Is MTTR More Important Than MTTF For Improving User-Perceived Availability?

Yee Jiun Song, Wendy Tobagus, Jeff Raymakers, Armando Fox

Computer Science Department, Stanford University

353 Serra Mall, Stanford, CA 94305, USA

Email: {yeejiun, tobagus, jeph, fox}@cs.stanford.edu

Abstract

High availability of internet systems can be achieved either through high mean-time-to-failure (MTTF) or low mean-time-to-recovery (MTTR). Traditionally, system designers have focused on maximizing MTTF as a way of providing high availability. However, recent work in recovery-oriented computing has emphasized recovery from failures, rather than failure avoidance [1]. In this paper, we investigate the impact of MTTR and MTTF, as well as user retry rates, on the user-perceived availability of internet systems through simulation experiments. Previous work suggests that between two systems with a given availability, the one with lower MTTR always yields higher user-perceived availability [3]. Our results show that while this is true for certain ranges of system parameters, it is not always the case. This leads us to propose a method for determining whether improving MTTR or MTTF will yield greater user-visible benefits for a given system, allowing a system administrator to invest in system improvements that yield maximum benefits.

I. Introduction

Maintaining a high level of availability is important to many internet systems. In fact, in commercial systems, downtime costs anywhere from $89 thousand an hour, in airline reservation systems, to $6 million an hour, for brokerage operations [2]. As such, much research has been devoted to ensuring a high level of availability of such systems. The availability of a system is given by equation 1, which shows that high availability can be achieved either through a high mean-time-to-failure (MTTF), a low mean-time-to-recovery (MTTR), or a combination of both. Traditionally, system designers have focused on maximizing MTTF. However, recent work in recovery-oriented computing [1] has taken a different approach, and emphasized fast recovery from failure, rather than failure avoidance.

$$Availability = \frac{MTTF}{MTTF + MTTR} \tag{1}$$

Most work on improving availability has focused on the system's view and not the user's view of availability. For web services, these two quantities can be different. To see why, observe that users only perceive availability or unavailability when they actually access the site, and not during the interstitial or "think" times between page views. If the site goes down, a given user may not see the failure right away (i.e. until the next page view); similarly, if the user experiences a site error and the site comes back up, the user may not notice this until the next time she actually tries to access the site. Therefore, in order to study user-percevied availability, it is crucial to separate the two factors that contribute to it: actual system availabilty and user behavior.

Recent work by Xie et al. applied Markov regenerative process models to study user-perceived availability of internet systems and found that (i) user-perceived availability depends not only on the system's characteristics, but also on the user's behavior, and (ii) for two systems of a certain availability, the one with the lower MTTR provides higher user-perceived availability [3]. Clearly then, ensuring that a system has high availability is not sufficient; it is also essential to study the *composition* of that availability, and its impact on the system's user-perceived availability.

In this paper, we use simulations to verify the trends found in Xie et al's mathematical analysis. We use a wide range of system parameters and find that under certain conditions MTTR is indeed more important that MTTF for providing high user-perceived availabilty, but under certain other conditions, the converse can also be true. Intuitively, as MTTR decreases, the decreasing marginal gains of lower MTTR eventually catches up with the marginal cost of lower MTTF, and it becomes more important to improve MTTF. Based on our results, we propose a method for deciding, for a given system, which of MTTR or MTTF to improve, to extract maximum gains in user-perceived availability.

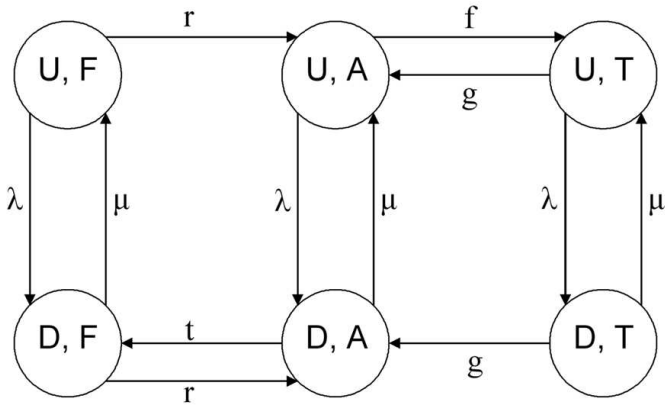
The rest of this paper is organized as follows: section II explains the user-server interaction model that is used throughout this paper; section III presents the experimental setup and procedures; IV describes the results of our studies; V proposes a method for determining whether an improvement in MTTR or MTTF will yield greater user-visible benefits; section VI discusses related work; section VII suggests possible directions for future work; and finally section VIII concludes this paper.

II. User and Server Models

A. User-Server Interaction Model

There has been much work on the characteristics of web traffic and web users' behavior pattern [4]–[8]. While the results of these studies exhibit conflicts in some respects, they agree that web user behavior exhibit strong self-similarity, and is best approximated by heavy-tailed distributions such as the Weibull or Pareto distributions. In our study, we use the user model first proposed by Deng [8], and used by Xie et al. in [3]. This ON-OFF web

Fig. 1. User-Server Interaction Model



user model was first proposed by Deng [8] based on the examination of empirical web data, and a similar model was independently proposed by Crovella and Bestavros [9]. In this section, we provide a brief presentation of the model. For details and an explanation of the model's derivation, please refer to [8].

Figure 1 shows a state machine that represents the user-server interaction model. Each web user can be in one of three states, $\Omega_U = \{A, T, F\}$, which represents the user in an active state, in a thinking state, and seeing a failure, respectively. The state of the server is represented by $\Omega_S = \{U, D\}$, representing the server when it is available, and when it has encountered some error and is unavailable, respectively. Thus, at any one time, the state of the entire model is represented by a combination of Ω_S and Ω_U. This model is identical to the one used in [3].

