Measurement-based availability analysis of Unix systems in a distributed environment

Cristina Simache, Mohamed Kaâniche

Measurement-based Availability Analysis of Unix Systems in a Distributed Environment

Cristina Simache and Mohamed Kaâniche

LAAS-CNRS
7 avenue du Colonel Roche
31077 Toulouse Cedex 4 — France
{crina, kaaniche}@laas.fr

Abstract

This paper presents a measurement-based availability study of networked Unix systems, based on data collected during 11 months from 298 workstations and servers interconnected through a local area computing network. The data corresponds to event logs recorded by the Unix operating system via the `Syslogd` daemon. Our study focuses on the identification of machine reboots and the evaluation of statistical measures characterizing: a) the distribution of reboots (per machine, time), b) the distribution of uptimes and downtimes associated to these reboots, c) the availability of machines including workstations and servers, and d) error dependencies between clients and servers.

1. Introduction

Today, computing environments are mainly based on distributed and interconnected systems and networks. These computing environments are designed to support different sorts of traffic, from interactive terminal and mail service, to printing and file service. The analysis of failures occurring in such environments is needed to ensure that the availability and quality of service provided to the users fulfils their expectations.

There is no better way to understand the dependability characteristics of an operational computing system than by direct measurement, analysis and assessment. Measuring a real system means monitoring and recording naturally occurring errors and failures in the system while it is running under user workloads. The data collected from such measurements provides valuable information on actual error/failure behavior, and can be used to quantify dependability measures and identify system bottlenecks.

Error logs maintained by the operating systems (sometimes called event logs) have been widely used to characterize the behavior of computing systems in the presence of faults and failures. Event logs are notoriously difficult to analyze. They include a large set of information about the occurrence of different types of events in the corresponding environment. Some of these events result from the normal activity of the target systems, whereas others are recorded when errors and failures affect local or distributed resources. The latter events are particularly useful for the availability analysis.

Although several studies have been published on dependability analysis based on error logs, there is still a lack of papers addressing networked systems. In this paper, we present a measurement-based availability study of networked Unix systems, based on data collected during 11 months from 298 Unix workstations and servers in a local area computing network. The target systems run different SunOS and Solaris versions. The data corresponds to event logs recorded by the Unix operating system via the `Syslogd` daemon.

Identifying trends from large event logs is a long process that requires thorough manual analyses. In the current stage of our study, we have focused on the identification of machine reboots and the evaluation of statistical measures characterizing: a) the distribution of reboots (per machine, time), b) the distribution of uptimes and downtimes associated to these reboots, c) the availability of machines including workstations and servers, and d) error dependencies between clients and servers.

The paper is organized as follows. Section 2 discusses related work. Section 3 presents the target system architecture. Section 4 presents the strategy that we developed to collect the data. Section 5 focuses on data processing and presents several results obtained from the data. Finally, Section 6 concludes the paper and presents future work.

2. Related work

Measurement-based dependability analysis of computer systems, using data collected from the field, or event logs maintained by the operating system, have given rise to a wide variety of research. Various types of systems have been studied, including mainframes and largely deployed commercial systems [3, 4, 6, 9, 11, 13, 15, 18]. Also, several techniques have been proposed to support the processing and analysis of large event logs [5, 7, 17]. A

comparative analysis of some of these techniques is reported in [2].

The main issues addressed in these studies include:

- Investigation of the classes of errors/failures reported in the field, their relative importance, and the correlation among errors;
- Analysis of error/failure inter-arrival times and recovery times distributions;
- Correlation between failures and system workload;
- Analysis and modelling of software and hardware error detection & recovery mechanisms and their efficiency.

An extensive survey detailing these issues is presented in [1].

Distributed and network-based computing systems are notoriously difficult environments in which to detect and diagnose faults. The research reported in [10] provided an elaborate discussion on the problems related to the diagnosis and analysis of network anomalies and proposed a methodology based on the monitoring of performance degradation or deviations from expected behaviour as a means for characterising dependability related problems in networked environments. Dependability analysis of networked applications in distributed environments requires the definition of representative fault models and meaningful dependability measures that characterise the system behaviour as perceived by the users. An example is provided in [19] where a framework for analysing and measuring availability as perceived by the users in client-server distributed environments is defined and illustrated with data collected from different sources (customers, network component providers, university studies, etc.). The lack of real data collected from networked distributed systems is one of the reasons for the lack of published results on dependability analysis and modelling of interconnected systems. Examples of such data are reported in [16], where event logs collected from a network of 69 Sun workstations monitored over a period of 32 weeks are analysed, and in [8], where availability analyses are carried out based on event logs collected from 70 Windows NT mail servers.

Clearly, additional measurement-based results are needed to understand the dependability characteristics of networked distributed systems and to give better insights into the problems that one can face when processing and analysing event logs. Such issues are discussed in this paper based on event logs collected from a network of 298 SunOS and Solaris workstations and servers over a period of 11 months.

3. Target system architecture

The LAAS computing network is composed of a large set of heterogeneous workstations and servers interconnected through an Ethernet-based LAN. These systems are organized into subnets, according to their physical location and the research group they belong to. The subnets are interconnected through dedicated communication switches to a central switch. The latter provides connectivity to the servers shared by the whole network (SMTP, NIS+, Backup, HTTP, FTP, etc.) as well as to the Internet. Some of these servers are replicated on several machines (e.g. NIS+), and some machines host more than one server. In addition, each research group has a set of servers dedicated to its users (NFS, POP, Application, Printing, etc.).

