF and TTR were exponential. This assumption is often made more for analytic simplicity than out of a conviction that it is the best model of reality. For example, some analyses of replication protocols using Markov models [Pâris86, Long89] depend on that assumption. We investigate the accuracy of this assumption in §3.1.1 and §3.2.1.

*Availability* is the fraction of time a system is functioning. More precisely, it is the stationary probability of the system being in a state where it can be accessed. For a replication protocol, for example, it gives the probability that a replica site will be functioning when a client starts executing a read or write protocol. Availability can also be used in conjunction with MTTF to derive an estimate of MTTR.

*Reliability* is the (non-stationary) probability that a system will remain constantly available over a fixed time period. Consider a system that functions for one second, then fails, but recovers in a small number of milliseconds. This system would have a high availability, but would not be useful for applications like stock trading services or process control that must remain continuously functioning for extended periods.

Reliability is a more appropriate measure for these applications because it includes duration.

There have been few analyses of host system reliability published, and most of those have been for specific systems. Recent studies include analyses of Tandem systems [Gray85, Gray90] and the IBM/XA system [Mourad85]. It is certain that most companies perform reliability studies of their products internally.

We have measured the reliability of a wide variety of host systems connected to the Internet. For this study, we monitored nearly 1 200 hosts for an extended period, polling them to determine how long they were available. The emphasis on a heterogeneous set of hosts, selected so that its composition should be similar to the overall population of Internet hosts, makes our results more generally applicable than studies specific to one type of system.

We conducted a similar study four years ago [Long91], but the method we used to estimate time-to-failure was biased. Estimates of MTTF were derived by querying systems for their up-time. This was the best information available from the host system, since it is not generally in a position to give its failure time as its dying gasp. As a result, there was a bias towards more reliable hosts which means that the estimate of MTTF may be larger than the true value.

Our new study uses direct measurement of TTF and TTR, rather than an estimate. We measured these distributions by polling each selected host every few minutes. The resulting distributions should reflect all but the shortest periods of failure. Instead of estimating parameters such as MTTF based on a large sample with an unknown distribution, we recorded the actual events (with an epsilon error). Since the quantities are being directly measured, questions such as the governing distribution are less important.

The current study used a distributed, fault-tolerant measurement system (the *Tattler* [Long92]) to reduce bias from the measurement approach. The *tattler* system consists of replicated monitoring sites placed at strategic locations around the Internet. Individual *tattlers* were placed to minimize the amount of shared network so that a failure of a router or a link was unlikely to disable more than one monitor. The *tattlers* replicated their measurements using a weak-consistency group communication mechanism so that even the permanent failure of some monitors would not cause a significant loss of information.

In the sections that follow, we present our experimental method, followed by the results we obtained and our analysis.

## 2 Experimental method

We began the study by selecting a large number of candidate hosts for measurement, and eliminating unsuitable ones. The *Tattler* system then monitored these hosts for an extended period to measure their TTF and TTR. In this section we detail how these steps were accomplished.

### 2.1 Selecting hosts

Our first step was to find a list of at least a thousand hosts that could respond to our polling (using RPC calls to `rpc.statd`, which is common on systems using NFS). We wanted a method that would probably yield a set of hosts statistically similar to the overall population of sites on the Internet.

We began by compiling a list of all visible host names on the Internet. We used the Census tool [Ganatra92], which queries the top-level DNS servers for the names of the hosts at each site and for secondary domain servers, then recursively applies the process to the entire visible Internet DNS name-tree. This resulted in almost three million hosts—nearly ten times the figure we reported four years ago.

This approach does not consider all the hosts connected to the Internet. Some installations, particularly corporations, choose to shield their internal network behind a gateway; those subnetworks are thus invisible to the network at large.

From the census results we chose 15 000 hosts at random. This list was then filtered to ensure that the hosts actually existed, could respond to the poll, and that their administrators would not mind the poll. In the end we obtained a list of 1 170 hosts.

