

Failure Data Analysis of a LAN of Windows NT Based Computers

M. Kalyanakrishnam, Z. Kalbarczyk, R. Iyer

Center for Reliable and High-Performance Computing
University of Illinois at Urbana-Champaign
1308 W. Main St., Urbana, IL 61801
E-mail: [mahesh, kalbar, iyer]@crhc.uiuc.edu

Abstract. *This paper presents results of a failure data analysis of a LAN of Windows NT machines. Data for the study was obtained from event logs collected over a six-month period from the mail routing network of a commercial organization. The study focuses on characterizing causes of machine reboots. The key observations from this study are: (1) most of the problems that lead to reboots are software related, (2) rebooting the machine does not always solve the problem (in about 60% of the reboots, the rebooted machine reported problems within an hour or two of the reboot), (3) there are indications of propagated or correlated failures, and (4) though the average availability evaluates to over 99%, the machine downtime lasts (on average) two hours. Since the machines are dedicated mail servers, bringing down one or more of them can potentially disrupt storage, forwarding, reception and delivery of mail. This suggests that the average availability is not a good measure to characterize this type of network service.*

1 Introduction

Understanding the nature of failures in standalone and networked computer systems is essential to improving their availability and reliability. In most commercial systems, information about failures can be obtained from the manual logs maintained by administrators or from the automated event-logging mechanisms in the underlying operating system. Manual logs are very subjective and often unavailable. Hence they are not typically suited for automated analysis of failures. In contrast, the event logs maintained by the system have predefined formats, provide contextual information in case of failures (e.g., a trace of significant events that precede a failure), and are thus conducive to automated analysis. Moreover, as failures are relatively rare events, it is necessary to meticulously collect and analyze error data for many machine-months for the results of the data analysis to be statistically valid. Such regular and prolonged data acquisition is possible only through automated event logging. Hence most studies of failures in single and networked computer systems are based on the error logs maintained by the operating system running on those machines.

This paper presents methodology and results from an analysis of failures prominently found in a network of about 70 Windows NT based mail servers (running Microsoft Exchange software). The data for the study is obtained from event logs (i.e., logs of machine events that are maintained and modified by the Windows NT operating system) col-

lected over a six-month period from the mail routing network of a commercial organization. In this study we analyze only machine reboots because they constitute a significant portion of all logged failure data and are the most severe type of failure. As a starting point, a preliminary data analysis is conducted to classify the nature of observed failure events. This failure categorization is then used to examine the behavior of individual machines in detail and to derive a finite state model. The model depicts the behavior of a typical machine. Finally, a domain-wide analysis is performed to capture the behavior of the domain in a finite state model and to examine error propagation.

The investigation of failure data provides useful insights into the nature of observed failure behavior and improves understanding of the dynamics of typical problems in network systems. Some of our key findings are given below:

1. Most problems that lead to reboots are software related (only 10% are attributable to specific hardware components).
2. Connectivity problems cause most reboots.¹ A significant percentage of these problems are persistent.
3. Rebooting the machine does not appear to solve the problem in many cases. About 60% of the time, rebooted machines report problems within an hour or two.
4. Though the average availability evaluates to over 99%, a typical machine in the domain provides acceptable service only about 92% of the time, on average.
5. There are indications of propagated or correlated failures. Typically in such cases, multiple machines exhibit identical or similar problems at almost the same time.

2 Related Work

Analysis of failures in computer systems has been the focus of active research for quite some time. This section summarizes such analyses conducted on standalone and networked computer systems. Studies of failures occurring in commercial systems (e.g., VAX/VMS, Tandem/GUARDIAN) are based primarily on failure data collected from the field. The focus of such studies is on categorizing the nature of failures in the systems (e.g., software failures, hardware fail-

¹ Connectivity problems indicate that either a system component (e.g., a server) or a critical application (e.g., MS Exchange System Attendant) could not retrieve information from a remote machine.

ures), identifying availability bottlenecks, and obtaining models to estimate the availability of the systems being analyzed. Lee [8], [9] analyzed failures in Tandem's GUARDIAN operating system. Lee [8] examined processor halts and identified memory management software and interrupt handlers as the major causes for processor halts. Lee [9] examined software dependability of the GUARDIAN operating system.

Tang [16] analyzed error logs pertaining to a multicomputer environment based on VAX/VMS cluster. Thakur [18] presented an analysis of failures in the Tandem Nonstop-UX operating system. The study analyzed problems that resulted in panics and crashed the system. Thakur [19] described a simple yet effective methodology for collecting and analyzing failures in a network of UNIX-based workstations.

Hsueh [4] explored errors and recovery in IBM's MVS operating system. Based on the error logs collected from MVS systems, a semi-Markov model of multiple errors (i.e. errors that manifest themselves in multiple ways) was constructed to analyze system failure behavior. Measurement-based software reliability models were also presented in [8], [9] (for the GUARDIAN system) and [16], [17] (for the VAX cluster).

The impact of workload on system failures was also extensively studied. Castillo [1] developed a software reliability prediction model that took into account the workload imposed on the system. Iyer [5] examined the effect of workload on the reliability of the IBM 3081 operating system. Mourad [1] performed a reliability study on the IBM MVS/XA operating system and found that the error distribution is heavily dependent on the type of system utilization. Meyer [11] presented an analysis of the influence of workload on the dependability of computer systems.

Maxion [12] presented useful and interesting results on anomalies and their detection. Lin [10] and Tsao [20] focused on trend analysis in error logs. Gray [3] presented results from a census of Tandem systems. Chillarege [2] presented a study of the impact of failures on customers and the fault lifetimes. Sullivan [14], [15] examined software defects occurring in operating systems and databases (based on field data). Velardi [21] examined failures and recovery in the MVS operating system. An in-depth overview of experimental and analytical techniques for analysis of computer systems dependability can be found in [6].