When the user is in an active state, she actively makes requests to the server. This includes multiple requests made by the browser as a result of loading a single webpage, or multiple webpages that the user loads in a short period of time. The thinking state represents the user when she is thinking, or reading the results of previous requests. During this time, there are no requests made by the user. To represent server failures, the model includes a third user state to represent a user who has seen a failure in the server, and is waiting to retry her request.

$$f(t) = \frac{k}{\theta}(\frac{t}{\theta})^{k-1} e^{-(\frac{t}{\theta})^k} \tag{2}$$

$$g(t) = \begin{cases} \frac{C\alpha m^\alpha}{t^{\alpha+1}}, & m \leq t \leq n \\ 0, & otherwise \end{cases} \tag{3}$$

When the user is in active state, she makes requests to the server at a rate that is described by the Weibull distribution with parameters $k = 0.5$ and $\theta = 1.5$. The probability density function of the Weibull distribution is given by equation 2. If the user encounters no errors while in active state, she goes into thinking state after a certain

amount of time. This time is described by another Weibull distribution, with $k = 0.88$ and $\theta = e^{4.5}$. Once the user is in thinking state, she stays in that state until she is ready to make requests to the server again. The amount of time spent in thinking state is described by the Pareto distribution, with parameters $\alpha = 0.5$, $m = 60$, and $n = 6000$. The probability density function of a Pareto distribution is given by equation 3. Note that $C = \frac{1}{1-(\frac{n}{m})^{\alpha}}$ is the normalization factor. As mentioned earlier, this model, as well as the distribution parameters, are identical to those proposed and used by [8] and [3] respectively. This behavioral model of the user creates a difference between the user-perceived availability of the system, and its actual availability. When the system goes down, the user may not see the failure right away; conversely, when the system comes back up, the user may still think the server is down for some time.

The server has two states: up and down. We assume that both MTTR and MTTF have exponential distributions. Thus, as shown by Figure 1, the server fails at some rate λ, and recovers at some rate μ. In our experiments, we vary the values of λ and μ. If the web user encounters an error while in active state (server does not respond), the browser retries for a period of $t = 10$ seconds. This is intended to model HTTP retry time most browsers have. If the server recovers within that time, the user does not see the failure. Otherwise, the user goes into failure state, and stays in failure state for a certain amount of time before going back into active state to retry her request. The amount of time spent in failure state is assumed to be an exponential distribution with a rate r, the user retry rate. We discuss this user retry behavior in detail in the following section. A summary of the distribution parameters used in our model is shown in Table I and Table II.

TABLE I
SUMMARY OF FIXED SIMULATION PARAMETERS

Parameter	Default Value	Comment
k	0.88	Shape parameter of Weibull distribution
θ	$e^{4.5}$	Scale parameter of Weibull distribution
		(for the length of ON period)
k'	0.5	Shape parameter of Weibull distribution
θ'	1.5	Scale parameter of Weibull distribution
		(within the ON period)
α	0.5	Shape parameter of Pareto distribution
m	60s	Scale parameter of Pareto distribution
n	6000s	Truncation point of Pareto distribution
		(ON-OFF threshold)
T	10s	HTTP retry time

TABLE II
SUMMARY OF VARIABLE SIMULATION PARAMETERS

Parameter	Default Value	Range	Comment
r	100 s	100 s to 20 min	Mean time between user retries upon failure
T_R	2400 s	17 s to 1.5 hours	Duration of failure (TTR)
T_F	3.9 days	40 min to 8.6 days	Mean time before failure

B. User Behavior In The Face of Failure

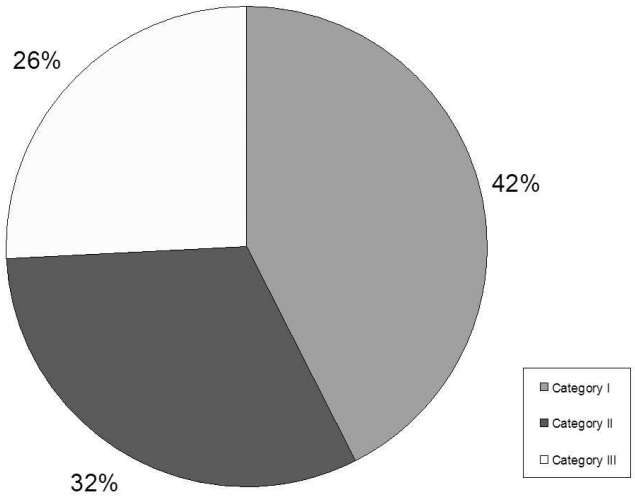
In our experiments, we chose to assume an exponential distribution of the time between user retries when users detect a failure in the server. This is to allow us to compare our results with previous work by Xie et al. However, we note that there are no existing empirical studies which support the use of an exponential distribution, or any other distribution, to model users' behavior in the face of a server failure. In this section, we investigate the plausibility of our assumption by examining a set of web server logs from a certain commercial shopping portal website. We collected web server logs for a particular page on this website for the duration of an entire day, during which there is a two hour period, 10 AM to 12 PM, of complete system failure. After the two hours, the system was restarted but was still suffering from intermittent problems. Although this data is not sufficient to deduce a complete user model for user retry behavior, we are able to identify certain trends.

From our logs, we extract all the cookied requests because they allow us to identify unique users. For the two hour failure period, 1015 unique users requested service from that page. We classify those users into different categories: I) users seen only once during the two hour period, and never again for the rest of the day; II) users seen more than once in the two hour period, but never again for the rest of the day; and III) users who were seen during the two hour period, and again at some point after that period. This is shown in Figure 2.

We observe that a large number of users, the ones in categories I and II, give up before the fault is repaired. In particular, users in category I are especially unforgiving - never coming back to the site for the rest of the day after experiencing one failed request. This may be because of the nature of the web site we're studying. Since the website in question is a shopping portal, users may simply go elsewhere if they encounter problems trying to access it. We speculate that user behavior will be different for a web site that offers critical service that is not available elsewhere. An example of such a web service is online banking.

For the users who do retry multiple times (category III), we plot the times between retries in a cumulative frequency distribution, as shown in Figure 3. We compare the actual cumulative frequency distribution with an exponential distribution, with a mean time between retry of 4500s, and observe that the empirical distribution has a significantly heavier tail than the exponential distribution. However, it appears that the exponential distribution is

Fig. 2. User Failure Behavior



not an unreasonable approximation of the empirical data.

We emphasize that the results presented in this section are preliminary, and meant only to provide some intuition into user retry behavior in the real world. The development of a complete user model based on empirical data from multiple sites is the subject of future work.

Given that there is no existing model for user retry behavior in the face of failure, we make the assumption that user retry behavior is approximated by an exponential distribution for all of our experiments. However, to demonstrate that our results are dependent on user retry behavior, we also experimented with an alternative model. Anecdotal evidence from our colleagues suggests that exponential backoff may be a suitable approximation of user behavior under certain circumstances. The results from this alternative model are presented in section IV-C.