These hosts were uniformly distributed over the name space, and all responded to our RPC polls. Geographic locations with more hosts had a stronger representation than those with fewer. Limiting ourselves to systems responding to the `rpc.statd` protocol biased the list, but we believe that the typical server system on the Internet—the kind of system that is of most interest to us—is likely to include this protocol.

In our previous study we grouped hosts by type, based on information reported in the name service's host information (HINFO) records. In conducting this newer

study, we found that in general these records are not as well maintained now as they were four years ago. Many sites did not provide HINFO records, and many of the ones that were provided did not include enough information to determine system models.

## 2.2 Measuring host systems

There were two ways we could have measured the systems. In our first study, we queried each site to determine the time since its last initialization and used this to estimate MTTF, and separately estimated overall availability. In this study, we chose to take direct measurements of the distributions of TTF and TTR by repeated polling each site for an extended period.

The first method has the advantage of requiring only a single query for each host, but the results are not as accurate as those of the second. Randomly sampling the length of time since the last system initialization, which we used in the first study, is distinct from sampling the length of time between initialization and failure. Sampling system up-time reports results in a skewed set of data, as hosts which have been up the longest are more likely to be polled. Analysis of the data must accommodate this effect.

In this study, on the other hand, we directly measured TTF and TTR by polling each of the selected hosts frequently. We used an exponential distribution of times between polls, with a mean of ten minutes, both for statistical purposes and to avoid synchronous behavior where multiple tattlers poll the same host at once.

The Tattler system, the monitoring tool we built, maintained a data base of *available epochs* for each host being monitored. Each epoch represented one period when the host was known to be functioning, and was stored as a tuple  $\langle \text{host}, \text{boot time}, \text{last sample time} \rangle$ . The intervals between the available epochs were treated as periods when the host was unavailable.

Each poll returned either the time since the host was last initialized, or a failure. The durations were merged into the list of available epochs for the host, extending an epoch if they had the same boot time.

At the end of the experiment, we extracted the list of available epochs for each measured host and computed the distributions for TTF and TTR. Our analysis then derived availability from these values.

The accuracy of our measurements depended on each machine maintaining an accurate record of its boot

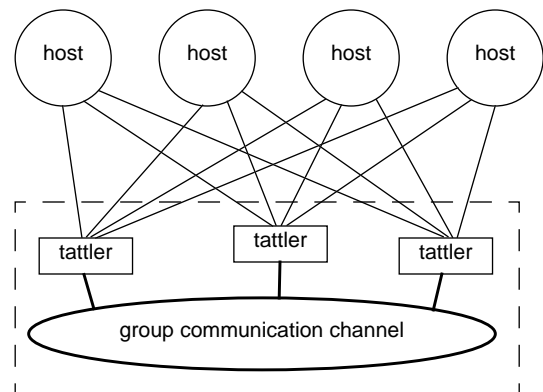
time and current time of day. We observed some systems that appeared not to maintain this information, resulting in negative or multi-decade uptimes; we removed these outliers before analyzing the data.

The measurements were also sensitive to the polling frequency. If a machine failed after less than ten minutes, the Tattler might well miss the period the host was available, though it would detect very short downtimes using the boot time in each sample.

## 2.3 The Tattler system

We built the Tattler as a distributed, fault-tolerant system. The Tattler is composed of a number of replicated monitors, the individual *tattlers*, at geographically dispersed sites, as shown in Figure 1. These monitors are called tattlers since they periodically inquire about other hosts and then “tattle” to each other about what they learn. In practice we used six tattlers: four at the University of California, Santa Cruz; one at San Diego State University; and one at the Georgia Institute of Technology.

