SAGE
THE SYSTEM ADMINISTRATORS GUILD

ISBN 1-880446 08-1

7 Sellens

System and Network Administration for Higher Reliability

SAGE

7

*Short Topics in*
Systems Administration

*Edited by William LeFebvre*

# System and Network Administration for Higher Reliability

*John Sellens*

SAGE
THE SYSTEM ADMINISTRATORS GUILD

## About the Series

This is the seventh in a series of booklets that SAGE is presenting to the system administration community. They are intended to fill a void in the current information structure, presenting topics in a thorough, refereed fashion but staying small enough and flexible enough to grow with the community. Therefore, these booklets will be "living documents" that are updated as needed.

**Series Editor: William LeFebvre**

***#1: Job Descriptions for System Administrators, Second Edition***
Edited by Tina Darmohray

***#2: A Guide to Developing Computing Policy Documents***
Edited by Barbara L. Dijker

***#3: System Security: A Management Perspective***
By David Oppenheimer, David Wagner, and Michele D. Crabb
Edited by Dan Geer

***#4: Educating and Training System Administrators: A Survey***
By David Kuncicky and Bruce Alan Wynn

***#5: Hiring System Administrators***
By Gretchen Phillips

***#6: A System Administrator's Guide to Auditing***
By Geoff Halprin

## About SAGE and USENIX

SAGE, The System Administrators Guild, is a Special Technical Group within the USENIX Association dedicated to advancing the profession of systems administration.

USENIX is the Advanced Computing Systems Association.

7

*Short Topics in*
Systems Administration

*Edited by William LeFebvre*

# System and Network Administration for Higher Reliability

*John Sellens*

Published by the USENIX Association for
SAGE, the System Administrators Guild
2001

ISBN 1-880446-08-1

To purchase additional copies and for membership information, contact:

The USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA USA 94710
Email: office@sage.org
Web: http://www.sage.org

First Printing 2001



Printed in the United States of America, on 50% recycled paper, 10–15% post-consumer waste.

# Contents

# Foreword

When I think of reliability, the first terms that come to mind are RAID arrays and redundant processors. But these are only the beginning. Building a reliable installation involves much more than choosing the correct hardware. Here John Sellens takes us to every edge of the reliability problem. This compilation of sage advice gives the reader a vast collection of pointers to higher reliability. The topics include items not normally considered part of the problem, showing us just how many different things must fall into place if we are to meet our users' expectations of high reliability. This booklet is a must-read for any system administrator who aspires to improve the service he or she provides.

*William LeFebvre*
*Alpharetta, GA, April 2001*

# Preface

This booklet is based on a series of articles "On Reliability" that appeared in *;login:* between June 1997 and September 1999, which have been revised and augmented for this publication.

My hope is that this booklet will have something to offer system and network administrators of all levels of experience, from near-beginners to seasoned experts. I've tried to assume only a basic background, and have tried to keep the acronyms to a minimum and to avoid using too much unexplained jargon.

This booklet is not intended to cover every situation, include every answer, or provide detailed prescriptions. It is instead intended to be a thought provoker, a discussion starter, and a jumping-off point.

Every situation is unique—what makes sense in one organization could be a complete failure in others. Technology is changing rapidly—specific solutions often have no staying power; they become obsolete too quickly.

I hope that this booklet will give you a few things to think about and that you will be able to apply the lessons offered here, along with your own knowledge and background, to increasing the reliability of your systems and networks.

By way of a disclaimer, while I happen to have written this booklet, I would like to make it clear that that doesn't mean that I'm "Mr. Know-It-All," except in the Bullwinkle the Moose sense.

Thanks and gratitude are due Bill LeFebvre for his encouragement and gentle prodding, through more than a few missed deadlines. And to Tina Darmohray and Rob Kolstad for their valuable feedback and enthusiasm while the original articles were being written for *;login:*. Many thanks to the volunteer reviewers, Josh Simon and Rick Otten, who provided many valuable comments and suggestions, which resulted in a much improved booklet.

Thank you to all those who contribute their time and effort to open source and freely available software—my working life is made much easier and far more enjoyable by the existence of excellent software like the FreeBSD operating system, Thomas Esser's TeX distribution, and countless other indispensible tools.