3 Error Logging in Windows NT

Windows NT operating system offers capabilities for error logging. This software records information on errors occurring in the various subsystems, such as memory, disk, and network subsystems, as well as other system events, such as reboots and shutdowns. The reports usually include information on the location, time, type of the error, the system state at the time of the error, and sometimes error recovery (e.g., retry) information. The main advantage of on-line

automatic logging is its ability to record a large amount of information about transient errors and to provide details of automatic error recovery processes, which cannot be done manually. Disadvantages are that an on-line log does not usually include information about the cause and propagation of the error or about off-line diagnosis. Also, under some crash scenarios, the system may fail too quickly for any error messages to be recorded.

Event logging in the Windows NT operating system is provided through a dedicated *Event Logging Subsystem*. This subsystem consists of multiple execution threads each carrying out part of the event logging functionality. Some threads are dedicated to receiving event log requests through transport layer *ports*, whereas other threads actually write the event to the event log.

The mechanism of event logging is slightly different for different types of events. For example, for events that are logged by applications or other subsystems, the Event Logging Subsystem provides an API (Application programmer interface) to log events. For events that are generated within the *Executive*² (e.g., device drivers), the events are directly written to the event log file by the *I/O Manager*, which is a part of the Windows NT Executive.

An important question to be asked here is: How accurate are event logs in characterizing failure behavior of the system? While event logs provide valuable insight into understanding the nature and dynamics of typical problems observed in a network system, in many cases the information in event logs is not sufficient to precisely determine a nature of a problem (e.g., whether it was a software or hardware component failure). The only reliable way to improve accuracy of logs is (1) to perform more frequent, detailed logging by each component and (2) instrument the Windows NT code with new (more precise) logging mechanisms. However, there is always a trade-off between accuracy and intrusiveness of measurements. No commercial organization will permit someone to install an untested tool to monitor the network. Consequently, we use existing logs not only to characterize failure behavior of the network (presented in this paper), but also to determine how the logging system could be improved (e.g., by adding to the operating system a query mechanism to remotely probe system components about their status). It should be noted that in many commercial operating systems (e.g., MVS) event logs are accurate enough to document failures.

3.1 Event log format

The event logs conform to a specific format employed by the Windows NT operating system. Events on Windows NT machines fall into one of three types.

- *Application events* are those that are logged by applications running on NT machines (e.g., error logged by MSeXchange MTA (Message Transfer Agent)).

² *Kernel* and *Executive* are two basic components of the Windows NT operating system. The two components run in *kernel* (privileged) *mode*.

- *System events* are those that are reported by components of the Windows NT operating system (e.g., server, redirector, NETLOGON service, etc.).
- *Security events* are those that relate to user authentication or verification of access permissions.

A sample event log is shown below, the format of the log may not be universal, but the fields in it are.

```
1997/09/11 15:02:53 4 0 5715 NETLOGON N/A
EXCHOU-CA0201 LSA \EXCHOU-CONN02 1
```

The fields in the event log are:

- *Date and time* of the event
- *Severity* of the event (4) (1 indicates an Error, 2 indicates a Warning, 4 indicates an Information message)
- *Event id* (5715)
- The *source* that logs that event (NETLOGON)
- The *machine* on which the event was logged (EXCHOU-CA0201)
- *Event-specific information* (EXCHOU-CONN02 1)

Typically, each source that logs an event provides a "message" file that contains a description of the event. All Windows NT systems have a built-in Event Viewer, which uses the message files to provide a textual description of the event. This particular method of displaying events poses quite a few problems: (1) the message files are shipped with applications. In order to interpret the event, the application that logged the message must be running on the machine performing the analysis. Because this is impractical, it is not possible to obtain the textual description of most application events, and (2) even for system events, not all message files are accessible. The Windows NT event logging source code contains message files for some, but not all, system components.

4 Data Processing

The first step in data processing is error classification based on the subsystems and components in which they occur. There is no uniform or best error classification, because different systems have different hardware and software architectures. Moreover, the error categories depend on the criteria used to classify errors.

4.1 Classification of Data Collected from a LAN of Windows NT-based Servers

The initial breakup of the data on a system reboot is primarily based on the events that preceded the current reboot by no more than an hour (and that occurred after the previous reboot). For each instance of a reboot, the most severe and frequently occurring events (hereafter referred to as *prominent events*) are identified. The corresponding reboot is then categorized based on the source and the id of these prominent events. In some cases, the prominent events are specific enough to identify the problem that caused the reboot. In other cases, only a high-level description of the problem can be obtained based on the knowledge of the

prominent events. Table 1 shows the breakup of the reboots by category.

Hardware or firmware related problems: This category includes events that indicate a problem with hardware components (network adapter, disk, etc.), their associated drivers (typically drivers failing to load because of a problem with the device), or some firmware (e.g., some events indicated that the Power On Self Test had failed).

Connectivity problems: This category denotes events that indicated that either a system component (e.g., redirector, server) or a critical application (e.g., MS Exchange System Attendant) could not retrieve information from a remote machine. In these scenarios, it is not possible to pinpoint the actual cause of the connectivity problem. Some of the connectivity failures result from network adapter problems and hence are categorized as hardware related.

Category	Frequency	Percentage
Total reboots	1100	100
Hardware or firmware problems	105	9
Connectivity problems	241	22
Crucial application failures	152	14
Problems with a software component	42	4
Normal shutdowns	63	6
Normal reboots/power-off (no indication of any problems)	178	16
Unknown	319	29

Table 1: Breakup of Reboots

Crucial application failure: This category encompasses reboots, which are preceded by severe problems with, and possibly shutdown of, critical application software (such as Message Transfer Agent). In such cases, it wasn't clear why the application reported problems. If an application shutdown occurs as a result of connectivity problem, then the corresponding reboot is categorized as connectivity-related.