Most of the network and group servers are implemented on SunOS and Solaris machines. The clients are a heterogeneous mix of Unix workstations from vendors such as Sun, HP, IBM, as well as PCs and Macintoshes.

In this study, we collected data from all SunOS and Solaris workstations and servers. We monitored 298 machines running different versions of the Unix operating system: 4.1.2, 4.1.3, 4.1.4, 5.4, 5.5, 5.5.1, 5.6 and 5.7.

4. Data collection

4.1. Event logging in Unix

The Unix operating system offers capabilities for event logging by means of the `syslogd` daemon. This background process records events generated by different local sources: kernel, system components (disk, network interfaces, memory), daemons and application processes that are configured to communicate with `syslogd`. Different types of events of various severity levels are generally recorded. Some of them result from the normal activity of the system whereas others provide information about hardware, software and configuration errors as well as system events such as reboots and shutdowns.

The configuration file `/etc/syslog.conf` specifies the destination of each event received by `syslogd`, depending on its severity level and its origin. The destination could be one or several log files, the administration console or the operator (notified by email).

The events that are relevant to our study are generally stored in the log file `/var/adm/messages`. Each event stored in this log file is formatted as follows:

- Date and time of the event
- Machine on which the event is logged
- Description of the message

Example:

```
Dec 15 16:39:29  napoli  unix: server butch not
responding still trying
```

The Unix operating system provides the possibility to automatically control the size of the log files. This is done by executing, on a weekly basis, the script `/usr/lib/newsyslog`, via the `cron` mechanism. The latter ensures that only the current log file `/var/adm/messages` and those recorded during the last four weeks (named as `messages.0`, `messages.1`, `messages.2` and `messages.3`) remain in the system. Therefore, data is lost if not archived within five weeks.

4.2. Data collection strategy

We have set up a data collection strategy to automatically collect the data stored in the `/var/adm/messages` log file of each SunOS and Solaris machine connected to the network. This strategy has been defined to take into account the dynamic evolution of the network configuration that is related, for instance to the frequent addition or removal of machines, the modification of machine names, etc.

The data collection strategy is decomposed into two main steps:

1) Identification of the list of Unix machines to be included in the data collection process.
2) Collection of data from these machines.

We implemented these steps using `Shell` and `Perl` scripts. They are executed on a weekly basis in accordance with the mechanism provided by the operating system to control log files size.

The identification of Unix machines from which data is collected is based on the analysis of the `hosts.org_dir` master table maintained by the NIS+ server. All IP devices connected to the network, including Unix machines, are declared in this table. However, this table generally contains redundant information that corresponds, for instance, to machines that are declared under different IP addresses or with different names. The script that we developed automatically detects and eliminates such redundant information to avoid collecting multiple copies of the same log files from the corresponding machines. The script also eliminates from the list of Unix machines those that are not relevant to our study; for example, those used to support offline maintenance activities, or those in specific experimental testbeds.

In the second step of the data collection strategy, the log files are remotely copied from the selected machines to a dedicated machine, collated into a single file that is sorted chronologically and compressed. A verification of the format of the collected data is also done at this step, and an additional field specifying the year is added to the date of each message. This enables easier analyses of data collected over several years.

The data collection script is executed automatically via `cron` each week. However, manual verification is needed when problems affecting some target machines occur during the execution of this script, e.g., these targets may not be alive, or they are alive but due to some local problems the script hangs. If the manual verification is not done, we might lose some data, or the same data may be copied more than once. (See [14] for more detail).

5. Data processing

Data processing consists of: 1) extracting from the log files the information that is relevant to the dependability analysis of the target systems and 2) evaluating statistical measures to identify significant trends. The log files contain a large amount of information that is not always easy to categorize. The identification of events corresponding to errors and the definition of error classification criteria according to the origin of the error or its severity requires a thorough manual analysis of log files. Some examples of classification criteria are presented in [16]. In this paper, we focus on the identification of machine reboots from the event logs and the evaluation of statistics characterizing: a) the distribution of reboots (per machine, time), b) the distribution of uptimes and downtimes associated to these reboots, c) the availability of machines including workstations and servers, and d) error dependencies between clients and servers.

5.1. Identification of reboots

Three methods can be distinguished to identify when Unix machines are rebooted:

1) Use of `"last reboot"` command
2) Analysis of `/var/adm/wtmp` log files
3) Analysis of `/var/adm/messages` log files

With the first and second methods, only the start timestamp of machine reboots can be identified. However, in our study, we are interested in identifying the start and end timestamps of machine reboots as well as the service interruption duration associated to these reboots. Moreover, the analysis of the causes of reboots can be done based on the analysis of the messages logged by the system before the machine is rebooted. Therefore, we have developed an algorithm providing such information based on the analysis of the `/var/adm/messages` log files collected from the target systems included in our data collection environment.

A manual analysis of collected data revealed that not all reboots can be easily identified from the corresponding log files. Indeed, whereas some reboots are explicitly identified by a "reboot" or a "shutdown" event, many others can be detected only by identifying the sequence of initialization events generated by the system when it is restarted.

Generally, an initialization sequence of the system is composed of about 70 messages, starting with "`unix: SunOS Release…`" or "`unix: Copyright…`"[1] messages, and ending with clock synchronization messages generated by the `ntpdate` and `xntpd` daemons. An example of such a sequence is presented in Figure 1.