Thanks also to the editors, proofreaders, typesetters, and the USENIX staff. The quality of the finished product is a result of all these contributors, named or not; any remaining faults are mine alone.

*John Sellens*
*April 2001*

# I *In the Beginning*

## 1 What Is Reliability?

Historically, the UNIX operating system has been for the computer cognoscenti and has seen a lot of use in research, operating system and application development, engineering, etc. In recent years, UNIX systems have become an integral part of many organizations' business operations—many UNIX systems are now "mission critical" production systems. As a result, the need for reliable and repairable UNIX systems has never been greater.

Much of what is discussed in this booklet is UNIX-centric, but the principles and goals apply to systems running any operating system. So while many of the examples assume a UNIX environment, remember that the lessons apply to other systems, such as Windows NT, PC desktops, and anything else you might have in your environment.

### 1.1 Definition

If we're going to have any chance of ending up with reliable systems, we must first define what we mean by reliability. The most obvious definition is that:

> A "reliable" system is one that's always working properly when I want to use it.

Instead of that simpleminded definition, let's try this one:

> A "reliable" system is one that is configured and maintained to provide the appropriate levels of Redundancy, Repeatability, and Repairability, in order to provide an appropriate level of support for the services that the system is intended to provide.

(The Key Principles of Redundancy, Repeatability, and Repairability are defined below in section 1.2, and the difference between services, servers, and systems is covered in section 2.1.)

Now, that's not a very exact definition—it contains two "appropriate"s and one "intended"; it doesn't seem to be very rigorous. But the point is that systems administration and configuration, like many other things, is a constant series of trade-offs and compromises, and that each decision has to be made based on the needs of a particular organization for a particular service. The most obvious trade-off you can make is between reliability and money—you can typically get "reasonable" reliability at a "reasonable" cost (uh oh, more weasel words), but to get high reliability you often have to pay a high price. But there are lots of other trade-offs made, including those between security and accessibility, performance and redundancy, and between providing 100% uptime during the day and not having to work every evening and every weekend.

Two more introductory points. This booklet is primarily about "higher reliability" rather than "high reliability." The emphasis is on the incremental steps that can be

taken to improve reliability in your systems, rather than on building 100% uptime, absolutely no-failure, no-compromise systems. And second, if you need a quick one-word summary to provide to your boss, use this one: RAS—reliability, availability, serviceability.[1]

Do you need higher reliability? As always, it's a matter of weighing the costs and benefits for your particular situation. For example, your home workstation may not need higher reliability, unless you sell Web space on it or provide other services, in which case you should consider the effects of system downtime on your customers and your business. In most organizations it will be the server systems that may benefit from higher reliability—client machines (e.g., desktop workstations) are less likely to justify the increased cost of higher reliability (not always though—consider the case of workstations used by financial market traders).

You will need to examine the services required or provided by your organization, and the nature of the organization itself, in order to identify where you may need to consider increased reliability. For example, what happens if:

- You have a disk failure on your file server?
- The network interface on your DNS server stops working?
- You have a power failure in your building?

Sometimes a failure in the simplest services can have significant effects on your operations—you will need to consider how your servers and services interact.[2] The nature of your organization will often dictate how reliable your systems must be. For example, if you're a network or computing service provider, you may wish to have very reliable systems, in order to provide excellent service and retain your customers in a very competitive market.

## 1.2 Key Principles and Practices

I'm going to talk about reliability primarily in terms of three Key Principles:

**Redundancy**

The replication of systems or system components to allow for continued operation in the event of a failure.

**Repeatability**

The ability to reliably and consistently repeat installation, configuration, and maintenance tasks.

**Repairability**

The ability to quickly and effectively identify and repair or replace failed components or systems.

1. RAS is often thought of in terms of service metrics for measuring achievement. More on metrics in section 1.4.

2. The distinction between servers and services is often a useful one to make. Servers are typically hardware (computers, network equipment, etc.), and services are what the hardware is used to provide, e.g., mail, DNS, web, file, print and other services. See sections 2.1 and 3.2 for more information.