Problems with a software component: Typically these reboots are characterized by startup problems (such as a critical system component not loading or a driver entry point not being found). Another significant type of problem in this category is the machine running out of virtual memory, possibly due to a memory leak in a software component. In many of these cases, the component causing the problem is not identifiable.

Normal shutdowns: This category covers reboots, which are not preceded by warnings or error messages. Additionally, there are events that indicate shutting down of critical application software and some system components (e.g., the BROWSER). These represent shutdowns for maintenance or for correcting problems not captured in the event logs.

Normal reboots/power-off: This category covers reboots which are typically not preceded by shutdown events, but do not appear to be caused by any problems either. No warnings or error messages appear in the event log before the reboot.

Based on the table, the following observations can be made about the failures:

1. 29% of the reboots cannot be categorized. Such reboots are indeed preceded by events of severity 2 or lesser, but there is not enough information available to decide (a) whether the events were severe enough to force a reboot of the machine or (b) the nature of the problem that the events reflect.
2. A significant percentage (22%) of the reboots have reported *connectivity problems*. Connectivity problems suggest that there could be propagated failures in the domain. However, it is not possible to say with certainty whether a given connectivity problem is due to the local machine or is a case of a propagated failure. Either the event logs do not have such information, or the information, though available, is not easy to interpret. In the absence of clear-cut evidence, one way to probe deeper into the issue of the propagation of failures is to use contextual information from the domain. Specifically, by examining the events logged by the machines between which a connectivity problem exists we could say, in a probabilistic sense, which of the two machines was faulty.
3. Only a small percentage (10%) of the reboots can be traced to a system hardware component. Most of the identifiable problems are software related.
4. Nearly 50% of the reboots are *abnormal* reboots (i.e., the reboots were due to a problem with the machine rather than due to a normal shutdown). Many of the reboots could not be categorized satisfactorily because Windows NT, at present, does not log an event that indicates the shutting down of the machine. A shutdown event in Windows NT would greatly reduce the uncertainty determining whether a given reboot is normal.
5. In nearly 15% of the cases, server problems with a crucial mail server application force a reboot of the machine.

Each of the categories identified above can be analyzed further to provide more detailed insight into reasons for observed system problems. Due to space limitations we do not provide detailed breakups for failure categories identified in Table 1 (the detailed breakups can be found in [7]).

5 Analysis of Failure Behavior of Individual Machines

After the preliminary investigation of the causes of failures, we probe failures from the perspective of an individual machine as well as the whole network. First we focus on the failure behavior of individual machines in the domain to obtain (1) estimates of machine uptimes and downtimes³, (2) an estimate of the availability of each machine, and (3) a

³ Definitions of uptime and downtime are as accurate as they can be using the available event logs. To improve the accuracy in determining these parameters we would need (1) a heartbeat mechanism for periodic polling machines in the network or (2) improved event-logging e.g., by adding a shutdown event.

finite state model to describe the failure behavior of a typical machine in the domain.

Machine uptimes and downtimes are estimated as follows:

- For every reboot event encountered, the timestamp of the reboot is recorded.
- The timestamp of the event immediately preceding the reboot is also recorded. (This would be the last event logged by the machine before it goes down.)
- A smoothing factor of one hour is applied to the reboots (i.e., for multiple reboots that occurred within an period of one hour, except the last one, are disregarded). (Since the intermediate reboots indicate an incomplete recovery from the failure, the machine would have to be considered as being down until the last of such reboots occurs.)
- Each uptime estimate is generated by calculating the time difference between a reboot timestamp and the timestamp of the event preceding the next reboot.
- Each downtime estimate is obtained by calculating the time difference between a reboot timestamp and the timestamp of the event preceding it.

Machine *uptimes* and machine *downtimes* are presented in Table 2. As the standard deviation suggests, there is a great degree of variation in the machine uptimes. The longest uptime was nearly three months. The average is skewed because of some of the longer uptimes. The median is more representative of the typical uptime.

Item	Machine Uptime Statistics	Machine Downtime Statistics
Number of entries	616	682
Maximum	85.2 days	15.76 days
Minimum	1 hour	1 second
Average	11.82 days	1.97 hours
Median	5.54 days	11.43 minutes
Standard Deviation	15.656 days	15.86 hours

Table 2: Machine Uptime & Downtime Statistics

As the table shows, 50% of the downtimes last about 12 minutes. This is probably too short a period to replace hardware components and reconfigure the machine. The implication is that majority of the problems are software related (memory leaks, misloaded drivers, application errors etc.). The maximum value is unrealistic and might have been due to the machine being temporarily taken off-line and put back in after a fortnight.

Since the machines under consideration are dedicated mail servers, bringing down one or more of them would potentially disrupt storage, forwarding, reception, and delivery of mail. The disruption can be prevented if explicit rerouting is performed to avoid the machines that are down. But it is not clear if such rerouting was done or can be done. In this context the following observations would be causes for concern: (1) average downtime measured was nearly 2 hours or (2) 50% of the measured uptime samples were about 5 days or less.