[1] Note that these messages may appear several times in the sequence.

```
2000 Jan 31 08:16:03 ripolin unix: Copyright © 1983-1996, Sun Microsystems, Inc.
2000 Jan 31 08:16:03 ripolin unix: SunOS Release 5.5.1 Version Generic_103640-29 [UNIX® System V
Release 4.0]
2000 Jan 31 08:16:03 ripolin unix: root nexus = SUNW,SPARCstation-4
2000 Jan 31 08:16:03 ripolin unix: Ethernet address = 8:0:20:82:23:f
2000 Jan 31 08:16:03 ripolin unix: avail mem = 61521920
……
2000 Jan 31 08:16:04 ripolin unix: SunOS Release 5.5.1 Version Generic_103640-29 [UNIX®
System V Release 4.0]
2000 Jan 31 08:16:04 ripolin unix: Copyright © 1983-1996, Sun Microsystems, Inc.
……
2000 Jan 31 08:16:13 ripolin unix: vol0 is /pseudo/vol@0
2000 Jan 31 08:16:13 ripolin unix: pseudo-device: vol0
2000 Jan 31 08:16:18 ripolin ntpdate[228]: step time server 140.93.0.15 offset 0.676382 sec
2000 Jan 31 08:16:23 ripolin xntpd[231]: xntpd 3-5.93 Tue Jul 6 18:01:08 MET DST 1999 (1)
2000 Jan 31 08:16:24 ripolin xntpd[231]: sched_setscheduler(): Operation not applicable
2000 Jan 31 08:16:25 ripolin xntpd[231]: tickadj = 5, tick = 10000, tvu_maxslew = 495, est. hz=100
```

Figure 1. Initialization sequence

```
2000 Jan 16 18:23:02 demeter unix: SunOS Release 5.7 Version Generic 64-bit [UNIX® System V
Release 4.0]
2000 Jan 16 18:23:02 demeter unix: Copyright © 1983-1998, Sun Microsystems, Inc.
2000 Jan 16 18:23:02 demeter unix: mem = 655360K (0x28000000)
……
2000 Jan 16 18:23:05 demeter unix: vol0 is /pseudo/vol@0
2000 Jan 16 18:22:56 demeter ntpdate[269]: step time server 140.93.0.15 offset -15.890010 sec
2000 Jan 16 18:23:01 demeter xntpd[273]: xntpd 3-5.93 Tue Jul  6 18:01:08 MET DST 1999 (1)
2000 Jan 16 18:23:02 demeter xntpd[273]: kvm_open failed
```

Figure 2. ntpdate message with negative offset at the end of a reboot

Nevertheless, we have identified several scenarios that do not fit the initialization sequence presented in Figure 1. Such scenarios occur for example: a) when multiple reboots are needed before the machine can restore its normal functioning state, or b) when the time synchronization messages do not appear in the corresponding sequence, or their timestamp precedes the timestamp of the messages identifying the start of the sequence. Typically the latter case corresponds to synchronization events with a negative offset value. An example of such scenario is presented in Figure 2. It can be seen that the timestamp of the `ntpdate` message (Jan 16 18:22:56) precedes the timestamp of the "`unix: SunOS Release…`" message because the negative value of the offset (-15.89).

To identify reboots from the log files, we have developed an algorithm, implemented in `Perl`, that is based on the sequential parsing and matching of each message in the collected log files to specific patterns or sequence of patterns characterizing the occurrence of reboots. These patterns correspond to explicit reboot messages or to sequence of events generated during the initialization of the system, as explained above. The algorithm is complex; it is detailed in [14]. This algorithm gives, for each reboot detected in the log file and for each machine, the timestamp of the start and of the end of reboot.

5.2 Results

The data collection strategy described in §4.2 allowed us to collect the event log files from 298 Unix workstations and servers running various SunOS and Solaris versions. Data collection started in October 1999 and it is still going on. In this paper, we present the results obtained from the analysis made on the first 11 months of data (about 368MBytes). It is noteworthy that data collection concerned only 34 Unix machines during the first two months, then it has been extended to all Unix machines connected to the network (i.e., 298).

The execution of the reboot identification algorithm on the collected data detected 2613 reboots for the whole network. To validate our algorithm we have compared, for each machine and each reboot recorded, the timestamp corresponding to the start of the reboot with the information contained in the `wtmp` log file of the corresponding machine. This comparison revealed that both results match in 99.7% of cases. This result increases our confidence in the validity of our algorithm. The advantage of our algorithm is that it gives an estimation of reboot duration. In the following we present various analyses of the 2613 reboots detected.

5.2.1 Distribution of reboots

The distribution of reboots among the machines that we monitored during the data collection period is presented in Figure 3. About 74% of the Unix machines included in our data collection environment had less than 10 reboots; among these only one machine has not been rebooted during the corresponding period. Moreover, as illustrated in Figure 4, it can be seen that about 50% of the reboots were caused by 80% of the machines. This shows that the detected reboots are not uniformly distributed among the machines considered in our data collection environment.