There are several advantages to replicating the tattlers around the network. First, it provides a fault-tolerant method for monitoring hosts. All but one of the tattlers can fail and the set of hosts can still be monitored (albeit in a degraded mode). It also provides a way of mitigating the effect of transient network failures. When monitoring hosts from a single point, the failure of one router can prevent any host from being polled. When several relatively independent polling daemons work together, it will be very unlikely that a total failure can occur. Second, because the tattlers are distributed they can perform many more queries than a single polling program—with six tattlers, each tattler polled each host



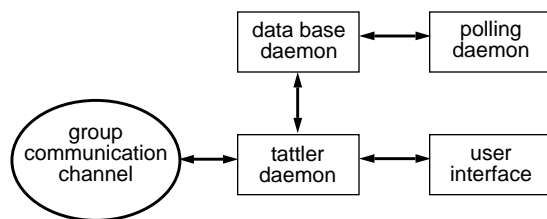
**Figure 1: The overall structure of the Tattler system**

on average only once an hour to achieve an overall mean time between polls of ten minutes. While a single polling program would create roughly the same message traffic, it would concentrate all the traffic onto a small number of network links from the monitoring host to the Internet backbone networks. The parallelism of using multiple nodes also decreases the load on them, so we could use ordinary workstations without disturbing their users. A non-replicated monitor would also take significantly longer to complete its task since it would have to poll for a longer period to make up for data lost due to failures.

Each tattler maintained a copy of the list of hosts to poll, and of the data base of available epochs for each host. The list of hosts was a sequence of tuples <host, poll method, poll interval>. The data base contained a sequence of tuples, one per available epoch for each host being monitored, of the form <host, boot time, sample time>.

The tattler replicas were coordinated using a weak consistency group communication protocol [Golding92]. This protocol provides operations for new replicas to join the group, sending the new replica a copy of the data base in the process; for leaving the group, when a tattler is to be shut down; and for propagating metadata and data base changes. The tattlers did not communicate in real time; instead, they proceeded independently and periodically merged their data bases. This allowed the system to continue functioning when individual tattlers were temporarily unavailable or the network had partitioned—common events for a system built on the Internet.

An individual tattler was composed of several components: a tattler daemon, a data base daemon, and a polling daemon, as shown in Figure 2. The tattler daemon coordinated the other daemons, and was responsible for managing the consistency of the replicated data base through the group communication channel. The data base daemon provided stable



**Figure 2: The structure of an individual tattler replica**

storage for sample observations (from the polling daemon), and metadata from the tattler daemon.

The polling daemon produced sample observations. It took samples at a specified rate, and could be requested to start or stop sampling. For this study, it used exponentially distributed random intervals with a mean of one hour.

The system also provided a user interface for controlling individual tattlers. It allowed hosts to be added and deleted from the monitoring list, and allowed a user to suspend monitoring of certain hosts. It could inform a tattler that it should shut down and leave the process group. New tattlers could be added equally easily. The user interface contacted a single tattler—preferably the closest—to perform all of these operations, and the group communication protocols ensured that the operation was eventually known by all tattlers.

### 3 Results

This study was conducted over a three month period. The study is continuing, and as more time passes the accuracy of our results will improve.

#### 3.1 Time-to-failure

During the three-month experimental period, we observed 4 692 intervals when systems were available. The average duration of these periods was 15.919 days, or 382.06 hours. Table 1 summarizes these measurements.

**Table 1: Measured MTF**

mean	15.92 days
	±0.28 (50% confidence)
	±0.82 (95% confidence)
	±1.08 (99% confidence)
median	5.530 days
<i>n</i>	4 692 intervals
$\sigma$	28.66

The distribution is rather skewed, with a few very long intervals. Table 2 shows the percentiles, while Figure 3 shows the density and Figure 4 shows the cumulative distribution.

