University of California
Santa Cruz

A study of the reliability of hosts on the Internet

A thesis submitted in partial satisfaction
of the requirements for the degree of
Master of Science
in
Computer and Information Sciences
by
K. B. Sriram
June 1993

The thesis of K. B. Sriram is approved:

Prof. Darrell D. E. Long

Prof. David H. Haussler

Prof. Charles E. McDowell

Dean of Graduate Studies and Research

Copyright © by

K. B. Sriram

1993

Contents

List of Figures

A study of the reliability of hosts on the Internet

K. B. Sriram

Abstract

This thesis is a study of the failure statistics of hosts on the Internet. To cross-verify our estimates of failure rates, we measure three different kinds of observations about hosts on the Internet – the time interval between failure, the number of failures in a certain time interval, and the length of time a host has been up at an arbitrary instant of time. These observations are made on different samples of hosts on the Internet.

We present statistical results that allow us to derive estimates about the failure statistics of the population from these three different types of observations. The estimates from the the three experiments are slightly different but comparable, providing three independent sources of evidence about the failure statistics of hosts on the Internet. We also attempt to explain the observations by modelling the statistical process with two distributions – the exponential and the Weibull distribution. The observed data are explained well (in a statistical sense) by both models, but the Weibull model fits the observed sample better. We believe this to be evidence that the failure rates of hosts on the Internet are nearly constant, but with a high initial value that quickly decreases to a constant as the machine continues to run.

Acknowledgments

I wish to thank Professor Darrell Long, my thesis advisor, for his technical contributions and support to this work. I am especially grateful to him for his encouragement and patience during times of crisis.

It is my pleasure to acknowledge Phil Long, Madhukar Thakur and Yoav Freund for several interesting discussions on applied statistics that helped me find statistical tools I could use in this thesis.

This thesis was supported by seed funds from the University of California.

Chapter 1
Introduction

1.1 Overview

According to a recent study [15], there are nearly 1.3 million hosts that are reachable on the Internet. The Internet is used to support a number of wide area network distributed applications. It would be useful to determine the reliability of these applications, and be able to predict their behavior. While there have been studies about network latency, response time and path length [2] for typical operations in such applications, there has been less work on studying the reliability of such applications on the Internet. There is also little published data on measured failure statistics of hosts on the Internet. This thesis is a study of the failure times of hosts on the Internet, and provides estimates about the failure rates of machines which can be used to predict the reliability of wide area distributed applications running on the Internet.

There are at least two other uses for this data. Systems administrators can profit by comparing their systems with the average to see if there is any unusual behavior in their system. Such comparisons can also be made to test the stability of a new operating system before its release. A third use, as mentioned before, is in modeling the reliability of distributed systems [14]. The results from our study can be used to validate the model used, and provide estimates for the reliability of the application.

Measuring the failure rate is non-trivial as much of the information necessary to determine this parameter is difficult to collect anonymously over the Internet. Continuous monitoring of hosts to determine when they fail is infeasible for large samples, as it generates a very high amount of network traffic which slows things for other applications using the Internet. Another problem with this approach is the presence of network partitions, where the network fails but not the hosts themselves. Such failures are sometimes indistinguishable from host failure. Useful observations that allow deducing failure rates almost always require the cooperation or consent of the administrator of the machines being sampled. Gathering information this way is time consuming because of the process of contacting and obtaining permission, and does not yield enough sample points to make reliable estimates.

Another factor to consider is that the sample we take from the Internet be representative of the whole. This is easier to achieve when the measurements are made anonymously, but harder for samples obtained through contacting the person administering the machine. Are people who respond prone to have machines that run differently from the norm? Is it skewed by the way in which they are contacted? It is also necessary that the samples be independent of one another. For instance, it is likely that hosts on the same local area network would show some correlation in the times they fail – a power outage would affect all these machines at the same time.

Our study analyzes results obtained through three different observations about hosts on the Internet, only one of which was made anonymously. The anonymous observation gives more sample points, but cannot be analyzed unless we make assumptions about the distribution of the failure rates of machines. The other two observations do give results that are not skewed through assumptions about failure rates. However, we could collect only a limited number of samples, so it is non-trivial to predict statistics about the Internet from such a small collection of data. We can make more reliable estimates with a careful choice of samples, even from limited amounts of data. If the sample population is representative of the whole, then even small sample sizes are sufficient to derive good estimates.

In this thesis, failure time is taken to be the time at which a machine stops running. Of course, a machine can stop for a variety of reasons ranging from an application freezing the machine, to simply shutting off the machine at the end of the day. We think this is a reasonable indicator of how hosts in normal circumstances behave, and is a practical indicator of how long a machine can be expected to stay up before one of a number of factors cause it to be rebooted or halted. Such an assumption is also useful when analyzing the behavior of a distributed database. We want to analyze its performance under "normal" circumstances, when machines can stop running for a variety of reasons.

We also assume that the failure times of machines are independent of each other. A straightforward sampling of hosts does not have this property. For instance, it is likely that the failure times of hosts on the same LAN are correlated, since a power failure may cause all hosts on the LAN to be halted at the same time. Similarly, a common phenomena is that a file server failing causes hosts that are using it to reboot shortly thereafter. We attempt to preserve independence with a careful choice of sample points where possible, but it is important to note that our analysis does depend on the assumption of independence between hosts.

The estimates presented in the study come from three observations of the Internet. The first is a measurement that can be made anonymously over the Internet, and yields a sample of the "age" of the host, in other words, how long it has been running when the measurement is taken. Under certain assumptions about the distribution of failure times, this enables us to estimate the average time to failure of hosts. A second is a sample of the times when a machine is restarted. This information cannot be gathered anonymously, and is obtained by examining records of files that are updated on most Unix machines when the machine is restarted. The last measurement is sampling the number of times a machine has failed between two points of time. This information can be gathered anonymously on machines running a particular kind of daemon, but as it changes some data on the host, we decided to obtain the permission of the systems administrators before collecting this data.

1.2 Previous Studies and Related Work

This work is a continuation of a study of the reliability of hosts using the Internet [6]. In this study, a large population of hosts were queried to obtain a sample of uptimes of machines.

From these observations, an estimate of time to failure was obtained. This estimate involved assuming that the underlying data followed an exponential distribution. A hypothesis fitting test however showed that the data was significantly different (in a statistical sense) from an exponential distribution. We present some alternate analysis of similar data that derives estimates based on the assumption that the data follows a generalization of the exponential distribution.

Other studies about the Internet have focused on network latency, message path, network load and the types of protocols used. For instance Pu, Korz, and Lehman [2] propose a general methodology to measure network routing and latency for applications over the Internet, and present some example applications of the technique. The focus of this work is on application response time, which is different from our study which estimates failure times of hosts on the Internet. A large number of studies have been made about the types of protocols used in typical Internet traffic [16]. Other studies involve analyzing the usage of particular protocols to determine characteristics of the Internet. For instance, one study analyzes files transferred in the File Transfer Protocol (FTP) to determine the size of files transferred and occurrence of duplicate file transfers. Another study by the same group analyzes electronic mail traffic to determine people with shared interests.

A different approach to measuring the reliability of computers was taken in a study made by Gray [11], analyzing the performance of Tandem computers. In this study, numerous reports from customers were studied to get a breakdown of the different causes of failure and their frequency. This work focuses on analyzing the kinds of failures that happen for a particular computer, and is not based on measurements through the Internet.

As examples of work where measurements made in our study can be used, Long, Carroll and Stewart [14, 3] study replicated databases, and model their reliability with failure models of hosts that can be validated with the results in this study. Resource discovery protocols have been proposed and implemented on the Internet (for instance by Schwartz and Tsirigotis [19]). Analyzing the performance of these protocols in the face of failures of hosts can benefit from the estimates available from our study. General techniques in building wide area distributed applications have also been suggested [20]. Analyses of the performance of such systems can also use the results from our study.

1.2.1 Organization of thesis

Chapter 2 discusses the theory behind the formulae used to estimate the average failure times from the observations. It places the experiments in the framework of renewal processes, and presents results from statistics that are useful in obtaining estimates about failure rates from these observations.

Chapter 3 describes in more detail the actual experiments that were performed, how the observations were collected, and the data obtained through these observations. It also describes the details about the distribution and the size of samples that were measured.

Chapter 4 concludes the thesis by deriving estimates and drawing conclusions from the collected data using the results presented in Chapter 2. It also suggests ways in which these estimates can be refined, and suggests new directions in which this study can proceed.

Chapter 2
Theory

We now present the statistical results used to arrive at estimates for the mean time to failure and the mean time between reboots. A full presentation of these results from classic statistical theory can be found in the references [18, 5, 4, 1, 9]. For the most part, this chapter confines itself to quoting appropriate results from various references and showing how they are applied.

Three different types of observations were made about hosts on the Internet.

- Observations about the time between reboots of a host,
- Observations about the "age" of the host, which is the time at which the host was last rebooted,
- Observations about the number of reboots of a host in a certain interval of time.

These observations differed in the size and distribution of the samples that were taken.

The first section places the experiments that were conducted in the context of *renewal processes*. This is a convenient framework to describe our observations, and makes it easy to apply results from statistics to our study. The next three sections describes how the three different types of observations about renewal processes lead to different methods to arrive at estimates for the interevent renewal time. The final section evaluates the appropriateness of each of these estimates.

2.1 Renewal Processes

Renewal processes have been studied for a long while [5, 4, 9] and form a convenient framework in which to describe our observations. Results from renewal theory apply to deriving estimates about failure rates from our observations. We first describe renewal processes, and show how our observations fit into this model. Then we present key results from renewal theory that are applicable to our study.