Redundancy and Repeatability are proactive—they help prepare for and prevent failures. Repairability, once prepared for, is reactive—it helps recovery from a failure that has already occurred.

In the presence of reasonable quality of execution, reasonable subject matter knowledge, and good efforts and intentions, reliable systems are achieved by satisfying the three Key Principles, through the four Practices:

**Planning**

The appropriate identification of tasks, processes, and materials in advance of starting work, including the identification of how to "back out" changes if necessary or appropriate.

**Preparation**

Ensuring that all necessary materials are available, appropriate training has been completed and/or knowledge is available, and appropriate notice has been given before work begins.

**Paperwork (record keeping)**

Appropriate documentation of all planning and execution, including such things as part and serial numbers, configuration parameters, locations, connections, etc.

**Performance**

Correctly executing according to plan and correctly following established procedures and guidelines, in a professional, efficient, and effective manner.

## 1.3 Goals

What are some of the goals that you might set when considering increasing the reliability of your systems?

**Service Quality**

Do you want to provide better service?[3] excellent service? Are you required to meet service availability guarantees or service level agreements,[4] either as a contractual obligation or as one of your department's goals?

**Mission-Critical Services**

Are your systems absolutely required in order for your organization to "do business"? If your database stops, does all other activity stop too? Do you have a requirement to provide user transparent "failover" in the event of a failure?

**Personal Satisfaction**

Do you want to be able to boast of the longest "uptime" on the net?

**Quality of Life**

Or do you simply want to avoid those late night phone calls from the office and the long drive into work in the dark?

3. Note that this is not the "Quality of Service" often talked about in relation to prioritizing network traffic.

4. Service level agreements are discussed in section 10.1.1.2.

And, of course, there are many other potential goals that you might set for your systems.

The key point in goal setting is one of appropriateness for your organization: are the systems goals reasonable when evaluated in the context of the organization as a whole? As with any other goals, you must consider these systems goals in terms of their related costs and benefits—it may be hard to justify spending thousands of dollars in order to save you from one late night phone call a year. And as with any "business" decision, you must consider your goals within the "big picture" of your organization.

## 1.4 Metrics for Reliability

Once you've set your goals, it's often useful to be able to determine whether or not you've met them—that's where metrics come in. The most common metric for computer systems is the percentage of time that the system is up and available. This can be modified to take such things as planned outages (e.g., for upgrades) and off-hours availability into account—a failure at 1 p.m. is often more disruptive than a failure at 1 a.m. (but then again, it's hard to give you credit for excellent system administration skills just because a disk happened to fail at night instead of during the day).

Just about any other measurable aspect of your systems' performance can be used as a metric for evaluation. Some examples are:

- network throughput
- Web hits per hour
- email messages handled
- phone calls to the help desk

But be careful—some of these possible metrics are a mixture of reliability and simply how well you sized your systems in the first place.

There are also potential metrics that are a lot harder to measure, such as:

- user or customer satisfaction
- disaster preparedness
- speed of recovery (if you never have a failure, how can you measure how fast you could recover from one?)

Choose the metrics that are right for your situation.

## 1.5 General Guidelines and Methods

To conclude this chapter, I would like to leave you with some general guidelines and methods for higher reliability system administration.

- Limit the number of different vendors and/or products you use. This can help ensure that you have the maximum possible experience on your equipment, help limit the number of spares you need, and help reduce the possibility of multiple vendor "finger pointing" when you have problems.

- Avoid putting all your eggs in one basket. A single monolithic server can seem very attractive, right up until the time there's a single, simple failure, and absolutely everything stops working. Redundant single-purpose servers are often a good idea, since they help limit the effects of a single failure, help prevent or reduce resource conflicts between different services, and usually make troubleshooting less complicated.
- Make your devices field replaceable units (FRUs) whenever possible. For example, try to avoid user-specific configuration of workstations or PCs, use network boot and configuration services (such as BOOTP or DHCP) when possible, and consider the use of X terminals, network computers, or just plain old dumb terminals when appropriate. Don't choose your hardware for the wrong reasons; choose based on sound technical, financial, and other quantitative measures rather than on the current fashion or the sex appeal of a particular device.
- Use your system logs and monitoring. Appropriate logging and proactive monitoring of the log files can often help you detect, diagnose, and fix problems (almost) before anyone else notices. An easy example of this is disk-space monitoring; if you pay attention, you can often keep file systems from filling up, fixing potential problems before they turn into real ones. See chapter 6 for more on monitoring.
- Plan for failure and recovery. No matter how careful you are, something is going to break sooner or later. If you're properly prepared, with recovery documentation and spare parts at hand, you stand a better chance of getting things fixed quickly and getting them fixed right the first time.
- Above all, remember the four Practices identified above: planning, preparation, paperwork, and performance.