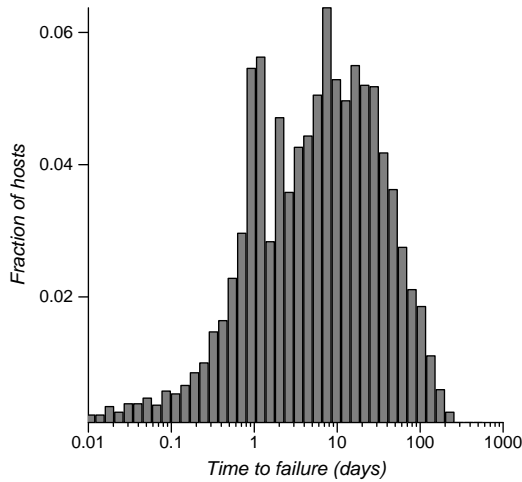
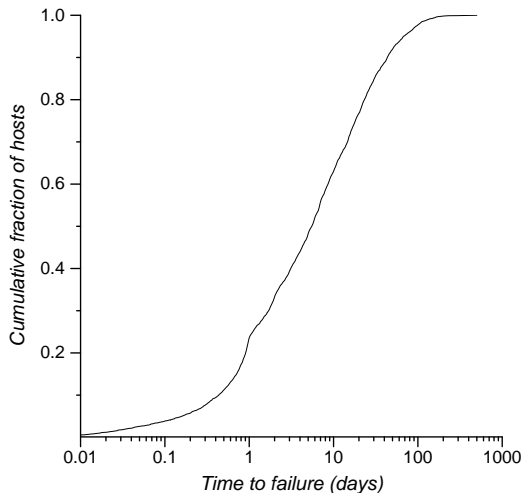
Since we monitored the systems for three months, we found few periods when a system was up for more than 100 days. We expect that we will continue to observe a longer and longer upper tail on the MTF distribution as

**Table 2: Distribution of MTTF (days)**

min	0.000914
25%	1.101
50%	5.530
75%	18.18
max	551.5

we monitor for longer periods, and that this will increase the skew between the mean and median values.

These values are close to what system administrators would expect, according to anecdotal reports we have gathered. The values differ greatly from the MTTF values reported by manufacturers because we are using a different definition of failure: manufacturers are generally concerned with permanent hardware

**Figure 3: MTTF distribution****Figure 4: Cumulative distribution of MTTF**

failures, while we are concerned with the ability to communicate with a service.

The number reported is also much shorter than the time between operating system or hardware “failures”. Manually shutting down a machine—for software maintenance or to conserve power, for example—make the system unavailable according to our definition. Many people reboot their systems nightly or weekly, contributing many short intervals. The spike at one day in Figure 3 bears this out.

### 3.1.1 Is failure a Poisson process?

In our previous study, we found evidence that the time-since-initialization values we measured were not exponentially distributed, and argued that if TTF were exponential, time-since-initialization must be as well—indicating that TTF most likely does not follow a Poisson process. In this section, we apply the same test statistic to the data we collected in this study.

We use a test statistic based on the parametric family of distributions with linear failure rate density, which has been shown to be applicable to a large class of nonparametric distributions as well, and has been shown to be applicable to machine behavior [Doksum84]. This test does not depend on advance knowledge of the mean of the proposed governing distribution. For  $n$  samples  $t_1$  through  $t_n$  with mean  $\hat{t}$ , the test statistic is given by:

$$T = \frac{1}{\sqrt{n}} \sum_{i=1}^n \left[ 1 - \frac{1}{2} \left( \frac{t_i}{\hat{t}} \right)^2 \right]$$

If the null hypothesis  $H_0$  that the samples come from a single exponential distribution is true, the test statistic  $T$  has a standard normal distribution when the sample size  $n$  is large. Thus the null hypothesis  $H_0$  can be rejected at a specified level of significance when the value of the equivalent formula

$$T = \frac{1}{2} \sqrt{n} \left[ 1 - \frac{\hat{\sigma}^2}{\hat{t}^2} \right]$$

is large, where  $\hat{\sigma}^2$  is the sample variance.

The test statistic  $T$  can be also used in testing the null hypothesis that the samples come from a population with a linear failure-rate density as well as a population with a nondecreasing failure-rate average. For these cases, the null hypothesis can be rejected for extreme values of the test statistic  $T$ ; the significance probability is calculated from the standard normal distribution.