Suppose we have a population of components whose failure time X is a continuous random variable with density function $f(x)$. Suppose further that we start with a new component at time zero. Suppose this component fails at time X_1. Let it be immediately replaced by a new component with failure time X_2. Then the second failure will occur after a total time $X_1 + X_2$. Let this process be continued, a component being replaced immediately on failure by a new component. The failure time of the r^{th} component is X_r, and the r^{th} failure occurs at time $S_r = X_1 + \ldots + X_r$. If $\{X_1, X_2, \ldots\}$ are independently identically distributed random variables (also referred in short as *iid* variables), we call the system an ordinary renewal process. Each replacement is called an *renewal event.* The lifetime X_i of any component is called the *inter-event renewal time.*

There are two associated random variables that are of interest to us. The first is the age U of the component currently in use at some random instant of time. This is also called the *backward occurrence* variable, since it is the time since the last renewal event occurred. The second random variable is the (discrete) random variable N_t that counts the number of renewal events in the time interval t.

In the context of the experiments we conducted, the the act of restarting a machine is a renewal event. The corresponding inter-event renewal time is the time between reboots. The act of a machine halting is another type of renewal event, and the corresponding inter-event renewal time is the time to failure.[1]

[1]This actually assumes that the interval between a machine failing and it being restarted is zero. The time scale in this case is really a sliced up version of real time, where the slices we consider are the times when the machine is up, and we cut out the portions of time when the machine is down.

The parameter we want to estimate in a renewal process is E$[X]$, the mean inter-event renewal time. The next three sections consider three different observations we can make about the renewal process, and present appropriate statistics that estimate the mean inter-event renewal time. In the context of the discussion in this section, the three observations we took are:

- A random sample of observations of X, the inter-event renewal time, which is the time between reboots of a host,
- A random sample of observations of U, the backward occurrence time which are the observations about the "age" of a host, and
- A random sample of observations of N_t, the number of renewal events in a time interval t, which is the sample about the number of reboots of a host in a certain interval.

2.2 Sampling the inter-event renewal time

This sample of the renewal process provides us with n independent observations $x_1, x_2, \ldots, x_n$ of the inter-event renewal time X. A good estimate for the mean inter-event renewal time is simply the mean of the observations $\overline{X} = (\sum_{i=1}^{n} x_i)/n$.

Standard statistical results (see for instance [1, 8, 18]) show that this estimate is unbiased, consistent and sufficient. The following is a useful result from these references that we state without proof.

Result 2.1 *If $X_1, X_2, \ldots, X_n$ are governed by a normal distribution with mean μ, the random variable*

$$T = \frac{\overline{X} - \mu}{S/\sqrt{n}}$$

follows a Student T *distribution with $n-1$ degrees of freedom.*

Here S is the standard deviation estimator $S = \sqrt{\frac{\sum_1^n (X_i - \overline{X})^2}{n-1}}$. Empirical studies (see [8] for instance) suggest that if n is large, this result holds even when X differs considerably from normal. In practice, the equation is taken to be true for $n > 30$ no matter what distribution X follows, which leads to this confidence estimate (again from [1, 8, 18]) for the mean.

Result 2.2 *A* $(1-\alpha)$ *confidence interval for the mean* μ *is*

$$(\overline{X} - t_{\alpha/2}\frac{S}{\sqrt{n}}, \overline{X} + t_{\alpha/2}\frac{S}{\sqrt{n}})$$

where if T *is a Student* T *distribution with* $n-1$ *degrees of freedom,* $\mathrm{P}[T > t_{\alpha/2}] = \alpha/2$.

This result is applicable to our experiment that samples the time intervals between reboots of a machine. We can compute a confidence interval for the mean time between reboots using this result. Notice that we made no assumptions about the underlying distribution to obtain the estimate. This is a particularly nice result to have, since it assumes nothing about the underlying distribution but still gives us confidence intervals for the estimate.

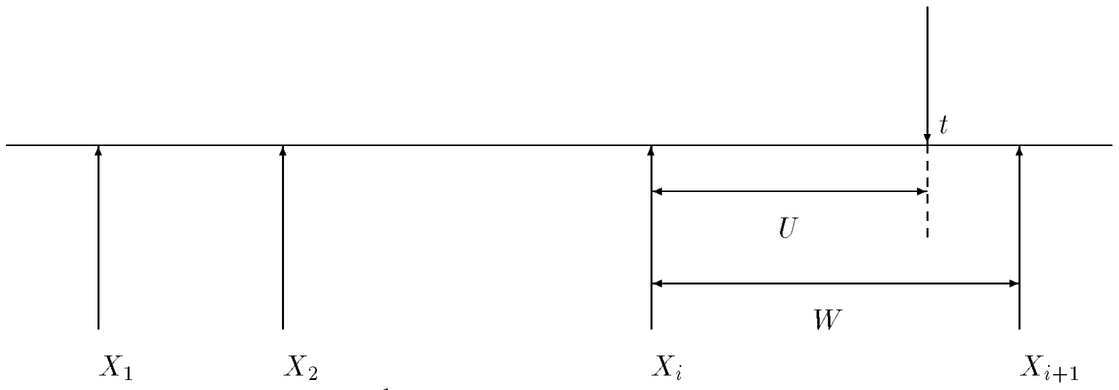
2.3 Sampling the backward occurrence

The backward occurrence variable U_t is defined as follows: pick some fixed time instant t. U_t is the random variable that is the length of time measured backwards from t to the last renewal at or before t. If there have been no renewals before t, U_t is defined to be t. We are interested in situations where t is large, in other words when the renewal process has been running for a long time. The distribution of U_t is independent of t for large t [4], so we will just refer to the distribution of U instead of U_t in the discussion that follows.

In the experiments that we conducted, this sample corresponds to measuring the uptimes of machines (*viz.* how long it has been running) at some instant of time.

We now derive results for estimates (see [4] for this approach) for the mean inter-event arrival time given a sample of backward occurrence times. Let the random variable U denote the age of the component in use at the present time.

We first obtain the distribution of U in terms of the distribution of X as follows. For convenience, let W denote the length of the interevent time into which we enter by randomly



X_i: the occurrence of the i^{th} renewal event
U: the backward occurrence time
FIGURE 2.1: Backward occurrence time

picking a time instant. Observe that the distribution of W is different in general from the distribution of X, since we are more likely to "fall into" a longer interval than a shorter one. We obtain the density f_U of the random variable U in two steps. First, we obtain the density f_W for W, and compute the conditional density $f_{U|W}$ for U conditioned on a particular value for W. Next, we obtain the joint density $f_{W,U} = f_{U|W}f_W$, and then the unconditional density f_U by integrating the joint density $f_{W,U}$ over all possible values of W.

We assume that the probability that our random time instant falls in an interevent gap of length w is proportional to the length of the gap w, and the frequency of such gaps, which is $f(w)dw$. This means

$$f_W(w)dw = \frac{wf(w)}{C}dw$$

where C is an appropriate normalizing constant that makes f_W a density function. For f_W to be a density function, we need

$$\int_{-\infty}^{\infty} f_W(w)dw = 1$$

In other words,

$$\int_{-\infty}^{\infty} \frac{wf(w)}{C} dw = 1$$

Solving for C, we get

$$C = \int_{-\infty}^{\infty} wf(w)dw$$

and by definition of the mean $\mathrm{E}[X]$ of the random variable X,

$$C = \mathrm{E}[X].$$

Therefore the density function $f_W(w)$ for the random variable W satisfies $f_W(w) = wf(w)/\mathrm{E}[X]$.

We obtain the conditional density $f_{U|W}$ for U conditioned on a particular value for W with the following argument. Let our randomly chosen observation time fall into an interevent period of length w_0. Since the observation point is randomly chosen, the age of the component (given that we are in an interevent period of length w_0) is uniformly distributed over w_0. Thus

$$f_{U|W}(u|w = w_0) = \begin{cases} \frac{1}{w_0} & 0 \leq u \leq w_0 \\ 0 & \text{otherwise} \end{cases}$$

The joint density $f_{W,U}$ is simply $f_{U|W}(u|w)f_W(w)$, so substituting the values for $f_{U|W}$ and f_W, we get

$$\begin{aligned} f_{W,U}(w,u) &= f_{U|W}(u|w)f_W(w) \\ &= \frac{wf(w)}{w\mathrm{E}[X]}, \quad 0 \leq u \leq w < \infty \\ &= \frac{f(w)}{\mathrm{E}[X]}, \quad 0 \leq u \leq w < \infty \end{aligned}$$

and $f_{W,U}$ is 0 everywhere else.

We can now obtain the density function f_U by integrating $f_{W,U}$ over all values of w to get

$$\begin{aligned} f_U(u) &= \int_{-\infty}^{\infty} f_{W,U}(w,u)dw \\ &= \int_{w=u}^{\infty} \frac{f(w)}{\mathrm{E}[X]}dw \\ &= \frac{\mathcal{F}(u)}{\mathrm{E}[X]} \end{aligned}$$

where $\mathcal{F}(u) = \mathrm{P}[X > u]$.

We now derive a lemma that is used in later discussions of estimates.

Lemma 2.3 *$E[U^i] = \frac{\mathrm{E}[X^{i+1}]}{(i+1)\mathrm{E}[X]}$ whenever $E[X]$, $E[X^{i+1}]$ and $E[U^i]$ exist.*

Proof:

$$\begin{aligned} \mathrm{E}[U^i] &= \int_0^{\infty} u^i f_U(u)du \\ &= \left.\frac{u^{i+1}}{i+1} f_U(u)\right|_0^{\infty} - \int_0^{\infty} \frac{u^{i+1}}{i+1} f_U'(u)du \end{aligned}$$

Since $f_U(u) = \frac{\mathrm{P}[X>u]}{\mathrm{E}[X]}$, it follows that $f_U'(u) = -\frac{f(u)}{\mathrm{E}[X]}$. Therefore,

$$\begin{aligned} \mathrm{E}[U^i] &= 0 + \frac{1}{(i+1)\mathrm{E}[X]} \int_0^{\infty} u^{i+1} f(u)du \\ &= \frac{\mathrm{E}[X^{i+1}]}{(i+1)\mathrm{E}[X]} \end{aligned}$$

□

In particular, this gives us the well-known result [18] $\mathrm{E}[U] = \mathrm{E}[X^2]/2\mathrm{E}[X]$ as a special case of our lemma when $i = 1$.