# II *Hardware and Networks*

## 2 Computing Hardware

In chapter 1, I provided an introduction to "reliability," with a simple definition, a discussion of who might need reliability, goals, metrics, and some basic principles of reliability. Now let's look at the kinds of issues that should be considered when configuring computing-system hardware for higher reliability. Remember, we're not talking about 100% bulletproof systems, just the kinds of things that can be done to help you sleep better at night.

### 2.1 Services, Servers, and Systems

It is important to make a distinction between "services," "servers," and "systems." A service is a function or process, such as a file service, email transport, email post office, database access, or Web site. A server is a piece of computing hardware, or some other device, that can be used to provide one or more services. A system is a collection of cooperating or integrated servers, usually providing a more reliable or scalable service than that which could be provided on a single server.

> Remember that it's the services, and not the systems or servers, that are the most important. Most people don't care about computers or servers, they care about the services the systems provide.

This chapter focuses on the component parts that make up a server, that can be combined with other servers and components into a system, that can be used to provide a service. The reliability of the individual components, and the way they are combined into a running system, are what determines the overall reliability of a service.

### 2.2 Set Your Goals

As always, before you can design a system, you have to have an idea of the goals that you are trying to accomplish. The kinds of issues that you should consider when setting your goals include the following:

- What sort of service are you hoping to provide? Is your system going to be required primarily from 9 to 5, Monday to Friday? Or do you need to provide 24x7 uptime? What are the likely effects of unexpected (or expected) downtime? How many people will be relying on this system? Will they be using it constantly, or periodically during the day? If the system goes down, are you suddenly faced with 200 users with absolutely nothing else to do?
- How long an interruption would be "acceptable"? Is half an hour OK? Five minutes? Or is even one minute of downtime a big problem?

- Do you need to protect against every possible kind of failure, or is it acceptable to protect against just the most common? Remember that you can protect against some of the most common failures for a "moderate" amount of money, but protecting against "all possible failures" can be a very expensive proposition.

## 2.3 Consider Your Metrics

As mentioned in section 1.4, if you've got goals, you should be able to measure your progress towards those goals. Consider your metrics—what is important to you, and where do you want to concentrate your efforts? Is speed your most important goal? How do you measure speed?

- disk to disk file copy?
- network traffic?
- Web pages served per minute?
- database lookups?
- login or authentication time?

What is important to you? How will you measure it?

Before you design your implementation, know what you want so that you'll end up with a system that can provide it. And if you've got metrics in mind, it will make any benchmarking or pre-purchase testing you do much easier.

## 2.4 Basic Design

Once you've established your service goals and have considered some of the metrics you could use to measure your achievement of your goals, you will need to consider what kinds of mechanisms you're willing to put up with in order to reach your goals. If the only way to reach your goals is through a hugely complicated, very expensive collection of hardware and software, you may find yourself wishing to scale down your goals for the sake of your sanity (and your budget). There are some relatively simple mechanisms for reliability (the use of some form of RAID to provide disk space, for example), and some very complicated ones, involving "hot spare" machines, fast failover software, transaction monitors, and so on. Let's start with a review of some of the principles and guidelines of reliability, and then cover the areas most relevant to hardware in turn.

The most obvious principle for computing hardware is that of redundancy. This covers everything from RAIDed disks, to machines with multiple CPUs or power supplies that can handle failures "gracefully," to your storeroom of replacement parts and machines. Limiting the number of vendors and configurations you deal with is another useful principle for hardware, since it reduces the number of spares you need to keep on hand, and means that you're likely to have a greater familiarity with the different components of your systems. And finally, remember that FRUs are a good thing, especially for user workstations.