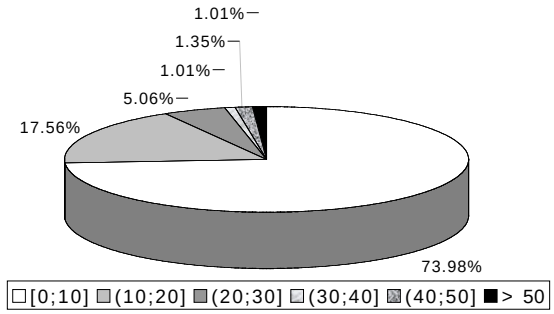

Figure 3. Distribution of number of reboots per machine

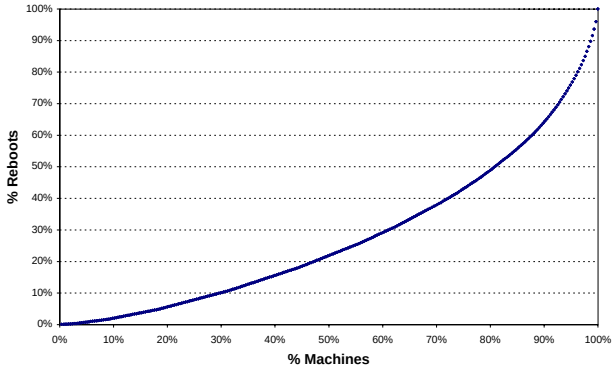

Figure 4. Percentage of reboots observed on percentage of machines

The Unix machines included in our data collection environment can be partitioned into 14 groups, each one corresponding to a particular group of users. The analysis of the distribution of reboots for each group gives an insight into the behavior of the corresponding machines as perceived by their users. Table 1 gives the distribution of machine reboots considering the group they belong to, together with the number of machines in each group and the average number of reboots per machine observed for the group. Whereas the average number of reboots per machine is around 8.77, there is a great degree of variation among the groups. This is mainly related to the fact that the user operational profiles corresponding to each group are different. They are mainly related to the types of software used and the user workloads.

The impact of the user workload on the distribution of reboots can be also observed when analyzing the distribution of reboots as a function of the hour of the day when the reboots occur. This is illustrated in Figure 5. This figure shows that the reboot rate follows the utilization rate of machines, the majority of the reboots occur during the "normal" working hours (8AM to 6PM). A similar result was observed in [12] for the case of four categories of VAX machines. Between 8AM and 6PM we have a uniform distribution, after 6PM the distribution has an exponential shape, but the corresponding number is not negligible compared with the number of reboots observed during normal working hours.

Group ID	Number of machines (NM)	Number of Reboots (NR)	NR/ NM
G1	12	99	8.25
G2	3	21	7.0
G3	17	113	6.64
G4	26	186	7.15
G5	7	46	6.57
G6	28	198	7.07
G7	20	202	10.1
G8	12	67	5.58
G9	28	337	12.0
G10	32	354	11.0
G11	5	34	6.8
G12	66	431	6.53
G13	15	305	20.34
G14	27	220	8.14
Total	298	2613	8.77

Table 1. Number of machines and reboots per group

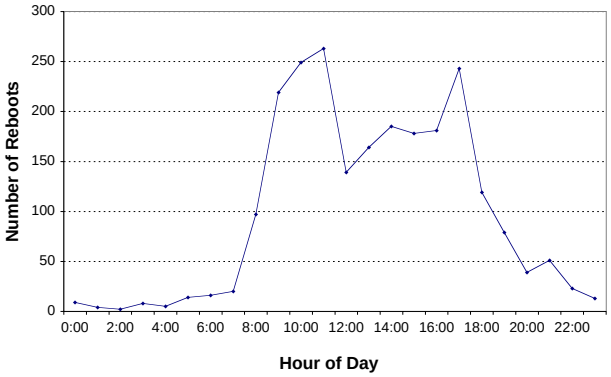

Figure 5. Number of reboots *vs* hour of day

5.2.2 Availability

In this section, we present various statistics characterizing the uptimes, downtimes and the availability of individual Unix machines, the groups of machines and the overall computing network. Similarly to the approach used in [8], machine uptimes and downtimes estimates are obtained as follows:

- For each reboot, the timestamps of the end of the reboot and of the event immediately preceding the reboot are recorded (this would be the last event logged by the machine before it goes down);
- Each downtime estimate is obtained by the time difference between the timestamp of the end of reboot and the timestamp of the event preceding the reboot;
- Each uptime estimate corresponds to the time interval between two successive downtimes.

Using uptime and downtime estimates, we can evaluate availability measures for each machine, for a

group of machines, or for the whole network. The availability and unavailability measures for machine i are computed with the following formula:

$$A_i = \sum uptime_i / \sum (uptime_i + downtime_i)$$

$$\overline{A}_i = 1 - A_i$$

For a group (named GR) of N machines, the mean availability and unavailability measures are computed from the availability measures of the machines belonging to the group:

$$A_{GR} = \sum_{i=1}^{N} A_i \Big/ N \quad and \quad \overline{A}_{GR} = 1 - A_{GR}$$

Usually, availability is expressed as a percentage value and unavailability is represented as an amount of downtime per year (e.g., in number of days per year).

Table 2 presents the statistics characterizing machines uptime and downtime estimates, considering all the systems included in our data collection environment. The corresponding distributions (in a logarithmic scale) are plotted in Figure 6. The results show a large variation of uptime and downtime values. Considering the median values, 50% of uptime values lasted 2.8 days or more and 50% of downtime values lasted 38.5 minutes or less. It is worth noting that about ~24% of uptime durations are lower than 16.6 minutes. These correspond to situations where successive reboots are needed before the corresponding machines restore their functional normal state. It is more realistic in this case to consider that all successive reboots corresponding to uptime values that are lower than 20 minutes correspond to machines that are still in a degraded operation mode. Therefore, these reboots could be represented by a single event, and the downtime value associated to this event should be updated by including the time between the first reboot and the last reboot of the sequence.

