Metrics for Management

Christine Hogan
Synopsys, Inc.

Metrics for Management

Christine Hogan – Synopsys, Inc.

ABSTRACT

We were recently asked by management to produce an interactive metric. This metric should somehow measure the performance of the machine as the user perceives it, or the interactive response time of a machine. The metric could then be used to identify unusual behavior, or machines with potential performance problems in our network. This paper describes firstly how we set about trying to pin down such an intangible quality of the system and how we produced graphs that satisfied management requirements.

We also discuss how further use can be made of the metric results to provide data for dealing with user reported interactive response problems. Finally, we relate why this metric is not the tool for analyzing system performance that it may superficially appear to be.

Introduction

Metrics are being used to produce charts and graphs of many areas of our work. For instance, at Synopsys, the number of calls opened and closed in a given day or month are charted, as are the number resolved within a given time period. There are also metrics that monitor more specific areas of our work, such as new hire installs, and more general things, such as customer satisfaction ratings. More metrics are constantly being devised by management, and put into place by the systems staff. One such metric was the ''interactive metric'', which was intended to measure the interactive response time of a machine and highlight any problems that might require further investigation.

While it is possible to monitor a number of different components of the system that could influence system response time [1, 2, 3], there are currently no tools available to monitor the performance of the system as the user perceives it. We were asked to develop such a tool. The interactive metric was meant to imitate a user as closely as possible, and measure how long it took to perform a ''typical'' action.

In this paper, we first provide some background information on the particular installation in which this metric was to be installed and describe some of the issues that arose in its design. Then we present the actual design of the metric. Thereafter we focus on the issues that arose in interpreting and presenting the data that the metric gathered. These issues, in fact, were the key ones when it came to presenting the results to management and determining to what extent we, as system administrators, could find them useful. The paper concludes with a discussion of the limitations of the metric, in particular from a system administrator's perspective, and describes some possible extensions that could enhance its current utility.

Background

In this section we present a description of the site at which the metric is running. Also included are some tables that relate machine names to architectures, fileservers, subnets and the role of the machine. This data provides some insight into the graphs produced later in the paper. The motivation and intentions behind the development of the metric are also briefly mentioned.

The Site

The site at which this metric was developed is in Mountain View, California, in the head office of a company with a number of sales offices throughout the US, Europe and Asia. Most of these sales offices are on the company WAN. The Mountain View campus network has a services backbone, to which the shared servers are connected [4]. The servers are often dual-homed with the other interface being on a departmental subnet. Each department has its own subnet, with a number of fileservers and compute servers, as well as desktop machines. Desktop machines are a mixture of X-terminals and workstations. The workstations generally have local swap, remote root, and shared, read-only usr partitions. There are also a large number of Macintoshes and PCs in this site. However, the metric is not run on those machines.

The metric was initially tested on a small subset of machines on a few of the subnets. For each subnet that was included in the testing phase a fileserver was selected, along with a number of machines that were known or suspected to be slow, and machines that represented some cross-section of the architectures at the site. The fileserver that was selected for a given subnet was one of the fileservers typically used by the machines on that subnet. Table 1 groups the testing machines into subnets and shows the fileservers used, while Table 2 gives the architecture of the fileserver.

During the initial phases of testing and designing the metric it was also run in a number of WAN-connected sales offices, which are not shown in Tables 1 and 2. The sales machines were dropped since the complaints that were being received from people in the sales offices related to WAN latency issues, and the metric is not a suitable tool for monitoring or graphing the performance of a WAN connection.

Motivation

The idea for such a metric was arose from the observation that sometimes a user will complain that the system is slow, or the network is down. There was a suspicion that ''the network was slow'' was becoming the modern, hi-tech, equivalent of ''the dog ate my homework''. However, without any form of metric or supporting data, it was impossible to try to refute those statements, or defend the state of the system/network as a whole. Therefore, there was a desire to try to quantify the experience of the ''average'' user, and to have some form of data to use for the basis for discussion.

An analogy that is frequently used at Synopsys[1] is the comparison of a network with the freeway system in Los Angeles. At any one time the network, or freeway system, is neither up nor down. Some segments of it may be down, or extremely congested, but others will be in perfect working order. In fact, this analogy extends to the metric. The metric is the equivalent of taking sample trips along certain routes at different times during the day, and measuring the time taken, to get a feeling for ''delay''. We recently discovered that such sampling is one of the methods actually employed by civil engineers in the study of traffic routes and delays.