III. Experimental Setup

A. Software and Hardware Components

Figure 4 is a block diagram of the experiment setup. On the left are simulated users. Each simulated user is implemented as a thread in a Python application. All the user threads were run on a single Pentium III 1.1 GHz Windows XP machine equipped with 256 MB of RAM. A proxy server runs on the same machine. The proxy is responsible for forwarding web requests from the users to the web server when it is in an "up" state, and simulating server failure by dropping requests when it is in a "down" state. The web server is an Apache 2.0.40 server running on a vanilla RedHat 8.0 installation of Linux. It ran on a PC with an AMD Athlon 1800XP processor and 512 MB of RAM. On all our experiments, we monitored CPU utilization on both machines to ensure that the processor

Fig. 3. Cumulative Distribution Frequency of User Retry Time

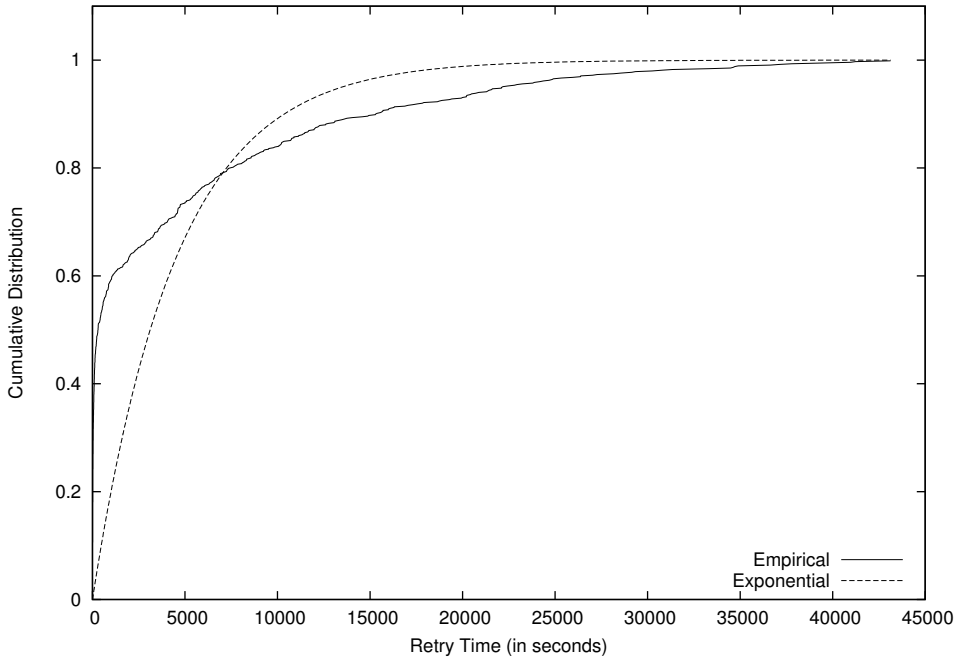
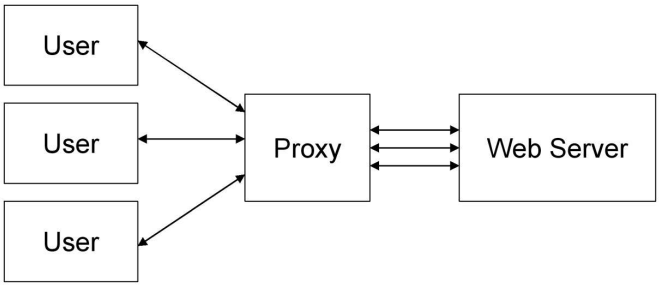


Fig. 4. Experiment Setup



was not the bottleneck. During our experiments, CPU utilization remained below 50% on the machine running the simulated users and proxy, and below 10% on the web server.

B. Experiments

We ran multiple trials with varying conditions to measure the user-perceived availability of systems with different characteristics, for users with different retry behavior. Each trial was run using 500 user threads. We first started all 500 user threads with the users initially in "thinking" state, and the server in an "up" state. We allowed the simulation to run for ten minutes before injecting a server fault of a certain duration. For the duration of the fault, the proxy stays in a "down" state and simply drops users' requests, rather than forwarding them to the web server. After the duration of the fault, the proxy goes back into an "up" state. We then continue the trial for another T_w

seconds, where T_w is five times the mean time between retries of the users.

Once users detect that the server is down, the amount of time they wait between retries is given by an exponential distribution. Thus, the probability that a user will take more than five times her mean retry time before seeing that the server has come back up is less than 0.007, allowing us to reasonably approximate that none of the users will still be in a "failed state" T_w seconds after the server comes back up.

$$A_U = 1 - \frac{\sum_{i=1}^{N} D_i}{(N)(T_R + T_F)} \qquad \begin{array}{l} A_U = \text{User-perceived availability} \\ D_i = \text{Server downtime perceived by user i} \\ N = \text{Number of users} \\ T_F = \text{Mean time to failure (MTTF)} \\ T_R = \text{Mean time to recovery (MTTR)} \end{array} \tag{4}$$

We first held the length of failure at 2400 seconds (40 minutes) while varying the mean user retry time from 100 seconds to 1200 seconds. We then held mean user retry time at 100 seconds while we varied the server failure length from 17 seconds to 5214 seconds. For each trial, we measured the number of seconds, D_i, each user perceived the server to be down. This is different from the actual amount of time the server is down, since users do not see failures right away, nor do they detect recovery from failure instantly. These measurements allow us to compute the user-perceived availability, using equation 4. For each set of parameters, we ran the experiment three times and took the average. The results of our experiments are presented in the next section.

Note that the definition of user-perceived availability as defined by equation 4 is different from that used in [3]. We define user-perceived downtime as the amount of time that users perceive the system to be down; all other times, the user is assumed to perceive the system as being up. User-perceived availability is then computed accordingly. The definition used in [3] computes user-perceived availability as the probability that a user thinks the server is down given that the user is *actively seeking service* from the server. This means that the user does not credit the server with any uptime that occurs while she is in "thinking" state.