The next four subsections describe four estimates for $\mathrm{E}[X]$, the mean inter-event time, and the assumptions that underlie the assumptions. Of these four estimates, only one provides confidence intervals, but it makes some strong assumptions about the distribution of the inter-event time. The other three make slightly weaker assumptions about the distribution, but don't give confidence intervals for the estimates. These three are presented mainly because they are interesting alternate ways to analyze the data.

2.3.1 The Exponential model

This method models the distribution of inter-event times as an exponential distribution. The exponential model of failure times is a natural choice both because it is a good approximation to a lot of real world component lifetime distributions, and also for its analytical simplicity.

In the exponential model, we assume that X, the variable representing the time between events follows the distribution $f(x) = \lambda e^{-\lambda x}$. In the previous section we showed that U, the backward occurrence variable follows the distribution $f_U(u) = \frac{\mathcal{F}(u)}{\mathrm{E}[X]}$. For the exponential distribution, $E[X] = 1/\lambda$ and $\mathcal{F}(u) = e^{-\lambda u}$. Therefore, $f_U(u) = \lambda e^{-\lambda x}$. This is an interesting result since it shows that the backward occurrence time follows the same distribution as X when X is exponentially distributed.

We state without proof standard results [12] that provide confidence intervals for the mean of an exponential distribution. A $(1-\alpha)$ confidence interval estimate for $\mathrm{E}[U]$ is

$$(\overline{U}\frac{2n}{\mathcal{X}^2_{2n;\alpha/2}}, \overline{U}\frac{2n}{\mathcal{X}^2_{2n;1-\alpha/2}})$$

where if X is a chi-squared distribution with $2n$ degrees of freedom, $\mathrm{P}[X > \mathcal{X}^2_{2n;\alpha/2}] = \alpha/2$. Since U and X are governed by identical distributions, it is also a confidence interval for $\mathrm{E}[X]$.

2.3.2 The Weibull model

The exponential distribution models a random process with a constant failure rate. The Weibull distribution is a more general version of the exponential model that allows modeling lifetimes with increasing or decreasing failure rates. A random variable X governed by the Weibull distribution has the two parameter density function $f(x) = \lambda\alpha x^{\alpha-1}\exp(-\lambda x^{\alpha})$. For the Weibull distribution, $\mathcal{F}(u) = \exp(-\lambda x^{\alpha})$, and $\mathrm{E}[X] = \frac{1}{\lambda}^{\frac{1}{\alpha}}?(1 + 1/\alpha)$. From the discussion on how the backward occurrence variable U is related to the underlying distribution of X, we have $f_U(u) = \frac{\mathcal{F}(u)}{\mathrm{E}[X]}$. Therefore, when X follows the Weibull distribution, we have $f_U(u) = \frac{\exp(-\lambda x^{\alpha})}{\frac{1}{\lambda}^{\frac{1}{\alpha}}\Gamma(1+1/\alpha)}$

If $u_1, u_2, \ldots, u_N$ are the observed samples of U, then we can derive a maximum likelihood estimate (MLE estimate) for λ and α as follows (see [1] for a treatment of the MLE technique to derive estimates). The following is a result we derive to obtain a maximum likelihood estimate for λ and α based in this technique.

The likelihood function for this set of observations is defined by

$$L = f_U(u_1) f_U(u_2) \ldots f_U(u_N)$$

As $f_U(u_i) = \frac{e^{(-\lambda u_i^\alpha)}}{\frac{1}{\lambda}^{\frac{1}{\alpha}}\Gamma(1+1/\alpha)}$,

$$L = \frac{e^{(-\lambda \sum_{i=1}^N u_i^\alpha)}}{((1/\lambda)^{1/\alpha}?\,(1+1/\alpha))^N}$$

Therefore, $\log L$ can be written (after some simplification) as

$$\log L = -\lambda \sum_{i=1}^N u_i^\alpha + N\frac{\log\lambda}{\alpha} + N\log ?\,(1+1/\alpha)$$

Now using the gradient ascent to maximize $\log L$ with respect to λ, we have

$$\frac{d}{d\lambda}\log L = -\sum_{i=1}^N u_i^\alpha + N/(\alpha\lambda) = 0$$

and solving for λ, we have

$$\lambda = \frac{N}{\alpha\sum_{i=1}^N u_i^\alpha}$$

Similarly, by taking the derivative with respect to α, we have an analogous equation for α

$$\alpha = \frac{\sum_{i=1}^N u_i^\alpha}{\sum_{i=1}^N u_i^\alpha \log u_i}(\Psi(1+1/\alpha) + \log(\frac{\alpha}{N}\sum_{i=1}^N u_i^\alpha))$$

where $\Psi(x)$ is the derivative of $\log ?\,(x)$. The second of these equations does not yield a closed form solution for α, but can be numerically estimated. Once α is available, λ can be obtained from the first equation.

2.3.3 The Gamma distribution model

The Gamma distribution is a different generalization of the exponential distribution that has the two-parameter density function $\frac{\lambda e^{-\lambda x}(\lambda x)^{\alpha-1}}{\Gamma(\alpha)}$. We use the method of moments (see [18, 1] for an explanation of this method) to obtain estimates for λ and α. The method of moments equates the sample moments with their expected values, and solves for the parameters of the model. I now derive estimates for λ and α based on the method of moments technique.

From lemma 1.3,

$$\mathrm{E}[U] = \frac{\mathrm{E}[X^2]}{2\mathrm{E}[X]}$$

and

$$\mathrm{E}[U^2] = \frac{\mathrm{E}[X^3]}{3\mathrm{E}[X]}$$

For a Gamma distribution, it is easy to derive $\mathrm{E}[X] = \alpha/\lambda$, $\mathrm{E}[X^2] = (\alpha + \alpha^2)/\lambda^2$ and $\mathrm{E}[X^3] = (\alpha(\alpha+1)(\alpha+2))/\lambda^3$

Equating the expected values of each moment with the sample moments, and solving for α and λ we get

$$\lambda = \frac{2\overline{x}}{3s - \overline{x}^2}$$

and

$$\alpha = \frac{3s - 5\overline{x}^2}{\overline{x}^2 - 3s}$$

We present the solution in terms of the sample mean and variance for convenience. Note that the sample mean and variance in this case are the mean and variance for the backward occurrence variable.

2.3.4 Histogram estimate

The last method to estimate $\mathrm{E}[X]$ first estimates the value of $f_U(u)$ at the origin based on the observed values of u. This estimate is interesting because it allows us to make an estimate that is consistent, and independent of the underlying distribution of the data. However, this method does not yield a way to obtain confidence intervals for this estimate.

This method first obtains an estimate for the probability density function (pdf) of U at the origin. Observe that since $f_U(0) = \frac{\mathcal{F}(0)}{\mathrm{E}[X]}$, and $\mathcal{F}(0) = \mathrm{P}[X > 0] = 1$, $f_U(0) = 1/\mathrm{E}[X]$. Therefore, $\mathrm{E}[X] = 1/f_U(0)$. Therefore, if we obtain an estimate for the pdf of U at the origin, its inverse is an estimate for the mean of X.

We obtain an estimate for the pdf at the origin as follows:

First plot a histogram of the observed samples of U. Dividing the fraction of samples in each bin by the width of the bin gives an estimate for the probability density function (pdf) for the midpoint of the bin [12]. This histogram is an approximation of the actual pdf $f_U(u)$. In fact, the histogram is also a consistent estimator [12] of the density $f_U(u)$ at the midpoint of each bin. We locate the first bin so that the midpoint of the first bin is at 0. Next, find out the value of $f_U(0)$ from the bin whose center is positioned on the origin. This gives us an estimate for $f_U(0)$, and from our previous discussion, $1/f_U(0)$ is an estimate for the mean of X.

This estimate has the nice property that it makes no assumptions about the distribution of the underlying model. On the other hand, it does not provide confidence intervals for the estimate, so it is just an interesting way of analyzing the data to compare with the other estimates.

2.3.5 Goodness of fit

Three of the previous four estimates propose a distribution for the inter-event time, and proceed to arrive at estimates for the parameters for the distribution. The Kolmogorov-Smirnoff test is a measure of the "goodness-of-fit" of the data with the model that is assumed to govern it. We use the Kolmogorov-Smirnoff test to determine how closely the model fits the data.

The Kolmogorov-Smirnoff test measures the maximum absolute difference of the empirical cumulative distribution function from the hypothesized cumulative distribution function.

This allows us to come up with a test that verifies or rejects the null hypothesis that the empirical cdf is "close" to the hypothesized cdf. The method works as follows ([18, 1]):

The given random sample of observations is first arranged in order of magnitude so that the values are assumed to satisfy $x_1 \leq x_2 \leq \ldots \leq x_n$. The empirical cdf $\hat{F}_n(x)$ is defined by

$$\hat{F}_n(x) = \begin{cases} 0 & x < x_1 \\ i/n & x_i \leq x < x_{i+1} \\ 1 & x_n \leq x \end{cases}$$

The Kolmogorov-Smirnoff statistic D_n is then defined by the equation $D_n = \sup_x |\hat{F}_n(x) - F_0(x)|$, where[2] $F_0(x)$ is the hypothetical cdf of the random variable X. We reject the null hypothesis at a level of significance α if the observed value of D_n exceeds the critical value $d_{n;\alpha}$. Values for $d_{n;\alpha}$ are available from tables.