The idea behind the metric was to develop a tool to measure ''performance'' at a high level, as the user sees it, rather than breaking the system down into a series of components, none of which in itself means anything to the user. The metric was not intended to be an all-purpose instant system and network analysis tool for the system administrator. However, we, the system administration staff,

[1]The inspiration for which was Eric Berglund's, with extensions from Arnold de Leon.

Machine	Subnet	Fileserver	Architecture	Purpose
mingus	72	anachronism	Sun 4/40	Desktop
gaea	64	anachronism	Solbourne S4100dx	Many services
underdog	72	anachronism	Sun SS4	Desktop
kency	72	anachronism	Sun SS10	X-terminal server
amnesia	100	anachronism	Sun SS20	NAC administration
fili	100	dempsey	Sun SS10	CPU server
mahogany	74	dempsey	Sun IPX	Build machine
redwood	74	dempsey	Sun IPX	Build machine
mordor	124	dempsey	Sun SS10/512	Porting
canary	68	anachronism	Sun SS4	Desktop
paris	68	mammoth	Sun 4/40	Desktop
goofus2	68	mammoth	Sun 4/50	Desktop
orac	68	mammoth	Sun 4/40	Desktop
millstone	92	almanac	SS20/61	Xterm Server
droid	92	almanac	SS10	LISP testing
mercury	92	almanac	Sun 4/60	Desktop
sark-92	92	almanac	Sun SS20	CPU server
jose	92	almanac	Sun 4/60	Desktop

Table 1: Subnet numbers and fileservers of tested machines

Server	Subnets	Architecture	Users
anachronism	72, 100	Network Appliance	NCS
dempsey	100, 116	Auspex NS6000	Porting Center
almanac	92, 100	Network Appliance	Design Verification
mammoth	98, 100	Solbourne S6/904	Product Engineering

Table 2: Architecture of the fileservers

thought it would be nice if it did give us some useful feedback in return for the time and effort expended. We will examine to what extent our goals and our hopes were met.

Designing the Metric

There were a number of issues involved in designing the metric. Not only did it necessitate deciding upon a typical set of actions that would be sufficiently non-intrusive, but it also involved determining a typical environment in which these commands would be run. Of particular interest were the differences in performance for different groups of users, who were on distinct subnets and using separate fileservers.

Design Issues

User perceived system performance cannot be simply measured by a single number. A user's view of the performance of a system is formed on the basis of how long it takes to perform a particular job, and thus two different users on the same system may have differing views on the performance of that system [1]. Thus, an essential part of designing a metric that would be of some use to us involved determining what a typical user's job involved. Since a significant portion of our user community is involved in software development, it was decided that the typical action that we would model would be the edit, compile, run cycle. While this series of actions is not typical of all sections of our user community, the only way that we could make comparisons between different networks and fileservers was by running the same metric on all groups of machines. Thus, this set of actions was decided upon, while acknowledging that it limited the usefulness of the results for some sections of the community.

The first phase also involved performing some sanity checks on the metric using data gathered from an initial set of metric runs. This initial examination of the data was intended to verify that the metric was producing at least superficially believable results. It also enabled us to alter it somewhat to emphasize any differences in performance that existed.

Simulating an Interactive Session

To simulate an interactive session, we first tried using Dan Bernstein's `pty` package [5] to both provide ptys for `vi` and to simulate the speed at which a user types. While the results showed some variation from machine to machine, it was not significant, and we did not feel that it represented the differences that we perceived when using the machines. Partial results from this run are shown in Table 3. The metric performed two edits, a `make` of a small project and a short execution run. The results here are correlated with the tables from the ''Background'' section, and it is noted these results provided a sanity check on the metric by displaying longer times for machines that seemed to give poor interactive response. Even ignoring the results produced by gaea, which suffered from some problems during that week, the relative ordering of the other machines at least made sense.