	Uptime	Downtime
Minimum	0 sec	0 sec
Maximum	7.6 months	3.2 months
Average	17.7 days	1.7 days
Median	2.8 days	38.5 minutes
Standard deviation	30.9 days	5.7 days

Table 2. Uptime & downtime statistics

In Figure 6, we also observe ~20% of downtime values between 1.15 days and more than 3 months. Some of these values correspond to systems that were not used during a long period of time or that were temporarily disconnected from the network. Such high downtime values might also correspond to machines that were hung for a long period of time due to severe problems, but such problems were not notified to the systems administration team. Unfortunately, the data collected from the event logs do not allow us to determine with enough confidence the real cause of such high downtime values.

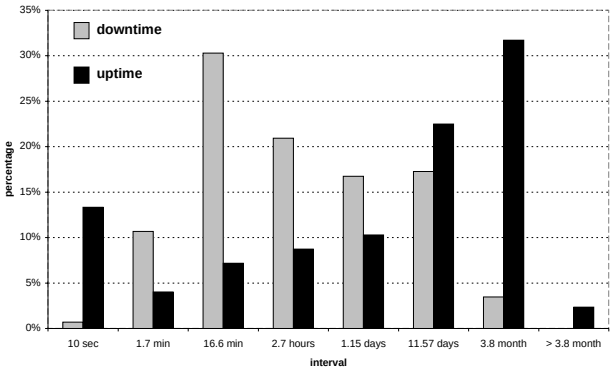

Figure 6. Uptime & downtime distributions

The average availability and unavailability measures for the entire network, computed from the uptime and downtime estimates presented in Table 2, are:

$$A = 89.03\% \ and \ \overline{A} = 40.02 \ days/year$$

Additionally, Table 3 presents the availability and unavailability statistics characterizing the groups of machines as defined in Section 5.2.1. It should be noted that in these availability estimations no distinction was made between critical machines (e.g.; SMTP, POP, NFS shared servers) and user workstations. This explains the relatively low availability measures presented above.

	Min	Max	Average	Median	Std Dev.
A	74.25%	97.71%	89.08%	89.19%	5.34%
$\overline{A}$	8.33d/y	93.97d/y	39.83d/y	39.42d/y	19.50d/y

Table 3. Group Availability/Unavailability statistics

Among the 298 Unix machines included in our data collection environment, 18 machines (denoted as "*main servers*") implement critical services that are shared by the whole network or by a subset of users. 115 reboots have been recorded for those machines. The availability and unavailability estimates corresponding to the main servers are presented in Table 4. In particular, the average unavailability of such servers is 5.27 days/year, which is significantly lower than the value computed for the whole network. This value represents the amount of unavailability that has the most significant impact on the users. Indeed, due to the distributed feature of the network, workstation failures are less critical than the failures of shared servers.

	Min	Max	Average	Median	Std Dev.
A	91.33%	99.99%	98.55%	99.70%	2.66%
$\overline{A}$	0.62d/y	31.62d/y	5.27d/y	1.06d/y	9.74d/y

Table 4. Availability and unavailability statistics for the main servers

5.2.3 Dependencies among machines

In a distributed interconnected environment, several types of functional dependencies might exist between the interconnected systems. These result from the client–server interactions that occur in such environment. Such dependencies are generally transparent to the users in the absence of failures. However, when failures affecting shared servers occur, errors might propagate and affect several clients. The identification of such failure dependencies is particularly useful for availability analysis from the user point of view.

To identify failure dependencies among the Unix machines, we have conducted three types of analyses:

1) Each time a machine is rebooted, all other machine reboots that occur within a short period of time before or after this event are identified
2) Each time a machine is rebooted, all the clients that are affected by this reboot within a short period of time are identified. This is done by identifying from the event logs all the machines that log a "Server Not Responding (SNR)"-like message related to the rebooted machine. Examples of such messages are:

```
unix: NFS server A not responding still trying
automountd[234]: server B not responding
statd[789]: statd host A failed to respond
```

3) Each time a client records a SNR message corresponding to server A, all other clients that record SNR messages related to the same server within a short period of time are identified

In the following, we present the algorithms that we developed and implemented in `Perl` to carry out these analyzes and we discuss the corresponding results.

5.2.3.1. Dependencies related to reboot events

We developed a clustering algorithm that identifies for each machine reboot event, all the reboots of other machines that occur within ±20 minutes of the corresponding reboot event. If multiple reboots of the reference machine occur in less than 20 minutes, they are grouped and the associated time window is updated as follows: it starts 20 minutes before the first reboot and ends 20 minutes after the last reboot. For the time window associated to each reboot, we evaluate:

1) How many different machines recorded a reboot event in the corresponding window. We call this analysis *Reboot-vs-Reboot* dependencies;
2) How many different machines recorded a SNR message informing that the rebooted machine is not responding. We call this analysis *Reboot-vs-SNR* dependencies.

Considering the *Reboot-vs-Reboot* dependency analysis, each machine reboot event leads to the generation of a cluster containing the list and the number of machines rebooted in the time window associated to this event. This number doesn't include the machine corresponding to the reboot event. If one machine belonging to this list is rebooted more than once in the same window, it is counted only once.

Figure 7 plots the distribution of the number of machines rebooted per cluster. The total number of clusters is 2322. The median value shows that for 50% of reboot events, only one other machine was rebooted within ±20 minutes of the corresponding event. It is noteworthy that among the 84.75% of clusters with less than five machines, 1000 clusters have a zero size (i.e., no other machine was rebooted around the reference reboot event). On the other hand, only 8.32% of clusters involve more than 15 machines. These results show that machine reboots are loosely correlated.