5.1 Availability

Having estimated machine uptime and downtime, we can estimate the availability of each machine. The availability is evaluated as the ratio:

$$[\langle \text{average uptime} \rangle / (\langle \text{average uptime} \rangle + \langle \text{average downtime} \rangle)] * 100$$

Table 3 summarizes the availability measurements. As the table depicts, the majority of the machines have an availability of 99.7% or higher. Also there is not a large variation among the individual values. This is surprising considering the rather large degree of variation in the average uptimes. It follows that machines with smaller average uptimes also had correspondingly smaller average downtimes, so that the ratios are not very different. Hence, the domain has two types of machines: those that reboot often but recover quickly and those that stay up relatively longer but take longer to recover from a failure.

Item	Value
Number of machines	66
Maximum	99.99
Minimum	89.39
Median	99.76
Average	99.35
Standard Deviation	1.52

Table 3: Machine Availability

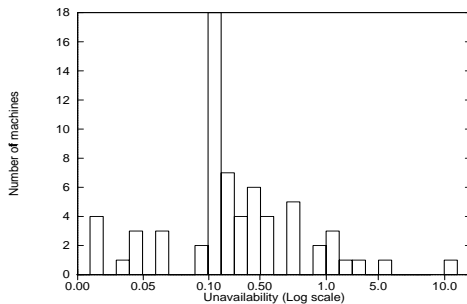


Figure 1: Unavailability Distribution

Figure 1 shows the *unavailability distribution* across the machines (unavailability was evaluated as: $100 - \text{Availability}$). The horizontal axis (in Figure 1) represents unavailability expressed in percentage, (e.g., *unavailability 0.1* means that the system *availability is 99.9%*). Less than 20% of the machines had an availability of 99.9% or higher. However, nearly 90% of the machines had an availability of 99% or higher. It should be noted that these numbers indicate the fraction of time the machine is alive. They do not necessarily indicate the ability of the machine to provide useful service because the machine could be alive but still unable to provide the service expected of it. Each of the machines in the domain acts as a mail server. Hence, if any of these mail servers has problems that prevent it from receiving, storing, forwarding, or delivering mail, then that server would effectively be unavailable to the user machines even though it is up and running. To obtain a better estimate of machine availability, it is necessary to examine how long the machine is actually able to provide service to user machines.

5.2 Modeling Machine Behavior

To obtain more accurate estimates of machine availability, we modeled the behavior of a typical machine in terms of a *state transition diagram*. The model was based on the events that each machine logs. In the model, each state represents a level of functionality of the machine. A machine is either in a fully functional state, in which it logs events that indicate normal activity, or in a partially functional state, in which it logs events that indicate problems of a specific nature.

Selection and assignment of states to a machine was performed as follows. The logs were split into time-windows of one hour each. For each such window, the machine was assigned a state, which it occupied throughout the duration of the window. The assignment was based on the events that the machine logged in the window. Table 4 describes the states identified for the model. Each machine (except PDC whose transitions were different from the rest) in the domain was modeled in terms of the states mentioned in the table. A hypothetical machine (described in Figure 2) was created by combining the transitions of all individual machines and filtering out transitions that occurred less frequently. In Figure 2, the weight on each outgoing edge represents the fraction of all transitions from the originating state (tail of the arrow) that end up in a given terminating state (head of the arrow). For example, if there is an edge from state A to state B with a weight of 0.5, then it would indicate that 50% of all transitions from state A are to state B. From Figure 2, the following observations can be made.

- Only about 40% of the transitions out of the *Reboot* states are to the *Functional* state. This indicates that in the majority of the cases, either the reboot is not able to solve the original problem, or it creates new ones.
- More than 50% of the transitions out of *Disk problems* state are to the *Functional* state. Also, we do not observe any significant transitions from the *Disk problems* state to other states. This could be due to one or more of the following (1) the machines are equipped with redundant disks so that even if one of them is down, the functionality is not disrupted in a major way, (2) the disk problems, though persistent, are not severe enough to disrupt normal activity (maybe retries to access the disk succeed), and (3) the activities that are considered to be representative of the *Functional* state may not involve much disk activity.
- More than half of the transitions out of the *Startup problems* are to the *Connectivity problems* state. Thus, the majority of the startup problems are related to components that participate in network activity.
- Most of the problems that appear when the machine is functional are related to network activity. Problems with the disk and other components are less frequent.
- The self loops on states *Connectivity problems*, *Disk problems*, *Browser problems*, and *Server problems* indicate that these problems might be more persistent than others.

State Name	Main Events (id/source/severity)	Explanation
Reboot	6005/EventLog/4	Machine logs reboot and other initialization events
Functional	5715/NETLOGON/4 1016/MSEExchangeIS Private/8	Machine logs successful communication with PDC ⁴
Connectivity problems	3096/NETLOGON/1 5719/NETLOGON/1	Problems locating the PDC
Startup problems	7000/Service Control Manager/1 7001/Service Control Manager/1	Some system component or application failed to startup
MTA problems	2206/MSEExchangeMTA/2 2207/MSEExchangeMTA/2	Message Transfer Agent has problems with some internal databases
Adapter problems	4105/CpqNF3/1 4106/CpqNF3/1	The NetFlex Adapter driver reports problems
Temporary MTA problems	9322/MSEExchangeMTA/4 9277/MSEExchangeMTA/2 3175/MSEExchangeMTA/2 1209/MSEExchangeMTA/2	Message Transfer Agent reports problems of a temporary (or less severe) nature
Server problems	2006/Srv/1	Server component reports having received badly formatted requests
BROWSER problems	8021/BROWSER/2 8032/BROWSER/1	Browser reports inability to contact the master browser
Disk problems	11/Cpq32fs2/1 5/Cpq32fs2/1 9/Cpqarray/1 11/Cpqarray/1	Disk drivers report problems
Tape problems	15/dlftape/1	Tape driver reports problems
Snmpelea problems	3006/Snmpelea/1	Snmp event log agent reports error while reading an event log record
Shutdown	8033/BROWSER/4 1003/MSEExchangeSA/4	Application/machine shutdown in progress