The rationale for our definition of user-perceived availability is three-fold:

- When a user is not interacting with the server, it is reasonable for the user to assume that the server stays in same state in which the user last saw it. Hence, after a period of error-free interaction with the server, if a user leaves the server for a certain amount of time (goes into thinking state), the user can reasonably assume that the server was up for that duration. In our model, this thinking time ranges from 60 seconds to 1 hour 40 minutes.
- System administrators are most concerned with minimizing user-perceived downtime of their system. Focusing

on this facet of user-availability gives us a metric that is most directly useful to system administrators.

- Compared with the alternative, the definition given by equation 4 is easier to measure experimentally, since simulations over a single failure are sufficient for computing user-perceived availability. This is because under this definition, users' perceived uptime during long periods of failure-free operation is simply the entire failure-free period. Thus, unlike the definition used in [3], simulations of user interactions with the server during normal operation are not necessary.

IV. Results

A. Varying User Retry Rate

To study the effects of user retry rates on user-perceived availability, we vary the user retry rate while keeping the other system parameters constant. We set the server's recovery time at 2400 seconds, the server availability at 99.3% and vary the mean user retry rate from 0.00083 to 0.01000 (see table III). Note that the mean user retry rate equals the reciprocal of the mean user retry time, i.e., a mean user retry rate of 0.01 translates into a mean user retry time of 100 seconds.

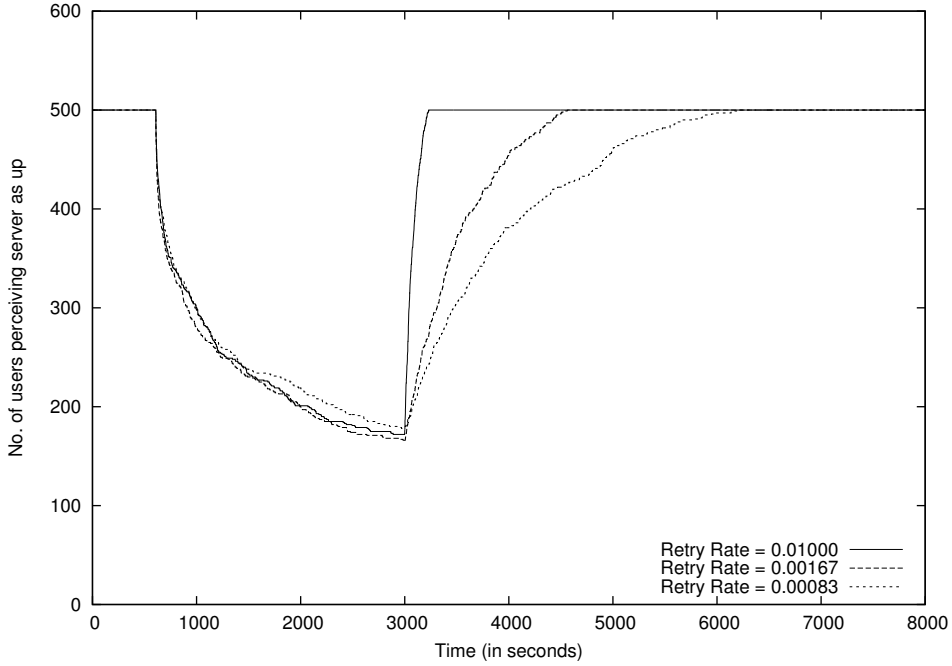
TABLE III
ACTUAL SERVER UNAVAILABILITY = 0.007, MTTR = 2400S

Retry interval (in s)	User Retry Rate (in s^{-1})	User-Perceived Unavailability
1200	0.00083	0.00546
1000	0.00100	0.00538
800	0.00125	0.00508
600	0.00167	0.00477
400	0.00250	0.00412
200	0.00500	0.00386
100	0.01000	0.00380

Figure 5 illustrates the user-side perception of the server uptime during the experiment. For illustrative purposes, the graph shows only three data sets. The x-axis is time, while the y-axis shows the number of simulated users who think that the server is up. At $t = 600s$, the proxy goes into a "down" state and begins dropping requests. This "down" state lasts for 40 minutes before the proxy comes back up. As the graph shows, users that have a higher retry rate notice the server's return to health faster than the users with a lower retry rate.

In this experiment, we vary user retry rate from $0.01s^{-1}$ to $0.00083s^{-1}$, while we measure the total amount of downtime seen by the users and compute the user-perceived availability for each user retry rate using equation 4, with the parameters $T_R = 2400s$ and $T_F \approx 340000s \approx 4$ days.

Fig. 5. User Perception vs. Time



As shown in Figure 6, users' perception of the server's downtime is worse when the user retry rate is high. Note that we plot unavailability rather than availability. This is to allow us to compare our results with Xie et al's results in [3]. User-perceived unavailability, $\overline{A}_U$, can be easily computed using the equation $\overline{A}_U = 1 - A_U$. Figure 6 also shows that there is decreasing marginal returns to increasing user retry rate. We observe that there is a "sweet spot" at retry rate = 0.0025 (mean time between retries = 400s). This suggests that beyond a threshold, there is little benefit to the user to increase her retry rate when faced with a server failure. Not surprisingly, this is consistent with the conclusions found in [3].

While this provides an interesting insight into the relationship between user behavior and user-perceived availability, that information is not useful to the administrator of a web server since she typically has no control over the users' behavior. In the next section, we examine the impact of different system parameters on user-perceived availability.

B. Varying MTTR and MTTF

In our next experiment, we keep the mean user retry time at 100 seconds and vary the server's recovery rate from 0.058824 to 0.000192 (MTTR=17s and MTTR=5214s). The server availability is kept constant at 99.3% by varying MTTF accordingly.