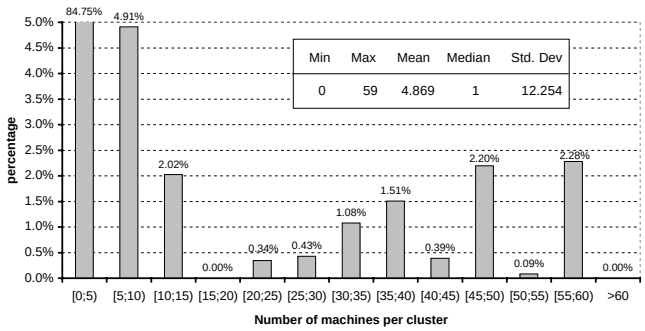

Figure 7. Reboot-vs-Reboot cluster size distribution

Even though most of the clusters involve a small number of machines, this does not preclude the existence of strong dependencies with respect to reboot events among some pairs of machines. Such dependencies can be identified by evaluating the conditional probabilities p_{ij}, that if machine i is rebooted, then machine j will be rebooted within ±20 minutes of this event. Probabilities p_{ij} are estimated as follows:

$$p_{ij} = \frac{\textit{number of times i and j are rebooted together}}{\textit{number of times i is rebooted}}$$

It is noteworthy that, in most of the cases, $p_{ij} \neq p_{ji}$. We computed these probabilities for all (i,j) machine couples, the distribution function characterizing this set of values is plotted in Figure 8, the corresponding number of (i,j) couples is 9394.

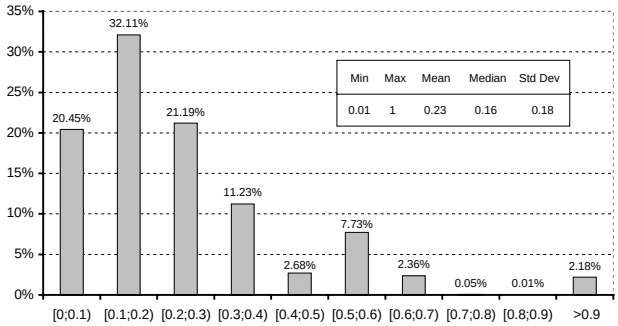

Figure 8. Distribution function of p_{ij} values

The results show that 12.34% of p_{ij} values are between 0.5 and 1. This confirms that there is a subset of machines that exhibit significant correlation with respect to reboot events. A detailed analysis of these (i,j) couples with $p_{ij} \geq 0.5$ revealed that in 54.96%, machines i and j belong to the same research group, and in 32.18% they belong to different groups, but they are physically located in the same building (i.e., they use the same switch to communicate with the rest of the network). Several machines include in particular important servers of the network such as NFS, mail, ftp, www or frequently used application servers.

We followed a similar approach to analyze *Reboot-vs-SNR* dependencies. For each time window corresponding to a reboot event, we identified the list and the number of machines that recorded at least one SNR – like message informing that the rebooted machine is not responding. Figure 9 plots the distribution function of the corresponding numbers. All machines are considered in this analysis, including the workstations and the main servers of the network.

Figure 9 does not reveal strong *Reboot-vs-SNR* dependencies. Indeed, the average number of machines affected by a reboot of another machine is 0.87 and for 91.98% of reboots there wasn't any SNR message referring to the rebooted machine in the event logs collected from the entire network. In fact, SNR messages have been recorded in only 186 clusters out of 2322. These clusters are associated to 61 different machines, only 15 machines among these belong to the set of main servers of the network.

The maximum number of machines affected by a reboot is 162. This case corresponds to the reboot of a critical server of the network that had disk problems during a few months. From Figure 9, we can also identify about 1.13% of cases where the reboot of one machine affected more than 15 machines of the network. These cases correspond to the reboot of some machines that offer various services shared by a subset of users, e.g., NFS servers, application servers, etc. It is noteworthy that these machines correspond to the same servers with p_{ij} values higher than 0.5 as discussed in the previous analysis.

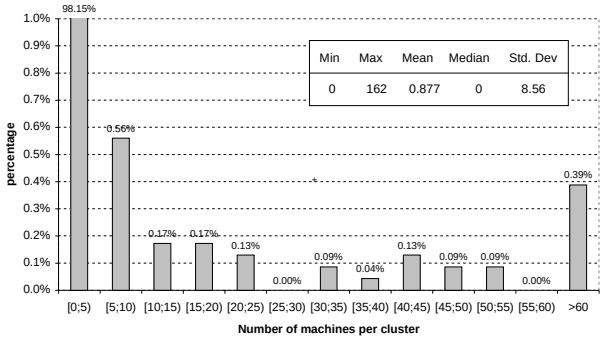

Figure 9. *Reboot-vs-SNR* cluster size distribution

In order to identify the subset of machines that are significantly affected by the reboot of other machines, we have also computed for each pair of machines *(i,j)* the probabilities q_{ij} defined as follows:

$$q_{ij} = \frac{number\ of\ times\ j\ recorded\ SNR(i)}{number\ of\ times\ i\ is\ rebooted}$$

where i corresponds to the machine rebooted and j is the machine recording a SNR message referring to machine i, noted SNR(i*)*. High values of q_{ij} denote a strong coupling of machines i and j, due for instance to client-server interactions.

The number of couples (i,j) for which q_{ij} is non null, is 835. The distribution function characterizing the probabilities q_{ij} and the corresponding statistical parameters are given in Figure 10. Even if the average probability is 0.3, about 17.84% of q_{ij} values exceed 0.5. The reboot events corresponding to these cases concern 19 machines, where only 9 among them belong to the set of main servers.