2.4 Sampling the number of events

The status monitor experiment provides us with a sample of the number of failures in a certain period of time. In terms of renewal processes, this experiments gives us a sample of the number of renewal events. We present results from renewal theory that allow us to estimate the mean inter-event time from a sample of the number of renewal events in a fixed interval of time.

This observation of the renewal process samples the number of renewal events in a time interval t. The idea is that by by dividing this time period with the number of renewal events, we get an estimate of the mean interevent time.

Consider a renewal process where the inter-event time is governed by the pdf $f(x)$. Let N_t be the number of renewal events in the time period $(0, t)$. Let S_r be the time of occurrence of the r^{th} renewal event.

[2] $\sup_x f(x)$ is defined to be the least upper bound of $f(x)$ for all values of x. More precisely, $\sup_x f(x) = a$, where for all values of x, $f(x) \leq a$, and there is no b such that for all x, $f(x) \leq b < a$

Observe that $S_r = \sum_{i=1}^{r} X_i$, where X_i is the i^{th} inter event period. The following result follows directly from the Central Limit Theorem (see for instance [7]), and is stated without proof.

Result 2.4 *S_r is asymptotically normally distributed with mean $r\mu$ and variance $r\sigma^2$, where $\mu = E[X]$ and σ is the standard deviation of X. Equivalently, for every fixed y,*

$$\lim_{n\to\infty} P[S_r < \mu r + y\sigma\sqrt{r}] = G(y)$$

where $G(y)$ is the cdf of a normal distribution with mean 0 and variance 1.

Observe that $N_t < r$ if and only if $S_r > t$. Therefore $\mathrm{P}[N_t < r] = \mathrm{P}[S_r > t]$. We can use this to derive an asymptotic result about the distribution of N_t. Let

$$r_t = t/\mu + y_t\sigma\sqrt{t/\mu^3}$$

where $y_t = y + \epsilon_t$ and ϵ_t is the smallest positive value that makes r_t an integer. Therefore,

$$\begin{aligned} \mathrm{P}[N_t < r_t] &= \mathrm{P}[S_{r_t} > t] \\ &= \mathrm{P}[\frac{S_{r_t} - r_t\mu}{\sigma\sqrt{r}_t} > -y_t(1 + \frac{y_t\sigma}{\sqrt{t\mu}})^{-1/2}] \end{aligned}$$

Now for any fixed y, let t approach infinity. Observe that ϵ_t approaches 0, and $(1+\frac{y_t\sigma}{\sqrt{t\mu}})^{-1/2}$ approaches 1. Hence

$$\begin{aligned} \lim_{t\to\infty} \mathrm{P}[N_t < r_t] &= \lim_{t\to\infty} \mathrm{P}[\frac{S_{r_t} - r_t\mu}{\sigma\sqrt{r}_t} > -y] \\ &= G(y) \end{aligned}$$

where the last equality follows from the asymptotic normal distribution of S_r. In other words, N_t asymptotically approaches a normal distribution with mean t/μ and variance $\sigma^2 t/\mu^3$. Therefore N_t/t is an estimate for μ. Furthermore, N_t is approximately normal for large values of t. We can apply results from section 1.3.1 to get estimates and confidence intervals about $\mathrm{E}[X]$. A more detailed analysis of N_t is shown in [4]. This analysis shows a stronger result, *viz.* $\mathrm{E}[N_t] = t/\mu$ for all t (However, the normality assumption for N_t is still valid only for large values of t).

This result will be used in computing estimates for the mean time between reboots from the observations made in the experiment that counts the number of reboots made in a particular interval of time.

2.5 Comparison of the three methods

The method presented in section 1.2 (estimate from an actual sample of the interevent distribution) is the simplest and also the most reliable one, since we obtain good estimates for the mean, together with confidence intervals without making any assumptions about the distribution of the data. Unfortunately, obtaining such a sample is difficult when making observations about hosts on the Internet. Such observations can only be made either by continuously monitoring hosts (uses a lot of network traffic) or by contacting systems administrators willing to record and divulge information about the times a machine is started or halted, not an easy task.

The method of observing the backward occurrence is good if the real data fits the hypothetical model. For any but the simplest hypothesized model however, obtaining confidence intervals is very difficult. This approach also suffers from the limitation that it assumes a certain distribution about the data. The histogram method of analyzing the backward occurrence is distribution free, but doesn't give us confidence intervals. The advantage of this method is that it is relatively easy to obtain large samples anonymously over the Internet.

The final method of counting the number of renewal events is also a reliable method of estimating the mean, since it gives us confidence intervals without making any assumptions about the underlying distribution. The problem with this method is that the normality assumption for N_t is true only for "large"[3] values of t, which means that the experiment must be conducted over a long period of time before we can trust the estimates provided by this method. The second problem is that getting the data involves obtaining the permission of the user of the machine, since our method of obtaining the data changes some data on the queried machine. Again, this restricts the amount of data we can collect anonymously over the Internet.

[3]Cox [4] presents arguments that about three times the mean value of the distribution is a good "large" number.

Chapter 3
Implementation

In this chapter we describe the three experiments that were performed to collect observations about hosts on the Internet. We describe the way in which the data was collected and details about the implementation of any programs to collect the data. We also describe the size and scope of the data collected, and the estimates arising from these observations are presented in the last chapter.

In the context of renewal processes, each failure of a host is a renewal event. Our goal is to estimate the mean of the interevent time. The first section describes how we used `wtmp` log files to obtain a sample of the interevent times. The second section describes an experiment using Sun's status monitor service to obtain a sample of the number of renewal events in a period of time. The final section describes an experiment using Sun's remote uptime service to obtain a sample of the backward occurrence distribution.

3.1 Analyzing `wtmp` log files

On Unix systems that maintain some degree of accounting, a file called `wtmp` is updated automatically by the system. This file stores a record of all logins and logouts. Whenever a user logs in or out of the system, a record is appended to this file noting the terminal the user was logged on, the user name and the time at which the user logged in or out. In addition, as part of the startup procedure on a Unix system, a record of the time at which the machine was started is appended to the `wtmp` log. The `wtmp` files can become very large on heavily used machines, so this file is periodically either deleted or saved in backup storage. By examining a contiguous sequence of `wtmp` logs for a machine and extracting the entries added when the machine is started, we can get a sample of the times at which the machine was started.

As mentioned previously, when each reboot of a host is considered a renewal event, this experiment gives us a sample of the interevent times of the renewal process. We can then use the formulae in section 2.2 to obtain an estimate and confidence interval for the mean interevent renewal time.

We now describe the way in which the `wtmp` log were collected and selected to yield a sample of the interevent times.

3.1.1 Obtaining and selecting the sample `wtmp` logs

We posted a message on the Usenet newsgroup `comp.os.research` describing our experiment, and requested that `wtmp` logs be mailed in to us along with a description of the type of the machine. Replies to this message were the primary source of `wtmp` logs in our analysis. In addition, we obtained a few more `wtmp` logs from hosts in our computer science department and undergraduate computer science resources.

We made two decisions in using the `wtmp` logs to come up with a sample set of observations of times between reboots. First, we decided to use logs that spanned at least two months. This was done to avoid difficulties with analyzing truncated data. If the length of the `wtmp` log is too small, it is possible that the machine was never rebooted during this time. All we can learn from such data is that the machine was continuously running for at least the period of time the wtmp log was maintained, but it does not give us a sample of the time between the time the machine was started. We cannot however choose to simply ignore those logs that do not contain any reboot entries but use others, because then we would artificially favor logs that have many reboot entries and hence bias the data toward hosts with small intervals between reboots.

To avoid encountering this problem altogether, we decided to use only logs that were maintained for sufficiently long periods of time. From previous experiments, two months was considered long enough to avoid the problem of truncated data.

The second decision we took in using the data was to pick one consecutive pair of reboot entries in the `wtmp` log and added the time interval between these two entries into our sample set of observations instead of adding all the intervals contained in a `wtmp` log. The reason we did not add all the intervals in the `wtmp` log was to avoid bias in our sample which arises because not all hosts fail with the same frequency. However, hosts that do fail frequently contribute far more reboot entries in their `wtmp` log than hosts that fail less frequently. If we had counted every single interval in each `wtmp` log, hosts that reboot frequently contribute far more (and far shorter intervals) than hosts that get rebooted less frequently. This biases the sample in favor of hosts that fail frequently. Therefore, to avoid to avoid biasing our sample with excessive contributions from hosts that fail frequently, we took only one sample interval from each host.

3.1.2 Obtaining intervals from the `wtmp` logs

Since `wtmp` logs are binary files, we asked that they be encoded with the `uuencode` program and then electronically mailed to us. Since some mailers cannot handle very large files, many logs were split across different mail messages. In addition, there can be a sequence of `wtmp` files for a machine, representing a consecutive series of backed up versions of the log. This first task was just to get back the original binary file from all the messages we received. This proved very time-consuming, and assembling the original log from all these mail messages was done "by hand", by sorting and saving the messages in the correct order. After running the `uudecode` program to get back the original binary file, any contiguous sequence of `wtmp` files for a host were merged together.