Figure 7 illustrates the user-side perception of the server uptime during the experiment. Again, for illustrative

Fig. 6. User-Perceived Unavailability vs. Retry Rate

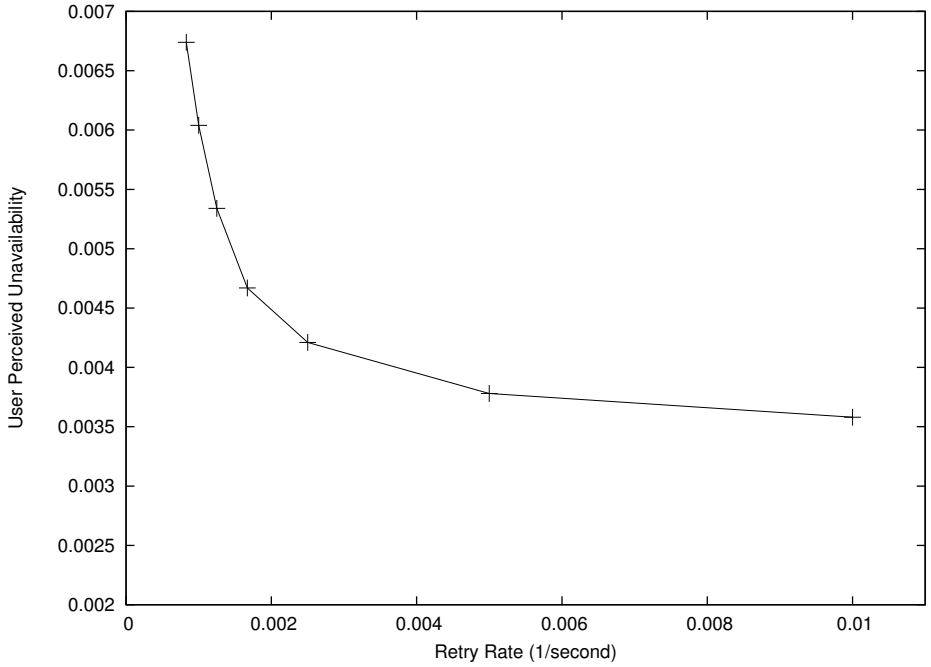


Fig. 7. User Perception vs. Time

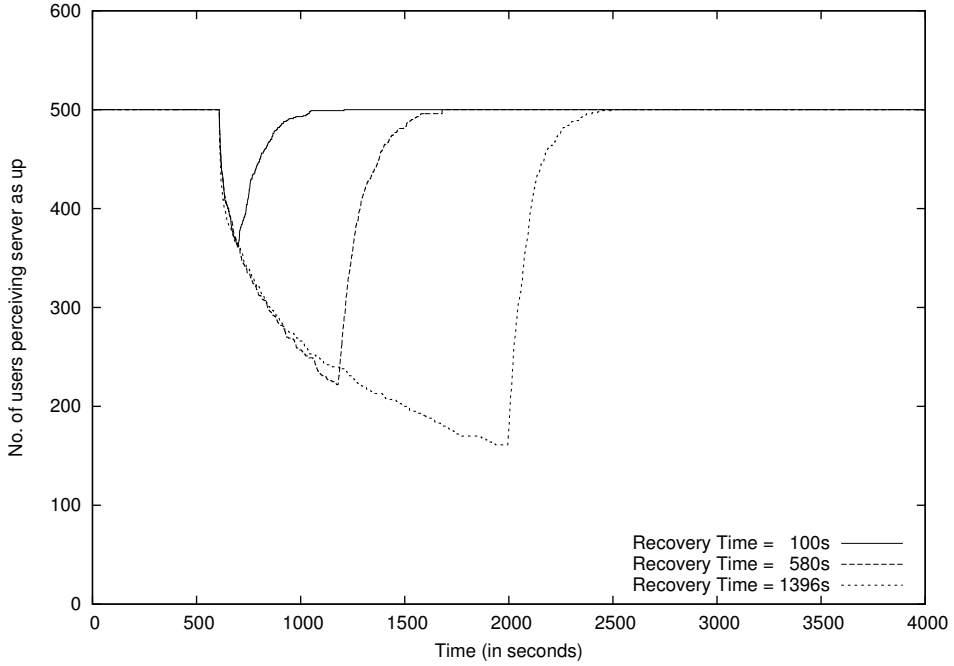
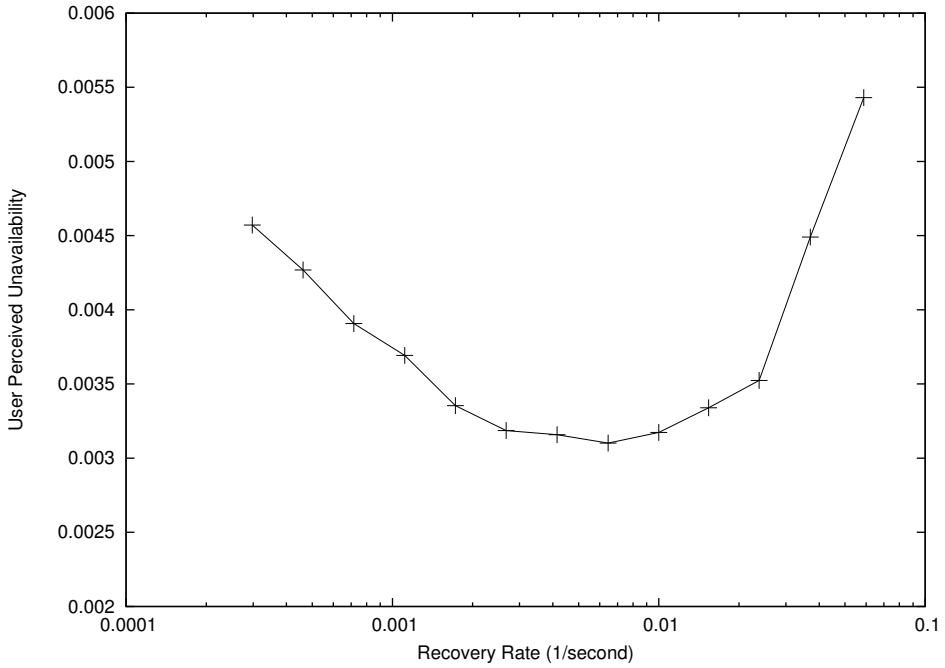


Fig. 8. User-Perceived Unavailability vs. TTR



purposes, the graph shows only 3 data sets. The x-axis is time, while the y-axis shows the number of simulated users who think that the server is up. At $t = 600$, the proxy goes into a "down" state and begins dropping requests. This "down" state lasts for 100, 580 and 3360 seconds before the proxy comes back up. As the graph shows, increasing the server recovery time increases the number of users who see a fault in the server as well as the duration that they perceive the server to be down.

From this set of experiments, we are able to calculate the user-perceived availability for various server configurations. User-perceived availability is again computed using equation 4, using the parameters $T_R = 2400s$ and $T_F \approx 340000s \approx 4$ days. This allows us to study the relationship between the recovery rate and the user-perceived availability of a system, as shown in Figure 8.