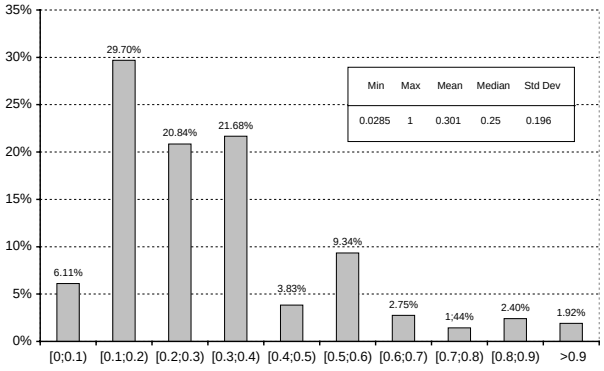

Figure 10. Distribution of q_{ij} values

These results show that only a few machines of the network exhibit strong failure dependencies. In fact, if we analyze the network file system architecture, in particular the information contained in the mount tables, it appears that a larger set of machines exhibit functional and data dependencies. However, these dependencies will not be activated if the clients do not access the servers upon the occurrence of server failures. Similar results were observed in the experimental study presented in [16].

5.2.3.2. Dependencies related to SNR events

This analysis is based only on the SNR messages recorded by the clients. We selected from the data, all such messages without considering if the server machine referred to in the SNR message was rebooted or not. To carry out this analysis, we have parsed sequentially the event logs collected from the network and identified the list of machines that are referred to in the SNR messages, and then we applied a tupling algorithm as follows:

Algorithm:

For each machine i from the list, do

- *if the next SNR message referring to machine i is within $\Delta_{tupling}$, then include message in the current tuple*
- *otherwise, create a new tuple for machine i*

The algorithm provides, for each machine of the list, the total number of tuples created, the duration of each tuple and the number of different machines that recorded the messages included in each tuple. This analysis was carried out for all machines from the list identified above (113 machines) and for a subset of this list corresponding to the main servers of the network (18 machines).

Table 5 gives the number of tuples obtained for all machines and for the main servers for different values of $\Delta_{tupling}$. Columns 2 and 3 of this table show that about 60% of the tuples are associated to a main server. The remaining tuples correspond to machines that host some applications or contain data that are accessed by a subset of network users.

$\Delta_{tupling}$	#tuples for all machines	#tuples for main servers
20 min	4582	2743
40 min	3985	2409
60 min	3635	2210

Table 5. Number of tuples for different $\Delta_{tupling}$ values

The statistics characterizing the duration of the tuples obtained for all machines are presented in Table 6. If we consider only those tuples associated to the main servers, only the mean and standard deviation values vary (from a few seconds to 10 minutes). The value of zero seconds of the tuple duration for the minimum and the median indicates that the corresponding tuples contain only one message, and only one client is affected by the temporary unavailability of the corresponding machine. These tuples correspond to transient errors that disappear in a short period of time. Regarding the tuples that have a long duration (the maximum value is around 25 hours), the number of clients affected is not necessarily correlated to the duration of the tuples. Indeed, a thorough investigation of these tuples showed that in some cases only a few clients are affected but the error manifests for a long time, and for other cases the tuple duration is short and the number of clients affected is high. For example, the duration of the tuple with the maximum number of machines (217) is 8.65 hours and for the tuple corresponding to the maximum duration only 61 clients were affected.

$\Delta_{tupling}$	Min	Max	Mean	Median	Std Dev.
20 min	0 sec	24.4 h	5.06 min	0 sec	29.9 min
40 min	0 sec	25.9 h	10.1 min	0 sec	40.0 min
60 min	0 sec	25.9 h	15.9 min	1 sec	51.0 min

Table 6. Tuple duration statistics

Figure 11 plots the distribution of the number of machines per tuple corresponding to a tupling interval of 20 minutes. Similar distributions are obtained for 40 and 60 minutes tupling intervals. This figure highlights the fact that only ~1.2% of tuples contain more than 15 machines. Most of the tuples involve less than five machines (~96%). This high percentage suggests that even though the clients depend on several resources distributed over the network, most of the errors affecting the main servers persist during a short period of time and are not perceived by a large proportion of network users. This result also reflects the need to take into account the users operational profile to analyze and assess the impact of server failures from the users perspective. Indeed, if the servers are not accessed when errors occur, these errors are transparent to the users.

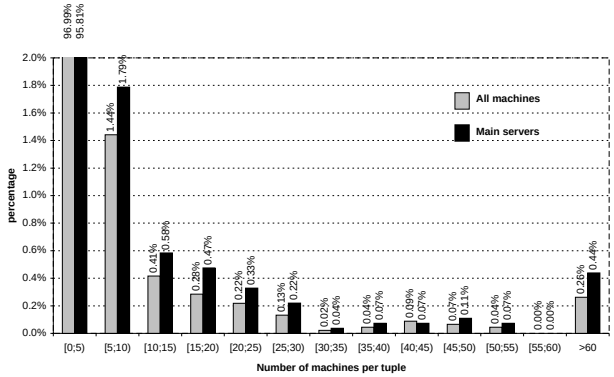

Figure 11. Number of machines per tuple distribution

7. Conclusion

This paper presented the results of an availability study based on error data collected during 11 months from 298 Unix workstations and servers interconnected through a LAN. This study focused on the identification of machine reboots from the event logs recorded by the operating system and the evaluation of statistical measures characterizing the availability of the target systems, and error dependencies between clients and servers. Several techniques and algorithms were presented to extract relevant information from the event logs and to carry out the analyses described above. The investigation of error propagation among the machines included in our environment revealed low correlation between clients and servers. This can be explained in part by the fact that errors affecting the servers persist for a short period of time and thus only clients accessing the servers when errors occur might be affected.