Table 4: Machine States

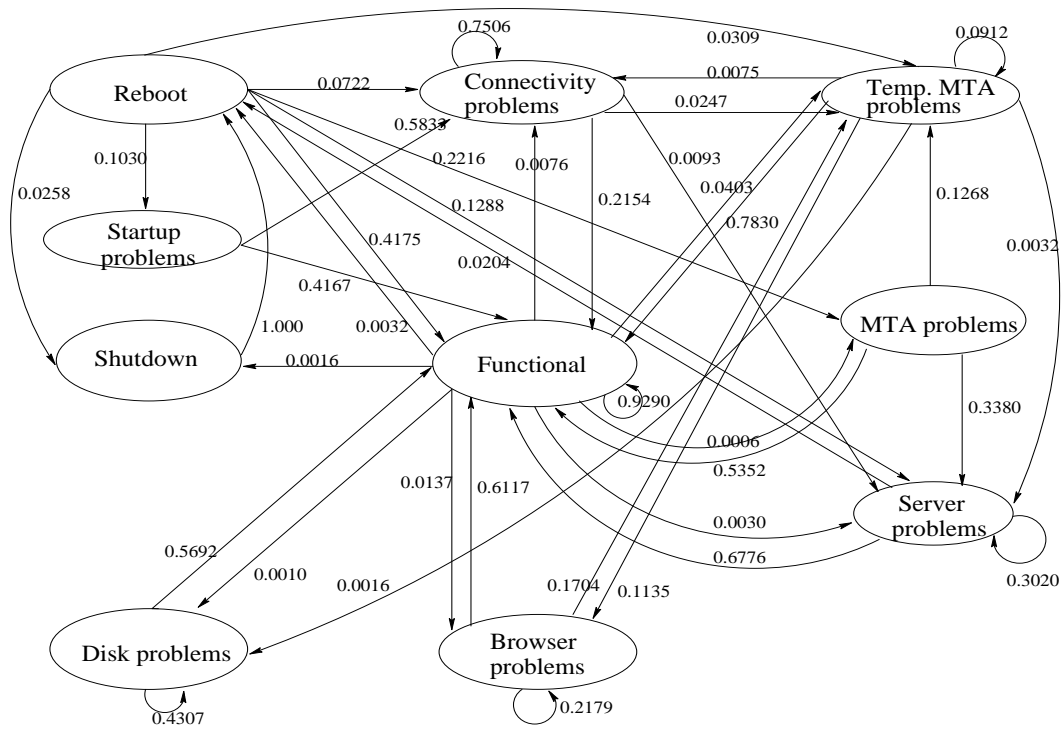


Figure 2: State Transitions of a Typical Machine

⁴ In the analyzed network, the machines belonged to a common Windows NT domain. One of the machines was configured as the Primary Domain Controller (PDC). The rest of the machines functioned as Backup Domain Controllers (BDCs).

- Over 11% of the transitions out of the *Temporary MTA problems* state are to the *Browser problems* state. We suspect that there was a local problem that caused RPCs to timeout or fail and caused problems for the MTA and BROWSER. Another possibility is that, in both cases, it was the same remote machine that could not be contacted. Based on the available data, it was not possible to determine the real cause of the problem.

To view the transitions from a different perspective, we computed the weight of each outgoing edge as a fraction of all the transitions in the finite state machine. Such a computation provides interesting insights:

1. Nearly 10% of all the transitions are between the *Functional* and *Temporary MTA* problems states. These MTA problems are typically problems with some RPC calls (either failing or being canceled).
2. About 0.5% (1 in 200) of all transitions are to the *Reboot* state.
3. The majority of the transitions into the *MTA problems* state are from the *Reboot* state. Thus, MTA problems are primarily problems that occur at startup. In contrast, the majority of the transitions into the *Server problems* state and the *Browser problems* state (excluding the self loops) are from the *Functional* state. So, these problems (or at least a significant fraction of them) typically appear after the machine is functional.
4. 92.16% of all transitions are into the *Functional* state. This figure is approximately a measure of the average time the hypothetical machine spends in the functional state. Hence it is a measure of the average availability of a typical machine. In this case, availability measures the ability of the machine to provide service, not just to stay alive.

It can be observed (from the analysis above) that Message Transfer Agent experience a lot of problems and is particularly vulnerable during the start up time after reboot. This indicates that making MTA fault resilient can significantly benefit the availability of the service provided by the system. Possible suggestion is to integrate additional mechanisms to ensure data consistency in the case of an unsuccessful operation and a system crash (observe (in Table 6) that one of MTA problems is related to internal databases).

6 Modeling Domain Behavior

Analyzing system behavior from the perspective of the whole domain (1) provides a macroscopic view of the system rather than a machine-specific view, (2) helps to characterize the nature of interactions in the network, and (3) aids in identifying potential reliability bottlenecks and suggests ways to improve resilience to operational faults.

Inter-reboot Times. An important characteristic of the domain is how often reboots occur within it. To examine this, the whole domain is treated as a black box, and every reboot of every machine in the domain is considered to be a

reboot of the black box. Table 5 shows the statistics of such inter-reboot times measured across the whole domain.

Item	Value
Number of samples	882
Maximum	2.46 days
Minimum	Less than 1 second
Median	2402 seconds
Average	4.09 hours
Standard deviation	7.52 hours

Table 5: Inter-reboot Time Statistics for the Domain

Figure 3 shows the distribution of inter-reboot times across the domain. Over a third of the reboots in the domain occur within 1000 seconds of each other. Probing these reboots further, about 25% of them (about 8% of the total) are due to the same machine being rebooted multiple times. Also, only about 43% of them occur during the working hours. So, it appears that the majority of such reboots are primarily due to planned shutdowns for maintenance. However, it is also possible that some of the reboots are correlated and that they are due to common cause. Such reboots are indications of propagated failures across the domain.