	Total	Edit 1	Edit 2
kencyr	0.837238	0.413107	0.419813
gaea	2.391545	1.100123	1.275356
mingus	1.064980	0.500288	0.563270
canary	0.525883	0.257806	0.267521
amnesia	0.417002	0.200812	0.215840
orac	1.377794	0.716232	0.660213

Table 4: Initial results from the metric using `chat2.pl`

While the relative performances made sense, the absolute differences in the numbers did not. These machines had been selected to show a wide variation. It seemed that the time was dominated by the speed that the `pty` program was playing back keystrokes, and that actually simulating user input was therefore probably not what we wanted, if we were looking for widely dispersed numbers. The metric was altered to use Randal Schwartz's `chat2.pl` package [6] to simulate user interaction. This method of providing input to interactive programs can send the input to the program as quickly as possible, without simulating a delay between keystrokes. Results from an initial run with this altered metric are shown in Table 4. This initial

	Total	Edit 1	Edit 2	Compile	Run
mingus	53.090136	28.652537	6.077702	18.288256	0.059256
gaea	373.525000	332.572501	7.661372	33.211638	0.067266
underdog	40.528716	28.717238	5.917166	5.866199	0.029712
kencyr	43.236301	28.745429	5.941636	8.513442	0.034764
amnesia	38.046358	28.390941	6.440619	3.173970	0.033065
canary	39.296980	28.499463	5.820487	4.952673	0.021444
orac	52.123264	28.618801	6.015613	17.426540	0.067794

Table 3: Initial results of the metric, using `pty`

run was executed in a different week to that in Table 3, but measured the same sequence of commands. These results show a much clearer separation between the machines, and were considered to be more representative of the differences between the machines. Thus the metric was altered to use `chat2.pl` instead of `pty`, in order to emphasize the differences between the machines.

The Execution Framework

This section briefly describes how, based on the configuration described in the ''Background'' section, we set the metric up to run on a number of subnets, using fileservers that we could argue were typically used by people in the group to which each subnet belonged.

The metric is currently being run on each of ten subnets that correspond to the primary business units of the company in Mountain View. For each subnet a number of machines were selected to give us a sample of about three machines of each class of machines present on that subnet. For example, if a subnet had Sun SPARCstation 20s (SS20s) that act as compute servers, Sun SPARCstation 10s (SS10s) that act as X-terminal servers, Sun SPARCstation 5s (SS5s) that are on desktops, Sun SPARCstation IPCs (IPCs) that are on desktops, and SS20s that act as servers for the desktop workstations, there would be five classes of systems on that network, and a sample of each class would be selected. In addition, machines that we expected to be especially heavily or lightly loaded were selected.

Each subnet is used by a particular business unit, which uses a number of fileservers. One of these fileservers contains home directories, whereas work areas containing product can be housed on a number of different fileservers. Thus the decision was made that the fileserver that would be used by the metric on a given subnet would be the one that contains the home directories of the people on that subnet. The metric therefore may not in fact be using the machine that a given user accesses while working on a product. However, it is guaranteed to access a fileserver that every user in the group accesses a number of times during the day. Therefore the chosen fileserver can be said to influence the users' perceptions of system performance, in some way.

Analyzing the Results

In ''Interactive Session'' section, we showed the results as running averages over the course of a week. While this was useful in the initial stages of attempting to produce a metric that gave believable results with significant variations, it yields no further information. The next stage was to produce graphs that were satisfactory from a management perspective, which involved considerable dialogue and a number of revisions in what graphs were produced. We thereafter examined the data produced to see if we could extract any useful information from the users' or the system administrators' perspectives. We discovered that all three goals were quite distinct from each other and involved taking different views of the same data.

Graphs for Management

Initially we produced some graphs that showed the average results on an hour-by-hour basis for each of the internal groups that we were monitoring, where each group was represented by a set of machines on the same subnet, using the same fileserver. Along with these graphs, we also