The results presented in this paper allowed us to analyze the availability of the Unix machines connected to our computing network. Similar analyses are currently being performed on Windows NT and 2000 systems. Our objective in the future is to take into account other types of systems that do not provide event logging mechanisms, e.g., MacOS based systems and network communication devices. For these systems, the data needed for availability analysis can be obtained through on-line monitoring, that consists of testing the availability of the corresponding systems, on a regular basis, using dedicated probes (ping, TCP connections on specific ports, etc.). This approach is currently under study.

Acknowledgement: The work presented in this paper has benefited from fruitful discussions with Yves Crouzet and Karama Kanoun. The authors are indebted for their helpful comments.

This work is partially supported by the European Community (Project IST-1999-11585: DSoS — Dependable Systems of Systems).

References

[1] J. Arlat, J. C. Fabre, V. Issarny, M. Kaâniche *et al.*, "Dependable Systems of Systems: State of the Art Survey", DSoS Project, IST-1999-11585 LAAS Report N° 00353, 2000.

[2] M. F. Buckley, D. P. Siewiorek, "A Comparative Analysis of Event Tupling Schemes", *26th Int Symp. on Fault-Tolerant Computing (FTCS-26),* (Sendai, Japan), pp. 294-303, IEEE Computer Society, 1996.

[3] S. E. Butner, R. K. Iyer, "A Statistical Study of Reliability and System Load at SLAC", *10th Int. Symp. on Fault Tolerant Computing (FTCS-10),* (Kyoto, Japan), IEEE Computer Society, 1980.

[4] J. Gray, "A Census of Tandem System Availability Between 1985 and 1990", *IEEE Transactions on Reliability*, vol. R-39, pp. 409-418, 1990.

[5] J. P. Hansen, D. P. Siewiorek, "Models of Time Coalescence in Event Logs", *22nd Int. Symp. on Fault-Tolerant Computing (FTCS-22),* (Boston, MA, USA), pp. 221-227, IEEE Computer Society, 1992.

[6] R. K. Iyer, D. J. Rossetti, "Effect of System Workload on Operating System Reliability: A Study on IBM 3081", *IEEE Transactions on Software Engineering*, vol. SE-11, pp. 1438-1448, 1985.

[7] R. K. Iyer, L. T. Young, V. Sridhar, "Recognition of Error Symptoms in Large Systems", *1986 IEEE/ACM Fall Joint Computer Conference,* (Dallas, TX, USA), pp. 797-806, IEEE-ACM, 1986.

[8] M. Kalyanakrishnam, Z. Kalbarczyk, R. K. Iyer, "Failure Data Analysis of a LAN of Windows NT Based Computers", *18th IEEE Symp. on Reliable Distributed Systems (SRDS-18),* (Lausanne, Switzerland), pp. 178-187, IEEE Computer Society, 1999.

[9] I. Lee, R. K. Iyer, D. Tang, "Error/Failure Analysis Using Event Logs from Fault Tolerant Systems", *21st Int. Symp. on Fault Tolerant Computing (FTCS-21),* (Montreal, Canada), pp. 10-17, IEEE Computer Society, 1991.

[10] R. A. Maxion, F. E. Feather, "A Case Study of Ethernet Anomalies in a Distributed Computing Environment", *IEEE Transactions on Reliability*, vol. 39, pp. 433-443, 1990.

[11] S. R. McConnel, D. P. Siewiorek, M. M. Tsao, "The Measurement and Analysis of Transient Errors in Digital Computer Systems", *9th Int. symp. on Fault-Tolerant Computing,* (Madison, Wisconsin,), pp. 67-70, IEEE Computer Society, 1979.

[12] P. Moran, P. Gaffney, J. Melody, M. Condon, M. Hayden, "System Availability Monitoring", *IEEE Transactions on Reliability*, vol. R-39, pp. 480-485, 1990.

[13] D. P. Siewiorek, V. kini, H. Mashburn, S. R. McConnel, M. M. Tsao, "A Case Study of C.mmp, Cm*, and C.vmp: Part I — Experience with Fault Tolerance in Multiprocessor Systems", *Proceedings of the IEEE*, vol. 66, pp. 1178-1199, 1978.

[14] C. Simache, M. Kaâniche, "Dependability of Interconnected Unix Systems: Data Collection and Analysis",, LAAS-Report 01130, 2001 *(in French)*.

[15] D. Tang, R. K. Iyer, S. Subramani, "Failure Analysis and Modeling of a VAXcluster System", *20th Int. Symp. on Fault Tolerant Computing (FTCS-20),* (Newcastle Upon Tyne, UK), pp. 244-251, IEEE Computer Society, 1990.

[16] A. Thakur, R. K. Iyer, "Analyze-NOW — An Environment for Collection & Analysis of Failures in a Network of Workstations", *IEEE Transactions on Reliability*, vol. 45, pp. 561-570, 1996.

[17] M. Tsao, D. P. Siewiorek, "Trend Analysis on System Error Files", *13th International Symposium on Fault-Tolerant Computing (FTCS-13),* (Milano, Italy), pp. 116-119, IEEE Computer Society, 1983.

[18] A. S. Wein, A. Sathaye, "Validating Computer System Availability Models", *IEEE Transactions on Reliability*, vol. 39, pp. 468-479, 1990.

[19] A. Wood, "Predicting Client/Server Availability", *Computer*, pp. 41-48, 1995.