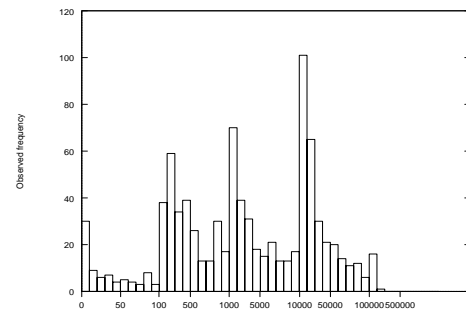


Figure 3: Reboot Time Distribution for the Domain

6.1 Finite State Model of the Domain

The proper functioning of the domain relies on the proper functioning of the PDC and its interactions with the Backup Domain Controllers (BDCs). Thus it would seem useful to represent the domain in terms of how many BDCs are alive at any given moment and also in terms of the PDC being functional or not. Accordingly, a finite state model was constructed as follows: (1) the data collection period was broken up into one hour time windows of a fixed length, (2) for each such time window, the state of the domain was computed, and (3) a transition diagram was constructed based on the state information.

The state of the domain during a given time window was computed by evaluating the number of machines that rebooted during that time window. More specifically, the states were identified as shown in Table 6. Figure 4 shows the transitions in the domain. Figure 4 reveals some interesting insights.

1. Nearly 77% of all transitions from the *F* state, excluding the self-loops, are to the *BDC* state. If these transitions do indeed result in disruption in service, then it is possible to improve the overall availability significantly just by tolerating single machine failures.

State Name	Meaning
PDC	Primary Domain Controller (PDC) rebooted
BDC	1 Backup Domain Controller (BDC) rebooted
MBDC	Many BDCs rebooted
PDC+BDC	PDC and One BDC rebooted
PDC+MBDC	PDC and Many BDCs rebooted
F	Functional (no reboots observed)

Table 6: Domain States and their Interpretation

2. A non-negligible number of transitions are between the *F* state and the *MBDC* state. This would potentially indicate correlated failures and recovery among BDCs. (These transitions are explored in detail later.)

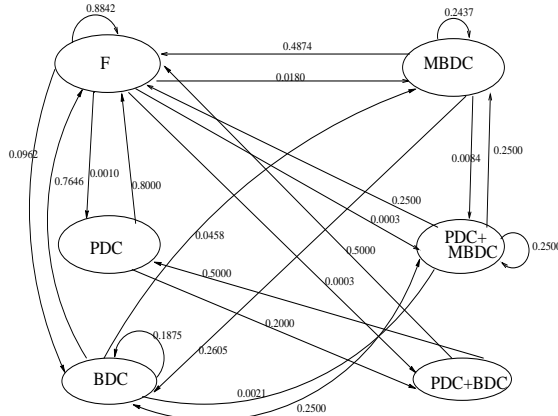


Figure 4: Domain State Transitions

3. Again, a non-negligible fraction of transitions are between states *BDC* and *MBDC*. This would indicate potentially correlated failures and recovery (explored in detail later). Thus, reducing correlated failures would improve service quality and reduce the burden on PDC (i.e., synchronization overhead associated with multiple BDC rebooting at the same time).

4. Majority of transitions from state *PDC* are to state *F*. This could be explained by one of the following: (a) most of the problems with the *PDC* are not propagated to the BDCs, (b) the PDC typically recovers before any such propagation takes effect on the BDCs, or (c) The problems on the PDC are not severe enough to bring it down, but they might worsen as they propagate to the BDCs and force a reboot.

However, 20% of the transitions from the *PDC* state are to the *PDC+BDC* state. So there is a possibility of the propagation of failures.

We obtain further useful insights into the dynamics of the domain by considering what percentage of the total transitions of the domain go into each state. Table 7 shows the details.

Target State	Percentage of Total Transitions
F	85.68
PDC	0.11
BDC	11.04
MBDC	2.74
PDC+BDC	0.04
PDC+MBDC	0.08

Table 7: Domain Transitions by State

From the table, we observe that (1) less than 1% of all transitions involve the reboot of the PDC. So the PDC exhibits a higher availability than the BDCs, (2) over 11% of all transitions are into the *BDC* state. If the domain is configured to tolerate the failure of a single BDC, then disruption of mail service could be reduced significantly, and (3) a non-negligible percentage (2.74%) of transitions end in *MBDC* state. Since each BDC is responsible for handling part of the mail service, the MBDC state represents a potentially serious disruption of service.

7 Error Propagation

Examination of the domain-wide behavior suggests the possibility of correlated/propagated failures. Below, we examine indications of propagation of problems within the domain. It should be noted that the networked-based systems are usually loosely coupled; consequently the propagation of failures may not be as strong or easily identifiable as in tightly coupled systems.

Classification of prominent events. An approach to studying propagation of failures in the domain is to classify each occurrence of the prominent events (i.e., the events upon which reboots in our target system were categorized) as caused by one of the following (1) a problem specific to the local machine (i.e., the machine on which the event was observed), (2) a problem related to a remote machine (this could be true in case of connectivity problems), or (3) a general, network-related problem.

Approach to classifying prominent events. Classification of the occurrences of prominent events was based on tests designed using event-specific information. Individual tests serve as a heuristic used to probabilistically characterize each even as being host or network related problem. These tests were designed as follows. For each event occurrence, the entire set of events occurring in the domain around that event (e.g., within an hour of it) was obtained. Based on the events that were found in this *event window*, the current occurrence of the event was classified into one of the above-mentioned categories.