No matter how large the sample size, no amount of testing can assure that a population distribution is exponential. By contrast, the test statistic  $T$  can quantify the prohibitively small probability that certain samples were derived from an exponential population distribution.

We calculated  $T$  for the TTF values for each host, and found  $T = -76.7607$ , which gives a vanishingly small probability that TTF is exponentially distributed.

### 3.1.2 Comparison with previous study

In our previous study, we observed MTTF values ranging from 15.85 to 20.14 days, depending on the kind of system, as compared to our observation of 15.92 days in this study. Those systems with a large population tended to be in the upper portion of that range, with MTTF between 17.96 to 20.14 days. However, the difference between the means is only 2.041 days, or 13% of our new observations.

The samples in our previous study were biased to over-represent more reliable systems, which increased the time-to-failure measure. These new results confirm the bias in the earlier study, and encourage our belief that we have remedied it.

### 3.2 Time-to-repair

The measured overall MTTR was 1.201 days (28.82 hours), with 3 500 epochs when systems were unavailable. Table 3 summarizes the results.

**Table 3: Measured MTTR**

mean	1.201 days
	±0.021 (50% confidence)
	±0.062 (95% confidence)
	±0.082 (99% confidence)
median	0.3394 days
$n$	3 500 intervals
$\sigma$	1.885

Once again the distribution is strongly skewed, with the median time (just over eight hours) much shorter than the mean. Table 4. reports the percentiles, while Figure 5 shows the distribution and Figure 6 the cumulative distribution.

#### 3.2.1 Is repair a Poisson process?

As with failure, repair is often assumed to follow a Poisson process. We applied the same test statistic to our measured repair times, which yielded

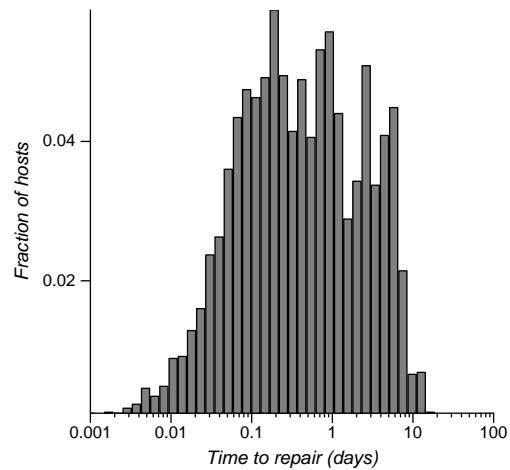
**Table 4: Distribution of MTTR (days)**

min	0.001100
25%	0.08988
50%	0.3394
75%	1.405
max	13.93

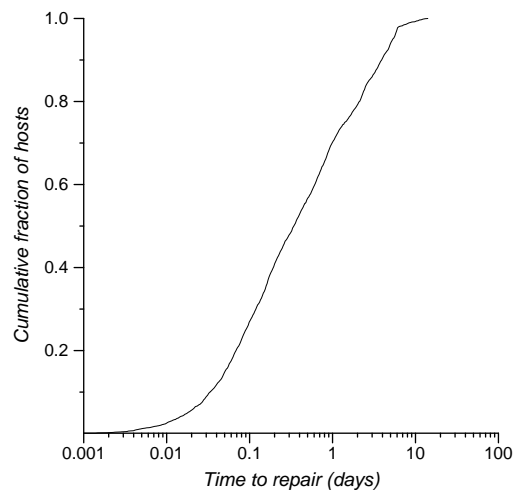
$T = -43.3191$ . Once again, the chance that repair is exponential is vanishingly small.

#### 3.2.2 Comparison with previous study

In our previous study, we estimated MTTR values between 1.86 and 2.96 days for the most common kinds of hosts, and with few exceptions the MTTR estimates for other models were notably longer than our current measurements.