The second task was to analyze this information to determine its viability and obtain boot time entries. Once we had all the `wtmp` files, this task was relatively easy to automate, and was done through a combination of Perl and shell scripts. The consolidated file from a host was analyzed to determine the length of time it covered. This was determined by comparing the first entry in the file with the last. Since `wtmp` files contain the login or logout times of the users of the system, the times contained in the first and last entries in the file bracket the time interval covered by the `wtmp` log. The time interval between this pair of entries is actually slightly less than the time period spanned by the `wtmp` log, since the log begins before the first entry is made into the log, and the log ends after the last entry is made into the log. For reasons described in the previous section If this interval was less than two months, the `wtmp` log was discarded. Otherwise, all the entries representing boot times were extracted and saved. This step gave us a set of boot times for the machine.

Lastly, we randomly picked two consecutive entries from this set of boot time entries for the host and added the time interval spanned by these two points into our sample set of observations. This gives us one sample observation from each log file. As described in the previous section, this step is necessary to prevent bias towards hosts that fail frequently.

We obtained a total of 457 responses to our posting sending in information about 171 hosts. As mentioned earlier, the difference in the number of responses and the number of hosts is because many responses mailed the `wtmp` files in multiple parts, or the `wtmp` logs themselves were in multiple files. We were unable to analyze a few of the logs that were mailed in because of lack of access to a machine of that type. (Each type of machine stores the data in a slightly different fashion.) This weeding process left us with `wtmp` logs for 86 hosts. To get an idea of the characteristics of the `wtmp` logs themselves, each log contained an average of 37.4 boot time entries. The average length of time spanned by a `wtmp` log in the collection we used was 126 days, more than four months. Of course, each host contributes only one time interval to the sample observations, so we had a total of 86 sample intervals.

One observation we made about the data is that it often consists of a “run” of reboot entries at relatively close intervals of a few hours followed by a reboot entry after a long interval of a few weeks. Apparently, it takes the typical machine a few attempts to iron out problems each time the system is brought down. In terms of the distribution of times between reboots, this indicates that the failure rate is very high when the machine is started, and drops down as the machine continues to run uninterrupted.

3.2 The status monitor Experiment

Sun implements a service called the *status monitor* as part of its network services. This service is performed by a daemon called `rpc.statd` that runs using the Sun RPC protocol [17]. This daemon offers Sun RPC based remote procedure calls that can be made to obtain information about the status of the system. The purpose of this service is to provide a generalized way to inform interested hosts about whether a machine is running or not, and also a way of letting clients of the service know when the machine is back on line. This is of importance, for example, in implementing file locking over NFS. A file server may go down between requests for locks on a file. By providing a mechanism of letting clients know when the server is back, lock requests can be reissued and file locking semantics can be preserved over file server crashes. In fact, Sun implements a network locking service with a daemon called `rpc.lockd` that uses the services offered by the status monitor service. The network locking service provides file locking services that work properly in Sun's network file system over both client and server failures.

The basic service performed by the status monitor is to agree to "watch" a set of machines, and report to another set of machines when the "state" of any of these machines changes. A state change occurs when a watched machine goes down and comes back up. (This is actually two state changes – machine goes down, comes back up – but the report is made only when the watched machine restarts.) The state of the machine, which is an integer that keeps track of the number of state changes that have taken place, is returned along with this report. This integer is called the *state number* for the machine. The state number is twice the number of times the machine has been rebooted, as there are two state changes for each reboot – one when the machine goes down and one when it comes back on-line.

The fortunate fact that the service actually records the number of state changes allows us to determine the number of times a machine has been rebooted in a certain interval of time by recording the state number at the beginning and at the end of the interval. This gives us the number of renewal events in that interval of time. We can then use the estimates presented in section 2.4 to obtain an estimate for the mean interevent renewal time.

The next few sections describe in more detail the mechanics of the status monitor service and how we used this to obtain the number of of times the machine was rebooted in a certain interval of time.

3.2.1 Using the status monitor service

The status monitor service provides these remote procedure calls.

1. A call that informs the status monitor on host A, that a particular host, host B is interested in host A's state. This call simply saves the host name of host B into a file on host A. Whenever host A restarts, the status monitor looks up the file and informs the status monitor on each of the hosts in this file that host A's state has changed.

2. A call that sends a host name B as input, asks the status monitor on host A to register a remote procedure callback that exists on the host making the request. The status monitor on host A will then call the callback whenever a state change is reported from host B. This call returns the state number reported from host B.

3. Calls to undo the effects of both these calls.

4. A debugging call to host A that triggers off a "fake" status change report to all hosts that are interested in host A. As part of this operation, the state is incremented by two.

The usual way to use the status monitor service is to first let host A know that host B is interested in host A. Second, register a procedure on host B that the status monitor on host B calls whenever host A reports a status change. Typically, the procedure registered on host B is a local procedure call into an application that is interested in the status of host A.

Our goal was to use the status monitor service to measure the state number twice, separated by a long period of time. The difference is then twice the number of times the machine was restarted in that interval. Our experiment to obtain the state number on a target host consisted of five remote procedure calls that did the following.

1. Tell the status monitor on the target machine that the target machine itself was interested in its status.

2. Ask the target host to call a procedure on our machine whenever a status report comes in from the target machine

3. Send the triggering debugging call to the target host.

4. This sends a status report about the target machine back to itself, which is forwarded to our machine.

5. Unregister all our calls on the target host.

We noticed flaws in the implementation of the status monitor service during the course of our experiment. Sun calls this an intermediate version of the service. For example, the call that lets host A know that host B is interested in it is supposed to return the state number on host A according to the protocol. This functionality does not appear to have been implemented, which made our job of obtaining the state number more difficult. One can also let host A know that a completely unknown or unreachable host called B is interested in its status. When host A restarts, it keeps periodically trying to reach host B, apparently because it thinks host B is down. Rebooting does not help, and the only solution to prevent host A from trying forever is to manually edit the file where the monitor keeps a list of hosts that are interested in its status. We also noticed occasional oddities, when the state number decreased between two measurements. We did not investigate this phenomena, but we believe it might be caused through installing a new version of the operating system during the interim.

3.2.2 Obtaining and selecting the sample

This experiment can be made anonymously, with no special privileges necessary to make and obtain all the information described in the preceding paragraphs. However, the experiment changes data on the machine (the state number changes during our third procedure call to the target machine, when the debugging call is made). There was also the possibility that the status report change could cause transitory delays in the system while the network file locking service completed updating any changes necessary to performed its functions. In view of the nature of this experiment, we decided to ask the permission of the administrators of the hosts involved before running this experiment. Unfortunately, this diminished the size of the data that we could collect.

We first posted a message on several Usenet newsgroups asking for a list of hosts on which we could perform our experiment. After consolidating the list of hosts obtained from the response to our postings, we ran a pilot query on the hosts in this list. We recorded the state number and the time at which the state number was obtained. Some hosts in this list did not run the status monitor server, and we removed those hosts from our collection. Some hosts did not respond or were unreachable, and we also removed these hosts from our sample after querying them once more after an interval of one day.

We repeated the same process after 16 days, this time discarding any hosts that did not respond to our queries. The reason for not querying this set again is because some of the remote procedure calls that we make are implemented over an unreliable communication protocol. Our querying mechanism is also “destructive” in that it increments the state number by 2 when it succeeds. Due to the unreliable nature of the communication protocol, it is possible for the state number to change even when we do not get a response. We repeated this step once more, 39 days from when the experiment was first begun. This process left us with a sample of state numbers from 374 hosts. This sample is slightly biased towards hosts that are highly available, since we discard any hosts that are not available during all three queries. The analysis of this data is presented in the last chapter.

3.3 The `rup` experiment

Sun Microsystems provides a network service called the kernel statistics server (typically implemented by the program `rstatd`.) This service is implemented over Sun RPC, and is much simpler to use than the status monitor service. This service provides information about a variety of statistics about the performance of the kernel like the paging activity, the number of jobs swapped, the number of device interrupts, number of context switches and so on. The kernel statistics service offers a single remote procedure call that returns a structure filled with information about the performance of the kernel. There are two entries in this structure that are of interest to our experiment. One contains the time at which the machine was started and the other contains the current time at that machine. The difference of these two entries is the length of time the machine has been running.

Our experiment consisted of invoking this remote procedure call on the target machine and recording the difference of these two entries. In terms of renewal processes, a renewal event occurs when a machine reboots. Our experiment therefore gives a sample of the backward occurrence distribution of the inter-renewal event time distribution. We can then use the results in section 2.3 to obtain estimates for the mean inter-renewal time, which in this instance is the mean time between reboots.

3.3.1 Selecting and obtaining the sample

Such an experiment was first performed by Long, Carroll and Park [6]. All the data was collected anonymously, and a vast amount of data was retrieved from this experiment. The list of hosts that were queried was obtained by traversing the entire Internet name-tree. Duplicate names were consolidated, and information about the type and operating system of each host was also collected. A subset of this collection of hosts that were likely to be running the network service were chosen. Calls to the network service were made to hosts in this collection to determine the length of time the host was running. We refer to this paper [6] for additional details about the experiment and the conditions under which the data was collected.

In [6], the data from this experiment was used to provide estimates about the mean time to failure of hosts on the Internet. Estimates about the mean time to failure were derived based on the assumption that the lifetimes of machines were drawn from an exponential distribution. The exponential hypothesis was also verified by running a test statistic on the sample data. The data proved to be statistically different from an exponential distribution, especially for large samples.

For this thesis, we performed a similar experiment to collect uptime observations about hosts on the Internet, but instead of information about all the hosts obtained by walking the Internet name tree, we obtained them from the same collection of hosts on which we performed the status monitor experiment. The reason for restricting the size of our sample was to compare estimates obtained through observations of the status monitor experiment through observations from the rup experiment. Each time we obtained the state number for a machine during the status monitor experiment, we also queried and logged information about the length of time the machine was up. Since the state number was queried three times, we had three measurements for each machine. We arbitrarily chose the uptime information returned at the last time the experiment was performed in deriving estimates from this experiment.