The anatomy of a test. The tests were generated based on manual analysis of events. However, they were applied to events in an automated fashion. Thus, the test had to be specified in a format conducive to interpretation by a script. A sample test is shown below.

The first line indicates the *event id* and the *source* of the event for which the test is applicable. The keywords *MP_test_begin* and *MP_test_end* denote that the set of entries in between comprise a test for a local machine problem (MP). Each line between these two keywords corresponds to a *test event*. A typical test event description contains the following fields:

- event id (4320),
- source logging the event (NetBT),
- time window within which the event should occur relative to the event being tested (7200) (a plus sign be-

fore the number indicates that the test event should occur after the event being tested), and

- other constraints on the test event observed.

```
Event 2006 Srv
MP_test_begin
4320 NetBT 7200 7 local-machine yes 1 1
3013 Rdr 3600 8 local-machine 7 !local-machine
yes 3 1
5711 NETLOGON +3600 7 local-machine yes 1 2
5715 NETLOGON +3600 7 local-machine yes 1 2
6005 EventLog +3600 7 local-machine yes 1 1
2006 Srv 3600 7 local-machine yes 3 3
MP_test_end
```

The constraints on the test event are better understood with an example. Let us consider the first test event in the sample shown above.

The entries *7 local-machine* indicate that the test will be passed only if an occurrence of the test event is found whose 7th field matches the local machine on which the event being tested was logged. If the same entries read *7 !local-machine*, then the test would be passed only if the 7th field did *not* match the machine name.

The field *yes* indicates that if a test event satisfying all the constraints is found in the event window, then that test is considered passed. A *No* for that field would indicate that the test would pass if no such test event was found.

The last two fields in the test event indicate, respectively, the *number of times* the event must occur (for the test to be passed) and the *group* to which the test event belongs.

Application of the test. A specific instance of a prominent event is considered to be due to a local machine problem if it passes the test designed for that purpose. The tests are applied in *groups*. If any one test in a group is passed by the given event instance, then the whole group is considered to be passed. If the majority of the groups are passed, then the test itself is considered to have passed. The tests for a remote machine problem and a network problem are performed in a similar manner.

Designing a test. The test events are framed based on manual analysis of the events. It is instructive to explore in detail the sample test event depicted above. The event to be tested has an *event id* of 2006 and is logged by the *Srv* (server). The text description for this event states that the *server received an incorrectly formatted request from an unknown source*. To evaluate if a given occurrence of this event represents a problem specific to the local machine, we perform the following tests:

1. Do we observe a problem with the transport service in the local machine? (This corresponds to event 4320.)
2. Do we observe at least three instances of the Redirector service on a remote machine trying to contact the Server on the local machine and reporting a failure? (This corresponds to event 3013.)

3. Does the local machine show evidence of normal network activity within an hour of the 2006 event? (This is indicated by the 5711/5715 event.)

4. Do we observe a reboot of the local machine within one hour of the 2006 event? (This corresponds to event 6005.) The reasoning behind this is that the system administrator might have suspected a problem with the local machine and attempted to solve it by rebooting the machine (possibly after some maintenance).

5. Do we observe at least three instances of 2006 from the same machine in a 1-hour period? (This corresponds to event 2006.)

As evident from the explanation, designing test events involves a heuristic approach. However, in the absence of more specific event descriptions, this seems to be reasonable. In an ideal situation, the accuracy of these tests could be improved by validating the predictions against operator logs that report the actual cause of the problem. However, in the data that was analyzed, no operator log was available.

Results of the tests. Table 8 summarizes the results obtained on selected events. It is observed that most of the identifiable problems are local machine related. This would suggest that propagation of failures is not observed on a regular basis. However, in quite a few cases, the tests were not able to classify the event one way or another. It is possible that some of these unknowns represent propagated failures. Besides, designing such tests requires a fair amount of operating-system-specific knowledge, which is not easy to obtain. However, the idea of designing event-specific tests and using them does appear to be a reasonable approach to implementing automated propagation detection.

Event	Source	Severity	Number	Local Machine Problem	Remote Machine Problem	Unknown
3012	Redirector	1	218	218	0	0
8021	Browser	1	790	261	0	509
2006	Server	1	19586	9907	0	9679

Table 8: Breakup of Selected Events

8 Conclusions

In this paper we present failure analysis of a LAN of Windows NT machines. The study focuses on characterizing causes of machine reboots. The error logs collected by the operating system are investigated to obtain a component-level breakup of the causes of the reboots and to quantify machine uptimes and downtimes. The behavior of a typical machine and the whole network system is modeled as a state transition diagram. Such a model characterizes the dynamic of typical problems encountered in the system and provides useful insights into potentially correlated or propagated failures. Our conclusions from this study are summarized below:

1. Most of the problems that lead to reboots are software related. Only 10% are attributable to specific hardware components.
2. Connectivity problems contribute to the most of reboots. A significant percentage of these problems are persistent.
3. Rebooting the machine does not appear to solve the problem in many cases. In about 60% of the reboots, the rebooted machine reported problems within an hour or two of the reboot. Also, more than a half of the reboots are due to genuine problems with the machine functionality rather than to shutdowns for maintenance.
4. Though the average availability evaluates to over 99%, a typical machine in the domain, on average, provides acceptable service only about 92% of the time.
5. About 1% of the reboots indicate memory leaks in the software.
6. There are indications of propagated or correlated failures. Typically, in such cases, multiple machines exhibit identical or similar problems at almost the same time. Most of these problems are related to the mail service software.