Note that the overall system availability is held constant at 99.3% for these experiments, thus, as the recovery rate increases, so does the failure rate. This implies that as we move from left to right along the x-axis, both the MTTR and the MTTF of the system drops. In other words, systems on the right side of the graph have shorter, but more frequent, failures. The actual values of user-perceived unavailabilty are also sensitive to the user retry behavior, which is set to be exponentially distributed with a mean time between retries of 100 seconds.

As the graph shows, initially, user-perceived unavailability drops as MTTR drops. However, as the system's recovery time decreases to a value close to, or lower than the mean time between user retries, we observe that the marginal returns to decreasing MTTR drops. Intuitively, when MTTR is significantly larger than the mean time

between user retry, MTTR is the dominant factor that affects user-perceived downtime, and thus, user-perceived unavailability. However, as MTTR drops and approaches the mean time between user retries, the user-perceived downtime becomes increasingly dominated by the user retry rate instead - since the system recovers fairly quickly, the amount of user-perceived downtime is mostly dependent on how quickly users notice the server's return to health, which is in turn dependent on users' retry rate. This interplay between the relative contributions of MTTR and user retry rate explains the diminishing marginal returns of decreasing MTTR.

At the same time, lowering MTTF imposes a marginal cost in terms of user-perceived availability, since more frequent failures means that more users will see the failures. Eventually, the increasing marginal cost of lowering MTTR catches up with the marginal costs of lowering MTTF. This causes the graph to flatten out and eventually start rising. As shown in Figure 8, for a system with availability of 99.3%, the graph has a minimum turning point with a user-perceived unavailability of 0.0031, and this optimal system has a recovery time between 155s and 374s.

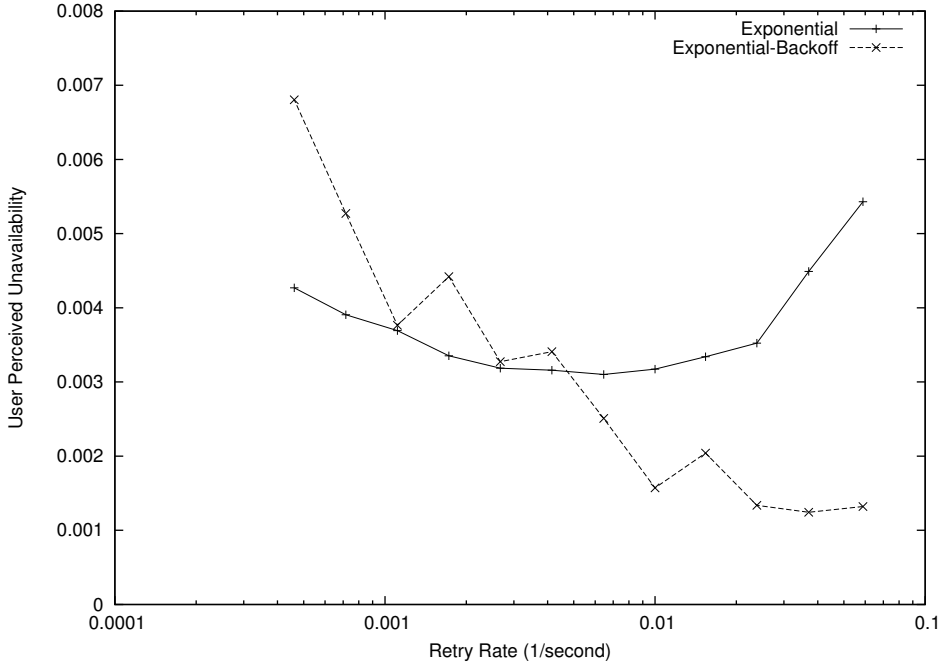
C. Exponential Backoff Model

As mentioned in section II-B, user-perceived availability is critically dependent on the user retry behavior. We have assumed that faced with a failure, user retry behavior is approximated by an exponential distribution. Our preliminary results from the examination of server logs from a commercial website show that this assumption is reasonable. However, to show the dependence of user-perceived availability on user retry behavior, we explored an alternative user retry model. Figure 9 compares the user-perceived availability to two different sets of users, holding the system availability constant while varying MTTR. One set of users retry with an exponential distribution, as in Figure 8, while the second set of users use exponential backoff - they wait 30 seconds before retrying when first faced with a failure, and double the retry interval upon each consequtive request failure. As the graph shows, the user-perceived availability differs greatly for the two sets of users.

D. Summary and Implications

Our experiments confirm one of the results found by Xie et al [3] - increasing user retry rates increases user-perceived availability. Xie et al. goes on to assert that between two systems of equal system availability, the one with lower MTTR always has better user-preceived availability. We find that this assertion holds only under certain conditions. In particular, when MTTR decreases beyond a certain point, user-perceived availability actually *suffers* from further decreases in MTTR, because the costs of decreasing MTTF eventually overtakes the diminishing returns of decreasing MTTR. This finding has an interesting implication for administrators of web servers - it implies that is not always beneficial to trade higher MTTF for lower MTTR. Thus, given limited resources, it is no longer clear

Fig. 9. Comparing Different User Retry Behavior



whether investing in improving MTTR or MTTF will yield greater user-perceivable benefits. In the next section, we propose a method for determining this.

V. Determining Optimal System Improvement

The problem of maintaining high system availability is well studied. As such, there are many options available to the system adminstrator who wants to improve either the MTTF or MTTR of her system. To increase the MTTF of a system, the system administrator needs to reduce the frequency of failures. One direct way of doing this is to make use of high quality components which have high MTTF. Another approach is to make use of fault-tolerant components such as ECC memory, or RAID storage arrays. Finally, when it is known that failures are occurring due to software faults, the only solution may be to change the software being used, or to locate the bug in the software, both of which may be costly and time-consuming. Another approach to increasing system availability is to decrease the MTTR of the system. When replication and overprovisioning are possible, the system administrator can add fast-failover machines to the system so that each machine is more lightly loaded and can take over a peer's load in case of a failover. The system administrator can also choose to use databases that have fast restart and recovery times, so that when a problem is encountered, the system can be very quickly restarted. Recent work by Candea et al. has also proposed recursive restartability [10] and microreboots [11] as new ways of reducing MTTR.

Where and how to invest limited resources to improving system availability is often a difficult question to answer.