As described in section 2.4, we analyzed this data in four different ways to reach estimates about the mean time to failure. The first estimate is identical to the estimate made in [6], and assumes that the underlying data follows an exponential distribution. The formulae used in this estimate are presented in section 2.4.1. Since previous experiments show that the data is probably related to the exponential distribution, we used two more estimates that assume that the data follow two generalizations of the exponential model – the Gamma distribution and the Weibull distribution. The formulae used in this estimate are presented in section 2.4.2 and 2.4.3 respectively. We also ran the Kolmogorov-Smirnoff goodness-of-fit test to determine how closely the model actually fit the data. Finally, we analyzed the data in a distribution-free fashion by looking at the histogram of the data generated. This method is described in section 2.4.4. Obtaining these estimates using the results in Chapter

2 is straightforward, and the formulae and algorithms used are presented in the Appendix. The results of these four analyses are presented in Chapter 4.

Chapter 4 Summary

In this chapter, we present the estimates derived based on the observations and calculations from the experiments described in the previous chapters. We compare the different estimates that were obtained, and how the assumptions made during the derivation of the experiments affects the estimate. We also describe some work in progress to obtain more reliable observations and estimates about hosts on the Internet, and finally present some future directions in which this work can be continued.

To present these results in the context of renewal processes, we wish to measure mean inter-event renewal time of a process where renewal events occur whenever a host is rebooted. We have three different observations of this renewal process

1. A sample of the inter-event renewal times (the `wtmp` experiment),
2. A sample of the number of renewal events in a certain time interval (the status monitor experiment), and
3. A sample of the backward occurrence distribution of the process (the `rup` experiment.)

Chapter 2 presents results from statistics that can be used to estimate the mean inter-event renewal time from such observations. Chapter 3 describes how these observations were actually selected and obtained. The next three sections presents the results of applying the formulae in Chapter 2 to the observations obtained in Chapter 3.

4.1 Estimates from the `wtmp` experiment

The **`wtmp`** experiment provides us with the smallest number of sample observations, 86 sample points. The small sample size is due to the difficulty of collecting the samples, as it requires the cooperation of administrators of the hosts on which the sample was taken. Secondly (as described in Chapter 3) to eliminate the problem of truncated data samples, only sufficiently large log files were used in obtaining sample points. Lastly, to avoid bias towards hosts that get rebooted frequently, only one measurement per log file was used. In spite of the small size of the sample, this measurement is the most direct observation of the quantity we wish to estimate (the inter-event renewal time) as it actually records the times at which the machine was rebooted. Therefore we can expect this estimate to be a very good indicator of the actual mean time between reboots of the hosts that were sampled. However the small sample size also means larger confidence intervals for the estimate, and the possibility that the sample is not truly representative of hosts on the Internet.

The following table summarizes the estimates obtained from the **`wtmp`** experiment for the *mean time between reboots*. The mean time between reboots is estimated as a little

TABLE 4.1: Mean time between reboots from the **`wtmp`** experiment

n	$\bar{x}$ (days)	90% confidence interval	σ	min	median	max
86	11.336854	8.415 14.259	16.344978	0.004	4.701736	86.561

over 11 days. The small sample size leads to a large 90% confidence interval for the mean, which lies between between 8 and 14 days. In comparison, estimates from Carroll, Long and Park [6] are significantly higher. For instance, the estimate for the mean time to reboot for Sun4/60 machines is about 18 days. Other systems too, are estimated to have similar mean times to failure. One explanation for this discrepancy is from the following interesting observation about our data. There is a very small value for the median, which for our data is a little under 5 days. The next two graphs shed more light on the distribution of the data obtained from this experiment. The first graph is a graph of the *survivor function* of the data. In other words, this is a graph of the probability of surviving for at least a certain time T, which is just $\mathrm{P}[t > T]$ against time T. The second graph is a plot of the log of the

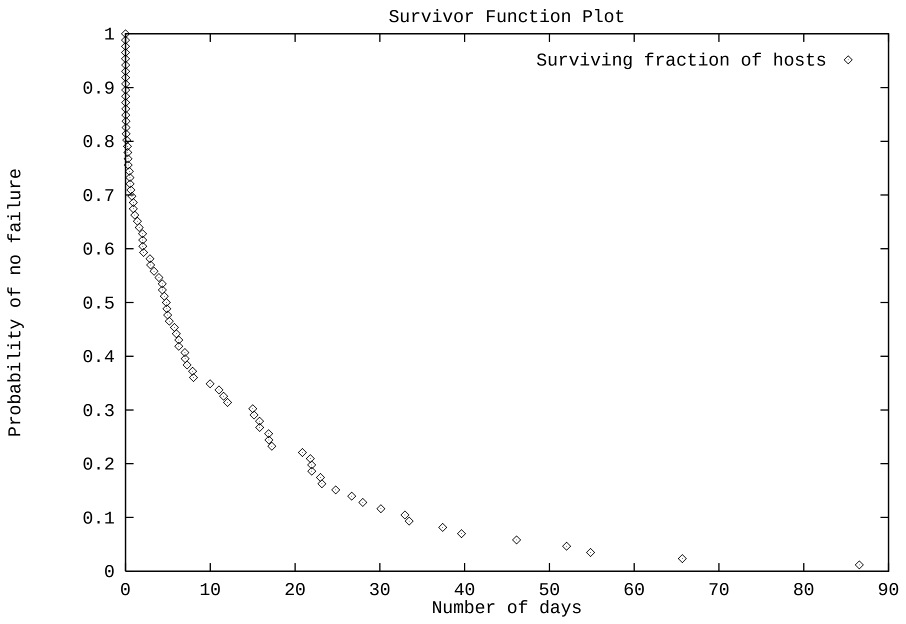


FIGURE 4.1: Survivor function plot from the **wtmp** experiment

survivor function $\log(\mathrm{P}[t > T])$ against time T. The second graph gives us some insight into the failure rate of the distribution. The failure rate is $-\frac{d}{dx}\log(\mathrm{P}[t > T])$. In other words, the failure rate is proportional to the slope of this graph. An exponential distribution has a constant failure rate, so the log survivor function plot for data following an exponential distribution would be a straight line. In the graph that we actually observe, there is a steep slope in the very initial portions of the graph indicating regions of high failure, and then becomes nearly straight in the rest of the graph, indicating an exponential-like constant failure rate. The distribution of the data from the **wtmp** data indicates that the underlying

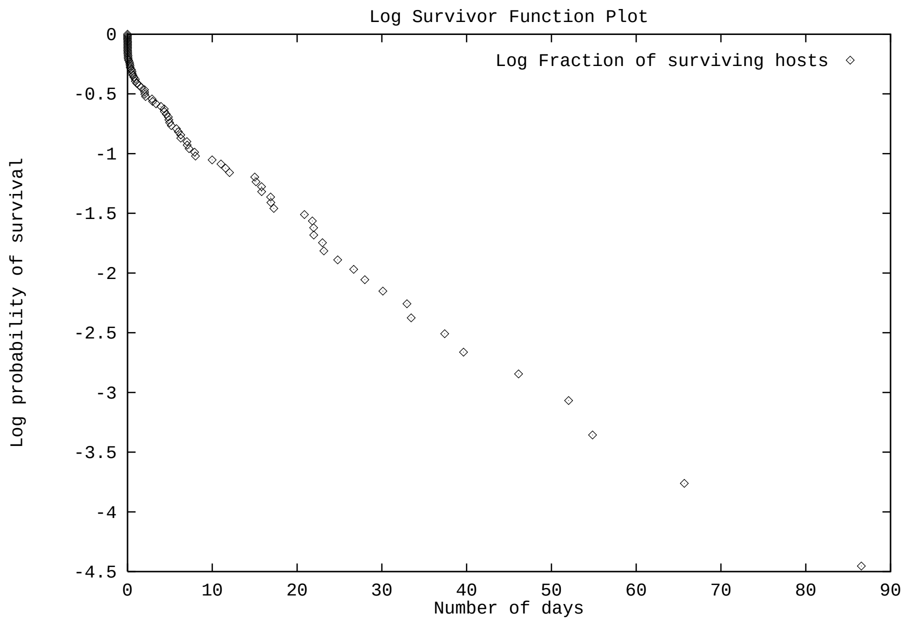


FIGURE 4.2: Log survivor function plot from the `wtmp` experiment

distribution for the time between reboots is one that has a high failure rate for small time intervals, and a nearly constant failure rate for larger time intervals.

The estimate from [6] assumes the underlying distribution is exponential, and therefore, that the backward occurrence distribution is also exponential. However, from our observation of the sample, the underlying data does not seem to be exponential, and we believe that part of the discrepancy in the two estimates comes from this fact. Of course, these two estimates come from different sample populations of the Internet, which could also contribute to the discrepancy.

4.2 Estimates from the status monitor experiment

The status monitor experiment obtains the number of renewal events in an interval of time. The results in section 2.4 then allow us to estimate the mean of the inter-event renewal time. The status monitor experiment gives more direct observations about the mean time between reboots of a machine than the uptime experiment as it allows us to make distribution free estimates about the mean. However, unlike the `wtmp` data, it is not possible to make any inferences about the distribution of the underlying distribution due to the nature of the observation.

The status monitor and the remote uptime experiments are based on a slightly larger sample of 374 hosts. As described in Chapter 3, even though the data could be collected anonymously, it changes some data on the target host. Therefore the sample was limited to those who agreed to let the experiment be performed on their machines. We obtained more responses for this experiment, perhaps because unlike the `wtmp` experiment, the only thing they needed to do was mail in a message agreeing that the experiment be performed.