The failure data analysis also provides valuable insights into the error logging mechanism. Event logging features that are absent, but desirable, in Windows NT can be suggested:

1. The presence of a Windows NT shutdown event will improve the accuracy in identifying the causes of reboots. It will also lead to better estimates of machine availability.
2. Most of the events observed in the logs were either due to applications or to high-level system components, such as file-system drivers. It is not evident if this is due to a genuine absence of problems at the lower levels or it is just because the lower-level system components log events sparingly or resort to other means to report events. If the latter is true, then improved event logging by the lower-level system components (protocol drivers, memory managers) can significantly enhance the value of event logs in diagnosis.
3. The Primary Domain Controller logs error events in bursts. These bursts correspond to exchanges of domain-specific information with the BDCs, which is an asynchronous activity. Periodic logging of a healthy event by the PDC would help to increase our understanding of its behavior.

This study convinces us that available event logs provide useful data on failure behavior of Windows NT based systems. On the other hand, we have also learned that in many cases the information contained in event logs is not sufficient to make a definite interpretation. We believe, however, that errors in interpreting the log data towards the conservative side, i.e., we do not overestimate the system.

Acknowledgments

This work was supported in part by the US National Aeronautic and Space Administration (NASA) under grant NAG-1-613, in cooperation with the Illinois Computer Laboratory for Aerospace Systems and Software (ICLASS). We would also like to thank Compaq Computer Corporation for the support extended to this work. We are pleased to thank Fran Baker for her assistance in editing this manuscript.

References

- [1] X. Castillo and D.P. Siewiorek, "A Workload Dependent Software Reliability Prediction Model," *Proc. FTCS-12*, pp.279-286, 1982.
- [2] R. Chillarege, S. Biyani, and J. Rosenthal, "Measurement Of Failure Rate in Widely Distributed Software," *Proc. FTCS-25*, 1995.
- [3] J. Gray, "A Census of Tandem System Availability between 1985 and 1990," *IEEE Trans. Reliability*, Vol. 39, No. 4, pp. 409-418, 1990.
- [4] M.C. Hsueh, and R.K. Iyer, "A Measurement-based Model of Software Reliability in a Production Environment," *Proc. 11th Annual Int'l Computer Software & Applications Conference*, pp. 354-360, October 1987.
- [5] R.K. Iyer and D.J. Rossetti, "Effect of System Workload on Operating System Reliability: A Study on IBM 3081," *IEEE Trans. Software Engineering*, Vol. SE-11, No. 12, pp. 1438-1448, 1985.
- [6] R. Iyer, D. Tang, "Experimental Analysis of Computer System Dependability," *Chapter 5 in Fault Tolerant Computer Design*, D.K. Pradhan, Prentice Hall, pp.282-392, 1996.
- [7] M. Kalyanakrishnam, "Analysis of Failures in Windows NT Systems," Master Thesis, Technical report CRHC 98-08, University of Illinois at Urbana-Champaign, 1998.
- [8] I. Lee and R.K. Iyer, "Analysis of Software Halts in Tandem System," *Proc. 3rd Int. Symp. Software Reliability Engineering*, pp. 227-236, 1992.
- [9] I. Lee and R.K. Iyer, "Software Dependability in the Tandem GUARDIAN Operating System," *IEEE Trans. on Software Engineering*, Vol. 21, No. 5, pp. 455-467, 1995.
- [10] T.T. Lin, D.P. Siewiorek, "Error Log Analysis: Statistical Modeling and Heuristic Trend Analysis," *IEEE Trans. Reliability*, Vol. 39, No. 4, pp.419-432, 1990.
- [11] J.F. Meyer and L. Wei, "Analysis of Workload Influence on Dependability," *Proc. FTCS-18*, 1988.
- [12] R.A. Maxion, "Anomaly Detection for Diagnosis," *Proc. FTCS-20*, pp.20-27, 1990.
- [13] S. Mourad and D. Andrews, "On the Reliability of the IBM MVS/XA Operating System," *IEEE Trans. on Software Engineering*, October 1987.
- [14] M.S. Sullivan, R. Chillarege, "Software Defects and Their Impact on System Availability — A Study of Field Failures in Operating Systems," *Proc. FTCS-21*, pp. 2-9, 1991.
- [15] M.S. Sullivan and R. Chillarege, "A Comparison of Software Defects in Database Management Systems and Operating Systems," *Proc. FTCS-22*, pp.475-484, 1992.
- [16] D. Tang and R.K. Iyer, "Analysis of the VAX/VMS Error Logs in Multicomputer Environments — A Case Study of Software Dependability," *Proc. Third Int. Symp. Software Reliability Engineering*, Research Triangle Park, North Carolina, pp. 216-226, October 1992.
- [17] D. Tang and R.K. Iyer, "Dependability Measurement and Modeling of a Multicomputer Systems," *IEEE Trans. Computers*, Vol. 42, No. 1, pp.62-75, January 1993.
- [18] A.Thakur, R.K.Iyer, L. Young, I. Lee, "Analysis of Failures in the Tandem NonStop-UX Operating System," *Proc. Int'l Symp. Software Reliability Engineering*, pp. 40-49, 1995.
- [19] A.Thakur, R.K.Iyer, "Analyze-NOW — An Environment for Collection and Analysis of Failures in a Network of Workstations," *IEEE Trans. Reliability*, Vol. R-46, No. 4, pp. 561-570, 1996.
- [20] M.M. Tsao and D.P. Siewiorek, "Trend Analysis on System Error files," *Proc. FTCS-13*, pp. 116-119, June 1983.
- [21] P. Velardi and R.K. Iyer, "A Study of Software Failures and Recovery in the MVS Operating System," *IEEE Trans. Computers*, Vol. C-33, No. 6, pp.564-568, June 1984.