Not only is there a multitude of options available, the effect of each is often difficult to determine before actual deployment. In this section, we make a contribution to tackling this problem by proposing a method for determining, for a given system, whether investing in an improvement in MTTR or MTTF will yield greater user-visible benefits.

For any given system we want to be able to decide whether it is more beneficial to improve MTTR or MTTF. To illustrate this, consider the following examples. Assume that we have two systems, A and B, with the parameters given in table IV.

TABLE IV
SYSTEM CONFIGURATIONS

	$\overline{A}_S$	$\overline{A}_U$	MTTR	MTTF
System A	0.00700	0.00390	65s	9220s (2.6h)
System B	0.00700	0.00310	1400s (23.3min)	19800s (2.3days)

Based on the simulation data that we gathered, we are able to plot Figures 10 and 11, which give an interesting insight to the problem.

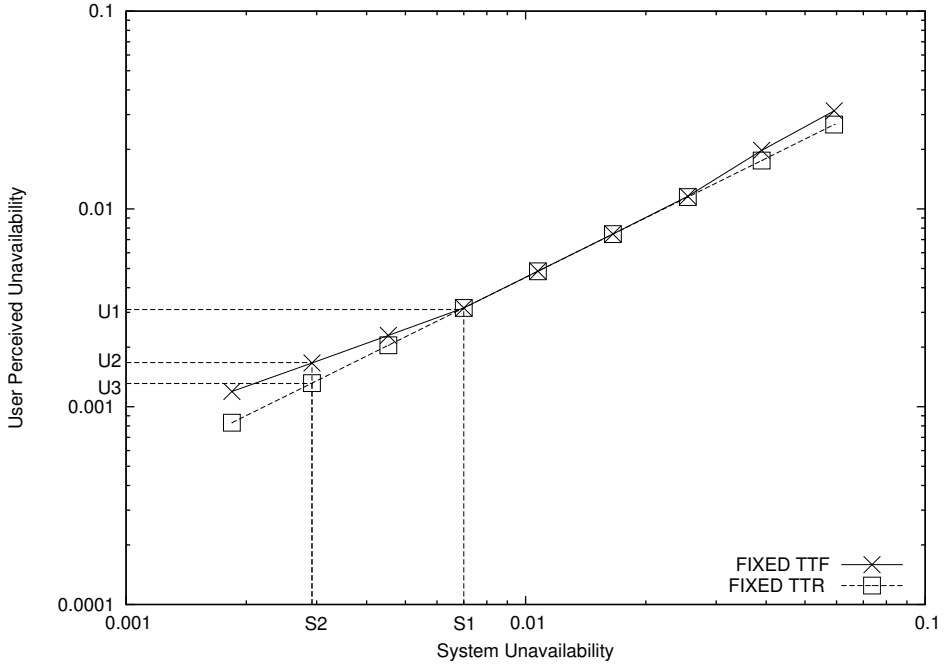
In the graphs, the x-axis represents the system unavailability, and the y-axis represents the corresponding user-perceived unavailability. The two lines in each graph represents the trajectories along which the system and user-perceived unavailability values will move according to the action taken. For instance, the "FIXED MTTF" line represents the path along which system unavailability, $\overline{A}_S$, and user-perceived unavailability, $\overline{A}_U$, will vary if the system administrator keeps her system's MTTF constant and lowers its MTTR.

First, consider system A. From Figure 10, we observe that by lowering the system's MTTR, the system administrator can move the system along the "FIXED MTTF" line, lowering $\overline{A}_S$ from S1 to S2 and $\overline{A}_U$ from U1 to U2. Alternatively, the system administrator can increase the system's MTTF and move the system along the "FIXED MTTR" line, achieving the same gain in lowering $\overline{A}_S$ from S1 to S2, but with a larger gain in $\overline{A}_U$ from U1 to U3, where $U3 - U1 > U2 - U1$. Although both improvements result in the same system unavailablity, the improvement in user-perceived unavailabilty is 7% greater if the system unavailablity improvement was due to an improvement in MTTF, rather than MTTR. In this case, the system administrator should invest resources in improving the system's MTTF.

TABLE V
SYSTEM A IMPROVEMENTS

Step taken	New $\overline{A}_S$	New $\overline{A}_U$	$\Delta\overline{A}_U$
Increase MTTF to 22200s (6.2h)	0.00292	0.00145	.00245
Decrease MTTR to 27s	0.00292	0.00163	.00227

Fig. 10. User-Perceived Unavailability vs. System Unavailability for System A



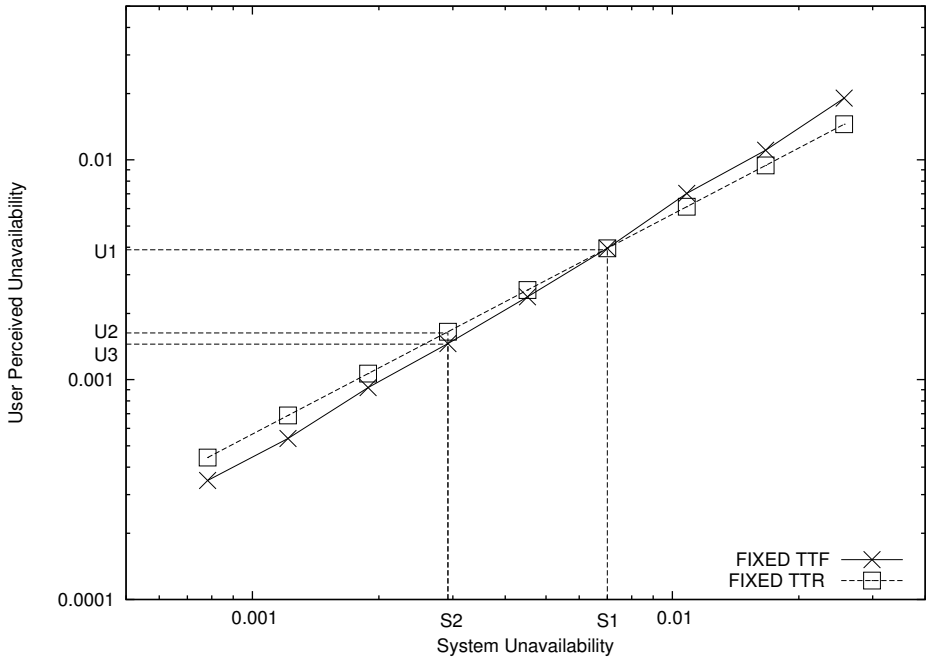
However, increasing the system's MTTF is not always better than decreasing its MTTR. To appreciate this fact, consider System B, which has the same system unavailability as System A, but different MTTR and MTTF. From Figure 11, we observe that by lowering the system's MTTR, the system moves along the "FIXED MTTF" line, lowering $\overline{A}_S$ from S1 to S2 and $\overline{A}_U$ from U1 to U3. If the adminstrator increases the system's MTTF, however, the system moves along the "FIXED MTTR" line, again lowering $\overline{A}_S$ from S1 to S2 but this time with a smaller change in $\overline{A}_U$, from U1 to U2. The effects of these two possible improvements to the system are tabulated in table VI. Although both improvements cause an equal improvement in system availability, improving MTTR increases user-perceived availability 25% more than improving MTTF. In this case, approaches to improve the system that focus on MTTR will be more effective.