**Figure 5: MTTR distribution**



**Figure 6: Cumulative distribution of MTTR**

How could significant errors have been introduced into our previous estimates of MTTR? In that study we derived MTTR from the MTTF and availability  $A$  by:

$$MTTR = \frac{MTTF(1 - A)}{A}$$

This is most easily derived from two state birth-death process, but by using results from the theory of renewal processes this result can be shown to be independent of the distributions of MTTF and  $A$  [Trivedi82].

The MTTR estimate is linearly dependent on MTTF. If the MTTF were overestimated, the estimated MTTR would be proportionally longer than the actual. Also, if the confidence interval for the availability of the MTTF is large, then the estimated MTTR can deviate significantly from the true value.

An accurate MTTR also depended on the accuracy of the availability measure. Consider the rate at which the value of MTTR changes when an error is made in the estimate of  $A$ :

$$\frac{\partial}{\partial A} \frac{MTTF(1 - A)}{A} = \frac{-MTTF}{A^2}$$

The error introduced into the estimate of MTTR is quadratic in the error in the estimate of  $A$ . As we will see, the availability measure from the earlier study was flawed.

### 3.3 Availability

In this study, we computed availability as the ratio of the time we observed that a host was available to the total time, both failed and functioning. We computed an overall availability of 0.9260 for a population of 1 162 hosts, or a mean of 27 days unavailable per year. Table 5 summarizes. Figure 7 shows a histogram of the distribution over the hosts. Again, the distribution is highly skewed. Only a small fraction of the hosts (6.7%) were available less than 75% of the time.

Many of the hosts (242, or 21%) have availability of 1.0. These hosts have not failed in the time we have been monitoring them. We expect that as we continue

**Table 5: Computed availability**

mean	0.9260
	±0.002 (50% confidence)
	±0.007 (95% confidence)
	±0.009 (99% confidence)
$n$	1162 hosts
median	0.9723

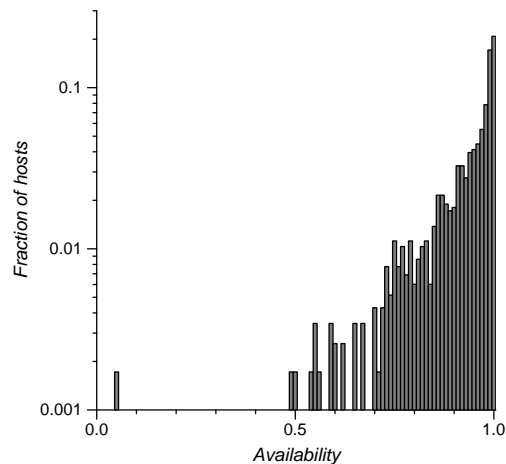
to monitor them we will be able to better characterize these highly-reliable hosts.

The time that these systems are unavailable—several days per year—is very different from anecdotal accounts of behavior of dedicated server systems. In general we believe that this is because we are monitoring the average host on the Internet, which is likely to be a workstation-class machine for a single user or for a small workgroup. These users do not generally invest in uninterruptible power supplies, redundant processors, or other special mechanisms for ultra-highly reliable systems.

#### 3.3.1 Comparison with previous study

In the previous study we reported much lower availability for most kinds of systems. This in turn led to substantially different results for MTTR. We measured availability by polling a large list of host names to determine which names corresponded to real systems, then polling again two months later to measure what fraction were reachable.

That method suffered from two problems. First, it over-represented reliable systems, since the poll actually measured the conditional probability of being able to reach the host a second time, and the probabilities of the



**Figure 7: Availability distribution.**

polls being successful were not independent. Second, it did not differentiate between the reliability of the network and the reliability of the host. A transient network failure that lost the query packets would be indistinguishable from a host failure, and packet loss is likely when network segments become congested.

#### 4 Summary

We have performed a longitudinal measurement study of the TTF and TTR of a sample of hosts connected to the Internet. We collected data from almost 1 200 hosts, using only data that could be obtained via the Internet with no special privileges or added monitoring facilities.