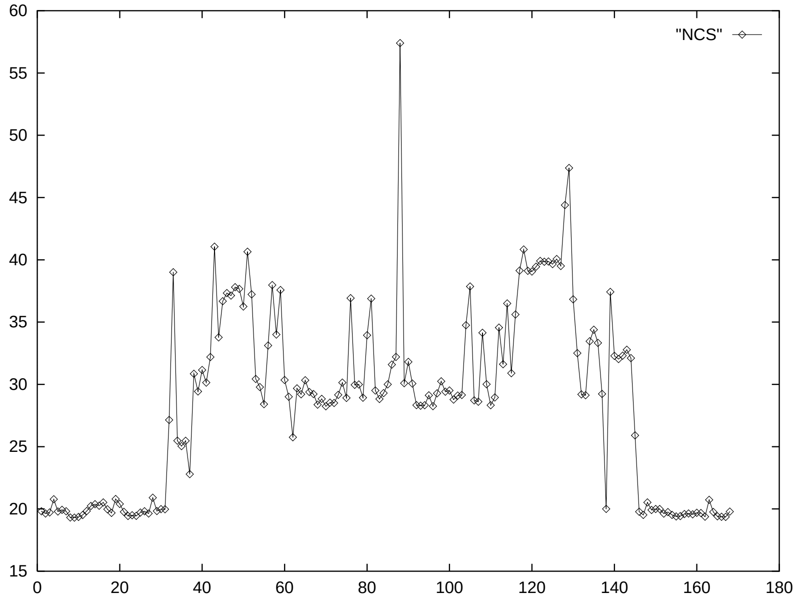

Figure 1: Average results for NCS

produced individual graphs for each of the three best and three worst machines. For example, the graph produced for the Network and Computer Services (NCS) group is shown in Figure 1, and the results from the machine jose are shown in Figure 2. The numbers on the horizontal axis are hours since 0:00 on Sunday, and each graph represents a week. The numbers on the vertical axis represent the length of real-world time that it took to run the metric, in seconds.

Other than noting that the systems in NCS are lightly loaded at weekends, we also noticed that there was generally a trend in the values produced on each machine. What management were interested in were variations from the trend. Since trends are machine-based, we normalized the results from each machine, and plotted the set of machines that were being monitored in a particular group on the same graph.[2] These graphs represent deviations of the machines from their normal behavior.

Another representation of the data that was perceived to be interesting was a stock-market style high, low and average chart, on a day by day basis, based on the normalized results, as described above. This form of graph would highlight machines with a large variation in performance, which could be indicative of a problem.

Relating to the Users

The metric is also being used to gather trend data for a given architecture on a given network. It is thought that this data will be useful as a reference point when discussing performance issues with a user. It should be possible to monitor a machine about which a complaint has been received, compare the data it generates with that of other machines of its class on that subnet, and either state that there is a problem with it, or that it seems to behaving normally for a machine of its class.

Information for the System Administrator

One way of representing the data was to produce a graph for each group that compared the different classes of machines in that group against each other. As expected, the SS20s were shown to be considerably faster than the SS5s or IPXs in these graphs. However, we were able to make a couple of interesting observations from these graphs. Firstly, our SS10s that are used as X-terminal servers yield worse performance than the SS5s that are on people's desks. We also noted that there seemed to be a base ''best'' performance for each class of machines, which seems to be primarily dominated by NFS response time, but is also a function of the class of machine. This ''best'' performance time was universal across our subnets with the exception of the Product Engineering network, on which we were using an Auspex[3] file server, rather than a Network Appliance[4] file server. The machines using the Auspex exhibited marginally worse results across the board for this metric. However, these were brief, once-off observations. Of more use and interest would be the use of the metric as a diagnostic or analytical tool.

[2] Due to the intentional clustering of the data around a single area – the ''normal'' line – these graphs are best viewed in color.

[3] HP IV processor; Storage processor III; File processor II; old-style I/O processor; running 1.6.1M1

[4] FAServer 1400, running 2.1.3

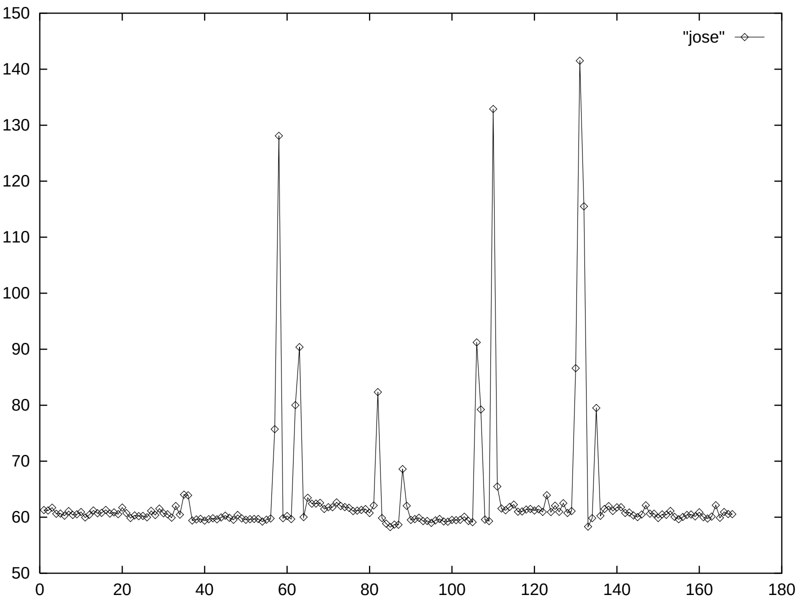

Figure 2: Average results for jose

It has been said [3] that all system performance issues are basically resource contention issues and that the biggest challenge is figuring out which kernel subsystem is really in trouble. In a large site, however, the primary challenge is discovering which individual servers or networks are suffering performance trouble. System administrators may never log into a user's desktop machine. This machine may, however, be central to the user's perception of system performance.