TABLE VI
SYSTEM B IMPROVEMENTS

Step taken	New $\overline{A}_S$	New $\overline{A}_S$	$\Delta\overline{A}_U$
Increase MTTF to 477000s (5.5days)	0.00292	0.00167	.00143
Decrease MTTR to 580s	0.00292	0.00131	.00179

Thus, for any given system, once we have determined the effect of changing MTTR/MTTF on user-perceived unavailability as well as system unavailability, and plot them in a graph similar to Figures 10 and 11, we can better

Fig. 11. User-Perceived Unavailability vs. System Unavailability for System B



decide where to allocate resources to improve the user-visible availability of the system.

VI. Related Work

The user model used in this paper was first proposed by Deng [8], while the server-user interaction model was first used by Xie et al. in [3], where they proposed the use of Markov regenerative process models to study user-perceived availability. Feldmann studied server-side TCP connection arrival distributions and found that they are best modeled by distributions with heavy tails, particularly Weibull distributions [7]. Barford and Crovell proposed representative web workloads for network and server evaluation based on user models developed from empirical data [4]. Their model share Deng's concepts of ON-OFF user states [8], as well as the use of Pareto and Weibull distributions. However, the parameters used in the various distributions differed. Mah proposed an empirically-derived model of HTTP network traffic [12]. As part of his study of packet-level traces, Mah found that the average think time of users ranged from 800 seconds to 2000 seconds, while the distribution of think time resembled a heavy tailed distribution, such as the Pareto distribution used in our study. Choi and Limb presents a web traffic model for the evaluation of communication networks [5]. They found that the ON-time of web users followed a Weibull distribution with parameters $k = 0.77$ and $\theta = 0.68$, which differs significantly from our parameters of $k = 0.88$ and $\theta = e^{4.5}$, as proposed by Deng.

VII. Future Work

We hypothesize that there are several categories of user behavior when users are faced with a system failure, and that the categories depend on the nature of the site in question. For example, a user trying to perform a time-critical banking transaction may repeatedly retry the transaction at regular intervals until it succeeds, whereas a user causally shopping may exhibit retry behavior that is most accurately approximated by an exponential backoff model. Our future work will focus on develping better end-user models characterized by different distributions, and understanding which kinds of sites encourage the behaviors captured in each model. This will allow us to fine-tune our recommendations to site-operators as to how best to spend their resources to increase user-perceived site availability.

VIII. Conclusion

We verified previous work and showed that higher user retry rate gives higher user-perceived availability, with decreasing marginal returns. We show that MTTR is more important than MTTF in some, but not all situations, for providing high user-perceived availability. For any given system, we proposed a method to determine which of MTTR or MTTF to improve, in order to obtain the maximum amount of user-visible benefits.

IX. Acknowledgements

We would like to thank operators of that anonymous website for giving us access to their server logs, and members of the Berkeley-Stanford ROC group for reading our draft and providing invaluable feedback. Finally, we would also like to thank Wei Xie for his insight with regards to the differences in our definitions of user-perceived availability.

References

[1] A. Fox, "Toward recovery-oriented computing," 2002. [Online]. Available: citeseer.nj.nec.com/fox02toward.html

[2] "Cost of downtime study," 1996. [Online]. Available: http://www.contingencyplanningresearch.com/cod.htm

[3] W. Xie, H. Sun, Y. Cao, and K. S. Trivedi, "Modeling of user perceived webserver availability," in *Proceedings of the IEEE International Conference on Communications, May 2003*, 2003.

[4] P. Barford and M. Crovella, "Generating representative web workloads for network and server performance evaluation," in *Measurement and Modeling of Computer Systems*, 1998, pp. 151–160. [Online]. Available: citeseer.nj.nec.com/barford98generating.html

[5] H.-K. Choi and J. O. Limb, "A behavioral model of web traffic," in *Proc. of International Conference of Networking Protocol*, 1999.

[6] V. Paxson and S. Floyd, "Why we don't know how to simulate the internet," in *Winter Simulation Conference*, 1997, pp. 1037–1044. [Online]. Available: citeseer.nj.nec.com/paxson97why.html

[7] A. Feldmann, "Characteristics of tcp connection arrivals," 1998. [Online]. Available: citeseer.nj.nec.com/feldmann98characteristics.html

[8] S. Deng, "Empirical model of www document arrivals at access link," in *Proceedings of the IEEE International Conference on Communication, June 1996.*, 1996. [Online]. Available: citeseer.nj.nec.com/deng96empirical.html

[9] M. Crovella and A. Bestavros, “Self-similarity in world wide web traffic: Evidence and possible causes,” in *Proceedings of SIGMETRICS’96: The ACM International Conference on Measurement and Modeling of Computer Systems.*, Philadelphia, Pennsylvania, May 1996. [Online]. Available: http://www.cs.bu.edu/fac/best/res/papers/sigmetrics96.ps

[10] G. Candea and A. Fox, “Recursive restartability: Turning the reboot sledgehammer into a scalpel,” in *International Conference on Dependable Systems and Networks (DSN-2002)*, 2001.

[11] G. Candea, J. Cutler, and A. Fox”, “Improving availability with recursive microreboots: A soft-state system case study,” *Performance Evaluation Journal*, vol. 56, no. 1-3, 2004.

[12] B. A. Mah, “An empirical model of http network traffic,” in *Proceedings of Infocomm*, 1997.