We chose to directly measure TTF and TTR, rather than trying to estimate them from other measures. We did this by polling each host regularly to determine how long it had been functioning. We ran the monitor for three months to obtain long-term measures of TTF and TTR. We used these measures to compute overall availability.

The results of the previous study reflected a bias toward more reliable hosts that needed addressing. Our use of direct measurement improves upon the earlier results. We now believe that we have eliminated all significant sources of bias in the measurements, and in so doing have presented the most accurate statistic on host reliability available.

In order to accomplish this task, we constructed the Tattler, a fault-tolerant distributed monitor that continuously polled hosts over the Internet. Its distributed nature allowed us to avoid putting too great a load on any particular host or on any particular segment of the Internet. By dispersing the tattlers geographically, we were able to minimize the effects of network failures, which biased the previous availability estimate.

From our TTF measurements, it appears that many systems stay up for about one week. The data suggest daily and weekly patterns, where systems are unavailable for a short period each day or week. For both TTF and TTR, the mean and median are significantly different, suggesting that only a few systems stay up a long time, but that most are unavailable for only a short time.

The TTF and TTR distributions are clearly not exponential. We believe that care needs to be taken in

future reliability analyses to ensure that assumptions about exponential behavior do not cause problems.

Those interested in participating in this study, by running tattlers on their systems, should contact the authors.

#### Acknowledgments

The work at the University of California has been supported by the Office of Naval Research under grant N00014-92-J-1807. Some of this research was performed on equipment donated by Sun Microsystems. We are grateful to the students who have contributed to the Tattler, including N. Ganatra, A. Grice, K. B. Sriram, D. Schreiber, and A. Sullivan. We are particularly grateful to Dr. J. Gray for his encouragement to pursue this line of inquiry.

#### References

- [Doksum84] K. A. Doksum and B. S. Yandell. *Handbook of Statistics*, volume 4. Elsevier, 1984.
- [Ganatra92] Nitin K. Ganatra. Census: collecting host information on a wide area network. Technical Report UCSC-CRL-92-34, Computer and Information Sciences, University of California Santa Cruz, 1992.
- [Golding92] Richard A. Golding. *Weak-consistency group communication and membership*. Ph.D. dissertation, Computer and Information Sciences, University of California Santa Cruz, December 1992.
- [Gray85] Jim Gray. Why do computers stop and what can be done about it? Technical Report 85.7, Tandem Computers, June 1985.
- [Gray90] Jim Gray. A census of Tandem system availability between 1985 and 1990. Technical Report 90.1, Tandem Computers, January 1990.
- [Long89] Darrell D. E. Long, John L. Carroll, and Kris Stewart. Estimating the reliability of regeneration-based replica control protocols. *IEEE Transactions on Computers*, 38(12), December 1989.
- [Long91] Darrell D. E. Long, John L. Carroll, and C. J. Park. A study of the reliability of Internet sites. In *Proceedings of the Tenth Symposium on Reliable Distributed Systems*, pages 177-186, Pisa, September 1991.
- [Long92] Darrell D. E. Long. A replicated monitoring tool. In *Proceedings of the Second Workshop on the*

*Management of Replicated Data*, pages 96–99,  
Monterey, November 1992.

[Mourad85] S. Mourad and D. Andrews. The reliability of the IBM/XA operating system. In *Proceedings of the 15th Annual International Symposium on Fault-tolerant Computing*. June 1985.

[Pâris86] J.-F. Pâris. Voting with witnesses: a consistency scheme for replicated files. In *Proceedings of the 6th International Conference on Distributed Computing Systems*, pages 606–612, Cambridge, 1986.

[Trivedi82] K. S. Trivedi. *Probability & Statistics with Reliability, Queuing and Computer Science Applications*. Prentice-Hall, Englewood Cliffs, New Jersey, 1982.