The graphs that are produced for management from this metric do not aid the system administrator in the task of identifying individual machines that have problems because they are based on the usual behavior of each given machine, with no consideration given to absolute performance values. Unusual behavior may, however, be detected from graphs relating the performance of one machine to others of its type, such as the graphs that are being produced for generating concrete data for dialogue between users and system administrators. The utility of both of these sets of graphs for system administrators will be discussed further in next section.

Another potential use of the metric for system administrators is in the area of performance tuning and reconfiguration. Machines at a large site, such as Synopsys, tend to be clones of a standard model. Before changing that hardware model, some testing and benchmarking is performed. This principle can also be applied to performance tuning. A single system could be reconfigured and the performance of the new configuration monitored over a period, with the results being compared with its previous results and those of other machines in that class during the same time period. This data could then be used to justify a decision on whether or not to similarly reconfigure the other machines in that class.

Limitations

While the metric has produced some information that management found interesting, and other information that can be useful in dealing with performance issues that can arise with users, we feel it is of limited use, as it stands, in producing information that a system administrator can use.

The metric can currently be used to detect abnormal behavior of a machine. However, if all of the machines of a particular type are identically mis-configured, and yielding less than the performance of which they are capable, that will not be detected. Even if abnormally bad performance is detected, the metric gives no indication of how the performance of the system may be improved. Also, the numbers that it produces cannot be used directly to say that a given machine is performing well or badly. The metric must be calibrated within the particular environment before the results are interpreted. Grouping machines into classes, and comparing the results of a given machine to those of others in its class during the same time period is a step in the right direction. However, there are many other things that can affect the performance of a system which are not taken into account in these graphs. For example, this metric does not indicate the amount of memory or swap space in a system; it gives no indication of whether a machine was otherwise active or idle at the time; and there is not, currently, any way of correlating the results to network load, or NFS performance of the server during the given time period.

Another data-point that we found interesting was that the SS4s produced better results for the metric than SS5s. We use generally SS5s rather than SS4s at our site as desktop machines, because it was felt that the lack of expansion possibilities, and the slightly different operating system outweighed any advantages that the SS4s might have. Thus, this result demonstrates that better performance is not everything when it comes to choosing a machine for the standard desktop model.

One area that we feel is a major limitation of the metric is the lack of any way test X performance, which is fundamental to a user's perception of system performance in our environment. Another correspondingly significant limitation is the inability to run this metric on our other platforms, such as Macintoshes and PCs.

Extensions

The metric could be extended in a number of ways to provide more information for the systems administrator. In considering possible extensions, we consider the reasons behind the aforementioned limitations. We also consider the approaches taken in monitoring and tuning the performance of a small number of systems. We then propose ways of extending the metric to provide additional information without adding unduly to the logging overhead.

Finding the Source of a Problem

While it can be argued that this metric aids the system administrator in locating machines in a large network that may be suffering from performance problems, the graphs represent information at too high a level to even give a feel for what component of the system might be at fault. Tracking down a performance problem on a single machine is discussed in the literature [1, 3]. Automating the testing that would be performed on a sick system could be incorporated into the metric. However, the inclusion of the results of this testing into the logs would result in more data than could reasonably be stored without filling up the filesystem. On the other hand, it is possible to set thresholds in the metric, so that if the time elapsed from the start of a run to the end (or any, clearly defined, intermediate point) is greater than the threshold, the additional information is logged. This form of additional logging could be a useful extension to the metric, because it would give some context for the spikes on the graph.

We could also consider the relationships between this metric and other tools that monitor the performance of a particular aspect of a machine or network, such as NFS [7, 8, 2]. We would like to consider how the results of these tools could be correlated with those of the metric to demonstrate what influence that aspect is having on the overall user-perceived performance. We feel that this would be a useful and interesting extension, but it has not been implemented as yet.