The geographical distribution of the sample is more widely distributed than the `wtmp` experiment since a larger group of volunteers agreed to let their hosts be used in the experiment. This tends to make this experiment more indicative of the behavior of the Internet as a whole, but the sample size is still rather small. Moreover, it is rather "clustered" data in that a volunteer would typically agree that a set of machines in a particular institution could participate in the experiment.

As described in Chapter 3, we took three measurements of the state number, and only used those machines that responded to all three requests. In making the estimate, only the state numbers from the first and last observations were used. The time difference between these two measurements were identical for all the hosts to within 5 minutes. We used the results in section 2.4 to compute the estimate for the mean of the interevent renewal period. The appendix also contains the formula used to arrive at the mean and the confidence interval for the mean.

TABLE 4.2: Mean time between reboots from the status monitor experiment

n	$\bar{x}$ (days)	90% confidence interval	
374	11.240	10.179	12.548

The estimate from this experiment predicts a mean time between reboots of just over 11 days. This is gratifying since it corroborates the prediction from the prediction from the `wtmp` experiment. This two experiments measure different characteristics about hosts on the Internet, and are taken on different samples. The fact that these two different experiments estimate about the same value for the mean time between reboots indicates that our estimate may not be too far from the truth.

4.3 Estimates from the `rup` experiment

The uptime experiment was performed on the same hosts that participated in the status monitor experiment. The same caveats that apply to the sample used in the status monitor experiment apply to this sample too. Machine uptime records were taken each time the state number was determined. Since the state number was obtained thrice, there were three uptime samples for each host. The first of these three observations was arbitrarily used to estimate the mean time to failure for these machines. This data is easily obtained, but hard to analyze without making assumptions about the underlying data. In all, we attempted to make four different estimates from the data in this experiment. Only one of them also provides confidence estimates, but it makes strong assumptions about the distribution of the underlying data, namely that it is exponential in nature. Two others assume that the data is a generalization of the exponential distribution, but do not yield confidence estimates. In addition to estimating the mean time to failure, we also test the goodness-of-fit of the data to the proposed distribution by running the Kolmogorov-Smirnoff test on the actual and proposed distribution. The fourth estimate is a different way of analyzing the data, by using histograms. This estimate does not yield confidence estimates either, but is presented as an interesting distribution-free method of obtaining the mean time to failure.

The first table are the estimates derived from the assumption that the distribution is governed by an exponential distribution.

TABLE 4.3: Mean time to failure from the `rup` experiment (exponential assumption)

n	$\bar{x}$ (days)	90% confidence interval		D_n
374	14.725	13.049	16.401	0.056558

The statistic for the Kolmogorov-Smirnoff test (the value of D_n) is interesting. We accept the hypothesis that the distribution is truly exponential at about a level of significance of 0.18. This suggests that the data is quite close to being an exponential distribution.

Assuming that the data follows the Gamma distribution, the method of moments for computing the parameters for the Gamma distribution on our sample data led to an equation with an impossible value for λ (which was smaller than 0), so the estimates could not be derived. This is probably caused due to the small sample size.

Assuming the data follows the Weibull distribution, we reach the following estimates for the mean time to failure.

TABLE 4.4: Mean time to failure from the `rup` experiment (Weibull assumption)

n	$\bar{x}$ (days)	α	λ	D_n
374	13.922	0.953791	0.082784	0.050227

Assuming that the data follows a Weibull distribution leads to a lower estimate for the mean time to failure. The value for the parameter α is of interest as it is smaller than 1. This indicates a decreasing failure rate, which ties in with our observation about the failure rate from the `wtmp` experiment. Since the uptime and `wtmp` experiment are two different observations about hosts, this provides independent evidence that the failure rates are indeed high for small time intervals. The data also indicates that the distribution is closer to being a Weibull distribution than an exponential distribution, and it is accepted at a level of significance of about 0.25.

Finally, the histogram method of estimating the data was used to estimate the mean time to failure. The bin size was chosen to be one day, and the histogram of samples was generated. The fraction of samples in the first bin divided by the bin width gives us an estimate (the histogram estimate) for the density function of the of the backward occurrence variable at the origin. (Chapter 2 has more details on the histogram method of estimating the data). Since the value of the density function of the backward occurrence variable at the origin is $1/\mu$, this is an estimate for the inverse of the mean. The graph of the histogram generated is also shown below. This method estimated the mean time to failure as 13.32 days.

4.4 Comparison of the estimates

All three experiments lead to unexpectedly low estimates on the mean time to failure for hosts on the Internet. These numbers need to be viewed in light of the fact that any time a machine is shut down, it is assumed to be a failure. Three different observations about hosts on the Internet lead to a total of five estimates about hosts on the Internet. All these estimates predict a time of between 11 and 14 days.

The `wtmp` experiment is the most reliable indicator of the true time between reboots, as it contains exact records of when the machine was started. The `wtmp` experiment also predicts a low average time between reboots, and is probably caused by the number of "false starts" that seem to occur each time a system is rebooted. However, the sample size is rather small, and tends to be clustered around a few sites. With a larger sample size, this method can be expected to yield more accurate estimates. The `wtmp` experiment also lets us observe the distribution of the data, and it appears to be fairly exponential in nature except for small values of time, when the failure rates are very high.

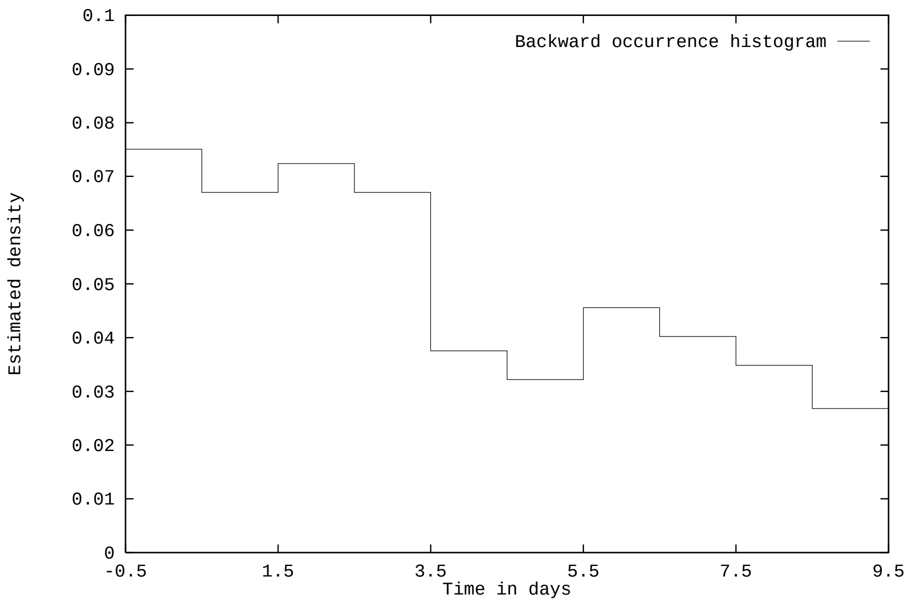


FIGURE 4.3: Histogram estimate of the density function from the `rup` experiment

The status monitor experiment becomes more reliable as the duration of the experiment is increased. Observations from this experiment predict a mean time between failures that are comparable with those predicted by the `wtmp` experiment. Both these experiments are reliable, in that they sample exactly the quantity that needs to be determined, but in different ways. Therefore it is not surprising that they lead to similar estimates about the mean time between reboots of the machine. These two experiments also provide separate and independent evidence about the mean time between reboots of hosts on the Internet.

The remote uptime experiment requires more careful analysis, and estimates tend to be based on assumptions about the distribution of the data. Unfortunately, assuming anything more complicated than an exponential distribution leads to difficulties in estimating parameters of the distribution and in obtaining confidence estimates. Estimates based on this experiment appear to be higher than those predicted by either the `wtmp` or the status monitor experiment. On the same set of hosts, the more reliable status monitor experiment predicts a slightly lower mean time between failures than the estimates from the remote uptime experiment. This appears to be a direct consequence of the high initial failure rate, and the inability of the models assumed in deriving estimates based on the uptimes to adequately capture this distribution. However, for the small sample sizes considered in this thesis, both the exponential and the Weibull model were good fits to the observed data, and the Weibull model fits the data a little better than the exponential model. This is to be expected, as the Weibull model is a generalization of the exponential model. The predicted Weibull model also had a decreasing failure rate, which validates our observations about the high initial failure rate of the observed data based on the `wtmp` experiment.

In comparison with the estimates made in [6], all of our estimates yield lower mean times between failure. However, it is interesting to note that same method used in [6] (the `rup` experiment with the exponential hypothesis) produced an estimate that was closest to those in [6]. The fact that they are still different is probably due to the fact that these two experiments were taken on different sample populations.

4.5 Future directions

The ideal observation about hosts on the Internet would involve continuous monitoring to accurately track failures. Work is in progress on continuously monitoring hosts on the Internet without generating a voluminous amount of traffic. A prototype of a monitoring system called the *tattler* has been implemented. This monitoring system is described in [13]. A *tattler* is a monitoring station that periodically gathers information about a subset of hosts on the Internet. A tattler also periodically contacts another random tattler in the system and exchanges information that it has collected about hosts. By having different tattlers monitor a particular host, the load generated on the network by the monitoring system is spread out. Moreover, interruptions in monitoring a host caused because of network partitions can be reduced since more than one tattler can monitor a particular host. A system of tattlers behaves essentially as a distributed database of information. The consistency of this database in the face of failures of tattlers is ensured by exchanging information sufficiently often, through a protocol called the *time-stamped anti-entropy* protocol [10]. This protocol guarantees that the database is "eventually consistent." This means that if all monitoring were to stop at any particular instant, (but tattlers still continue to contact one another) the probability of any two tattlers having non-identical databases approaches zero. The protocol is robust in the face of transient failures of tattlers, and is also able to detect when all the tattlers in the system have an identical collection of data in their database. A prototype of this system has been monitoring hosts anonymously by periodically checking if the host responds to a ping message. It also verifies if a host has been rebooted since the last time the host was queried by making an uptime call and comparing the boot times retrieved from the last query. This system is intended to provide us with a large-scale anonymous method of collecting information about hosts on the Internet that can be used to derive estimates about the time to failure of hosts without making any assumptions about underlying distributions.