Real-time Detection of Problems

An extension that was proposed within Synopsys was that notification of a problem could be sent to the administrator of that machine by email or pager, in real-time. This notification would allow the administrators to monitor the state of the machines as the problems were occurring. In addition, it was proposed that notification could be sent to the administrators when the script failed to complete for one reason or another. The most common cause of failure is lack of disk space on the fileserver that a given machine is using, or the server going down. I experimented briefly with automatic notification, and discovered that with even a fraction of the machines that are in the current operational run, the amount of notification is so large as to be overwhelming. I am of the opinion that this is not a useful extension, as it stands.

It could possibly be made useful, however, through employing `syslog` or by linking the metric in to a real-time monitoring system such as Netlabs. We have not experimented in this direction, but either of these approaches would conquer the flooding effect.

Using Alternate Sequences

It was mentioned in ''Design Issues'' section that the user action that we model is the edit, compile, run, cycle. It has also been suggested that we may want to monitor file transfers over the various WAN connections, or perhaps the amount of time it takes to perform a particular sequence of queries on one of our databases.

The metric is written in such a way that, providing you are comfortable with `perl 5` and `chat2.pl`, it is trivial to drop in any sequence of transactions that could be performed by a user at a standard shell prompt. It does not support timing X applications, however.

Conclusions

The metric produces some useful ways of visually demonstrating normal and abnormal behavior of machines in a network to management and the user community. It gives a high-level, generalized overview of how the performance of the machines, or ''network'' in a particular business unit is performing. The approach taken is similar to the civil engineering approach of taking sample trips to measure delay on particular routes, or in a particular network of roads.

The metric does not, however, produce data that in itself is useful to system administrators in finding trouble spots in a large network of machines. It could be extended to be somewhat more useful in this regard, but it would take a considerable amount of work. The amount of time that would be spent extending the metric in this way must be balanced against the usefulness of the results and the intentions behind running the metric. In our case, we decided that it was not worth the effort since the metric was not intended to produce information that would be directly acted upon by the system administration staff, but rather to provide a general overview of the performance of the network as a whole.

Acknowledgments

The encouragement and advice that I received from Aoife and Paul were instrumental in the production of this paper, and for that I am profoundly grateful. I owe thanks, and perhaps a few drinks, pizza, or something interesting to read, to my proofreaders, Aoife, Jeff and Becky. My thanks, also, to all the unwitting participants of my experiments – the user community of Synopsys, Inc. – for their patience in putting up with the additional load on their machines. And equally, to Eric who believed in the metric, and to Randy whose fault it was in the first place. Finally, not forgetting Arnold, who not only proof-read the paper and avoided implementing the metric, but also takes some of the blame for inventing it in the first place.

Author Information

Christine Hogan is the security officer at Synopsys, Inc., in Mountain View, California. She holds a B.A. in Mathematics and an M.Sc. in Computer Science, in the area of Distributed Systems, from Trinity College Dublin, Ireland. She has worked as a system administrator for six years, primarily in Ireland and Italy. She can be reached via electronic mail as chogan@maths.tcd.ie .

References

[1] Mike Loukides. *System Performance Tuning*. Nutshell. O'Reilly and Associates, Inc., 1990.

[2] Gary L. Schaps and Peter Bishop. A Practical Approach to NFS Response Time Monitoring. In *Proceedings of the 7th System Administration Conference (LISA VII)*. USENIX, November 1993.

[3] Marc Staveley. Performance Monitoring and Tuning. In *Invited Talks Track of the 8th System Administration Conference (LISA VIII)*. USENIX, September 1994.

[4] Arnold de Leon. From Thinnet to 10baseT, from Sys Admin to Network Manager. In *Proceedings of the 9th System Administration Conference (LISA IX)*. USENIX, September 1995.

[5] Daniel J. Bernstein. *pty documentation and man pages*, 1992. Available by anonymous ftp from mojo.eng.umd.edu in /pub/misc/pty-4.0.tar.gz.

[6] Randal L. Schwartz. The chat2.pl package. Posted to comp.sources.unix, June 1991.

[7] Matt Blaze. NFS Tracing by Passive Network Monitoring. In *Proceedings of the USENIX Winter Conference*. USENIX, January 1992.

[8] David A. Curry and Jeffrey C. Mogul. *nfsstat man page*, 1993.