Another promising area to focus efforts is to develop a model to predict the time to failure of hosts on the Internet. The simple exponential model is useful, but not completely accurate. The Weibull model is a little more accurate, but is hard to analyze when used in trying to predict the behavior of distributed systems. It would be interesting to explore other models to adequately summarize the data that we observed. Lastly, if the model of the application whose reliability we are trying to predict is already being solved numerically, the data collected through out experiments can be used directly as a source of time to failures of hosts.

Appendix

This section presents the formulae and algorithms used to arrive at the various estimates presented in the thesis. The theory behind these formulae and algorithms are presented in Chapter 2.

Sampling the interevent renewal time

Given: A sample of N observations $x_1, x_2, \ldots, x_N$ of the interevent renewal time of a process and a confidence level $(1 - \alpha)$.

Output: A $(1 - \alpha)$ confidence interval for the mean interevent time of the renewal process.

Algorithm:

1. Determine η (from statistics tables) such that $\mathrm{P}(T > \eta) = \alpha/2$, where T is a Student T distribution with $N - 1$ degrees of freedom.
2. Compute $\overline{X} = (\sum_{i=1}^{N} x_i)/N$
3. Compute $S = \sqrt{\frac{\sum_1^N (x_i - \overline{X})^2}{N-1}}$
4. Output the interval $(\overline{X} - \eta\frac{S}{\sqrt{N}}, \overline{X} + \eta\frac{S}{\sqrt{N}})$

Sampling the backward occurrence

This section describes all the formulae used in obtaining estimates for mean of the renewal process. In all these cases, we have the following scenario.

Given: A sample of N observations $x_1, x_2, \ldots, x_N$ of the backward occurrence variable of a renewal process.

The exponential model

Here we assume that the renewal process follows the exponential hypothesis, and that we are given a confidence level $(1-\alpha)$

Output: A $(1-\alpha)$ confidence interval for the mean interevent time of the renewal process.

Algorithm:

1. Determine η (from statistics tables) such that $\mathrm{P}(X > \eta) = \alpha/2$, where X is a chi-squared distribution with $2N$ degrees of freedom.
2. Compute $\overline{X} = (\sum_{i=1}^{N} x_i)/N$
3. Output the interval $(\overline{X}\frac{2n}{\eta}, \overline{X}\frac{2n}{\eta})$

The Weibull model

In this algorithm, we assume that the renewal process follows the Weibull distribution.

Output: An estimate for the mean interevent time of the renewal process.

Algorithm:

1. Pick a random positive value α, and a small error tolerance value ϵ.
2. Compute $\alpha' = \frac{\sum_{i=1}^{N} x_i^\alpha}{\sum_{i=1}^{N} x_i^\alpha \log x_i}(\Psi(1+1/\alpha) + \log(\frac{\alpha}{N}\sum_{i=1}^{N} x_i^\alpha))$. Here, $\Psi(x)$ is the derivative of $\log ?(x)$.
3. If the absolute difference between α' and α is larger than ϵ, set $\alpha = \alpha'$ and repeat the previous step. Otherwise, proceed to the next step.
4. Compute $\lambda = \frac{N}{\alpha \sum_{i=1}^{N} x_i^\alpha}$
5. Output $\frac{1}{\lambda}^{\frac{1}{\alpha}} ?(1 + 1/\alpha)$

The Gamma distribution

In this algorithm, we assume that the renewal process follows a Gamma distribution.

Output: An estimate for the mean interevent time of the renewal process.

Algorithm:

1. Compute $\overline{X} = (\sum_{i=1}^{N} x_i)/N$

2. Compute $S = \sqrt{\frac{\sum_1^N (x_i - \overline{X})^2}{N-1}}$
3. Compute $\lambda = \frac{2\overline{X}}{3S - \overline{X}^2}$
4. Compute $\alpha = \frac{3S - 5\overline{X}^2}{\overline{X}^2 - 3s}$
5. Output α/λ.

The histogram estimate

The estimate is computed by estimating the probability density function of the renewal process at the origin using a histogram.

Algorithm

1. Without loss of generality assume the samples are sequenced in increasing order. Define the bin width b to be $\log_2(x_N - x_1)$.
2. Compute n_0, the number of samples that fall in the interval $[-b/2, b/2]$. This is the number of entries in the bin centered around the origin.
3. Let $p_0 = n_0/(bN)$. This is the histogram estimate of the pdf at the origin.
4. return $1/p_0$. This is the estimate of the mean of the distribution f.

Counting the number of renewal events

Given: A sample $n_1, n_2, \ldots, n_N$ of the number of renewal events in a time interval L, and a confidence interval $(1-\alpha)$.

Output: A $(1-\alpha)$ confidence interval for the interevent renewal time.

Algorithm

1. For each i in $\{1, 2, \ldots, N\}$, compute $t_i = L/n_i$
2. Determine η (from statistics tables) such that $\mathrm{P}(T > \eta) = \alpha/2$, where T is a Student T distribution with $N-1$ degrees of freedom.
3. Compute $\overline{T} = (\sum_{i=1}^N t_i)/N$
4. Compute $S = \sqrt{\frac{\sum_1^N (t_i - \overline{T})^2}{N-1}}$
5. Output the interval $(\overline{T} - \eta\frac{S}{\sqrt{N}}, \overline{T} + \eta\frac{S}{\sqrt{N}})$

The Kolmogorov-Smirnoff test

The Kolmogorov-Smirnoff test is a measure of how closely a hypothesized distribution actually follows the sample that is obtained.

Given: A set of observed sample points $x_1, x_2, \ldots, x_N$ and a proposed cumulative distribution function $F_0(x)$ to explain the sample points.

Output: The D_N statistic for the data.

Algorithm

1. For each x_i, compute $m_i = \max_{x_i \leq x < x_{i+1}} |i/N - F_0(x)|$. Take $x_{N+1} = \infty$
2. Compute $m_0 = \max_{x < x_1} |1/N - F_0(x)|$
3. Output the maximum of all the m_i

Bibliography

[1] P. J. Bickel and K. A. Doksum. *Mathematical Statistics* (1977). Holden-Day Inc.

[2] F. K. C. Pu and R. C. Lehman. A Measurement Methodology for Wide Area Internets. Tech. Rep. CUCS-044-90 (Mar 1991). Columbia University.

[3] J. L. Carroll and D. D. E. Long. The Effect of Failure and Repair Distributions on Consistency Protocols for Replicated Data Objects. *Proceedings of the Twenty Second Annual Simulation Symposium* (Tampa), pages 47-60 (Mar 1989).

[4] D. R. Cox. *Renewal Theory* (1962). Methuen and Co.

[5] D. R. Cox and P. A. Lewis. *The Statistical Analysis of Series of Events* (1966). Chapman and Hall.

[6] D. D. E. Long, J. L. Carroll and C. J. Park. A study of the reliability of Internet sites. *Proceedings of the Tenth Symposium on Reliable Distributed Systems* (Pisa, Italy), pages 177-186 (Sep 1991).

[7] K. A. Doksum and B. S. Yandell. *Handbook of Statistics*, volume 4 (1984). Elsevier.

[8] W. Feller. *Introduction to Probability Theory*, volume II (1972). John Wiley and Sons.

[9] B. V. Gnedenko. *Mathematical Methods in Reliability Theory*, Moscow: English Translation (1968). Academic Press.

[10] R. A. Golding. Weak-consistency Group Communication and Membership. Tech. Rep. UCSC-CRL-92-52 (Dec 1992). University of California at Santa Cruz.

[11] J. Gray. A Census of Tandem System Availability Between 1985 and 1990. Tech. Rep. 90.1 (Jan 1990). Tandem Computers.

[12] W. Hardle. *Smoothing Techniques* (1991). Springer-Verlag.

[13] D. D. E. Long. A Replicated Monitoring Tool. *Proceedings of the Second Workshop on the Management of Replicated Data* (Monterey), pages 96-99 (November 1992).

[14] D. D. E. Long, J. L. Carroll, and K. Stewart. Estimating the Reliability of Regeneration-Based Replica Control Protocols. *IEEE Transactions on Computers*, **38**(12):1691-1702 (Dec 1989).

[15] Mark Lottor. Internet Domain Survey. Technical report (Jan 1993). Network Information Systems Center, SRI Intl.

[16] S. J. R. Caceres, P. B. Danzig and D. J. Mitzel. Characteristics of Wide-Area TCP/IP Conversations. *Proceedings of the ACM SIGCOMM* 91, pages 601-612 (1991).

[17] Sun Microsystems, Incorporated. RPC: Remote Procedure Call Protocol specification version 2. Tech. Rep. RFC-1057 (Jun 1988). USC Information Sciences Institute.

[18] K. S. Trivedi. *Probability & Statistics with Reliability, Queuing and Computer Science Applications* (1982). Prentice-Hall.

[19] M. F. Schwartz P. G. Tsirigotis. Experience with a Semantically Cognizant Internet White Pages Directory Tool. *Journal of Internetworking Research and Experience*, pages 23-50 (Mar 1991).

[20] M. F. Schwartz D. C. M. Wood. A Measurement Study of Organizational Properties in the Global Electronic Mail Community. Tech. Rep. CU-CS-482-90 (Aug 1990). University of Colorado.