## 2.5 Physical Environment

Before you begin designing and specifying your computing hardware, you need to consider the environment in which they will be installed, and how that affects reliability.

Will your machines be in a closet, a warehouse, an in-house computer room, a state-of-the-art colocation data center? Where will they be located—shelves, equipment racks, locked cabinets, the floor under someone's desk? Are there dust or moisture problems you will need to worry about? Vibration from heavy machinery?

Are the heating and cooling systems appropriate? What happens when they break down or malfunction? Is there always someone onsite or environmental monitoring in place so that you'll know if the air-conditioning fails and the temperature starts rising?

Are the machines easily accessible for maintenance and repair, or are they in a remote location or somewhere else that makes physical access difficult?

## 2.6 Power

One of the most obvious, though often overlooked, potential failures is that of your building power supply. A UPS (uninterruptible power supply) that does power conditioning and provides protection from power spikes provides greatly increased reliability at a modest price. A UPS can protect you against momentary municipal power failures, circuit breakers tripping due to overloaded circuits, minor machine room reorganizations, people knocking power cables out of the wall, and the proverbial building maintenance person with a large vacuum cleaner.

A UPS can also protect your equipment from damage due to power problems, and save you from long reboot processing or file system damage after a short power outage. Modern non-trivial UPSs will often allow you to replace the batteries and electronics while still running your equipment off the regular power feed, which helps your UPS avoid being the cause of planned or unplanned downtime. Make sure you size your UPS to provide the appropriate amount of standby power, and obtain and use the appropriate software to shut your systems down cleanly in case of an extended power outage. If your needs and goals can justify it, you may also consider the use of standby diesel generators, which can keep you going almost indefinitely in case of an extended power outage (consider the needs of hospitals during disasters, telephone companies, or emergency broadcast systems).

And while you're considering power, don't forget to include your environmental systems in your capacity planning and configuration—a generator for your computers isn't very useful if you'll soon have to shut down because the air-conditioning or sump pumps aren't working.

## 2.7 Disks and Storage

The next easiest and perhaps most important type of failure to guard against is disk-drive failure. Modern disks are very reliable, but the larger your installation, the more likely you are to have a disk failure. Large database servers often use RAID technology to provide redundancy (and sometimes improved performance) to protect the data—

when your only record of your business transactions is on your disk, it's often worth the extra cost for the added protection. Consider, for example, the situation seen more and more often these days—a Web site taking orders for consumer merchandise for delivery. Without adequate protection, a disk failure could lose orders, and customers, forever.

## 2.8 RAID Basics

RAID is an acronym for Redundant Array of Inexpensive Disks, originally defined by Patterson, Gibson, and Katz [*Patterson87*] at the University of California, Berkeley, as a way to provide increased reliability in disk subsystems. RAID configurations are usually described in terms of RAID "levels":

- Level 0 is data striping, where consecutive logical disk blocks are interleaved across two or more physical disks, providing increased performance over a single disk of comparable size (level 0 is not one of the original redundant methods).
- Level 1 is data mirroring, where two (or more) copies of the data are kept on separate physical disks, which provides data redundancy and increased read performance.
- Level 5 is parity striping, where data is written across multiple disks, with additional parity bits, which will ensure that the data is still available (and correct) in the case of a single disk failure; this level provides data reliability and increased read performance. Level 5 RAID requires at least three disks—five disks is typical.

There are other RAID levels, but these three are the most common.

RAID levels can be combined—the most common combination being 0+1, where data is striped across multiple disks, and the combined logical disk is itself mirrored on another set of disks. The different RAID levels provide different combinations of performance, security, and cost—level 0 provides no data redundancy (data is lost in the event of a single disk failure), level 1 provides good data redundancy, but at the expense of doubling your disk storage requirements, and RAID 5 provides data redundancy for a lower cost, albeit with somewhat lower performance.

One nice aspect of data mirroring is that the mirrored disks can be in separate disk enclosures, providing protection against power supply, interface, or cable failures. Some disk subsystems can even be located non-trivial distances from their host computers (e.g., up to 2km, via optical fiber connections), which can allow mirrored disks to be placed in separate buildings, providing excellent data reliability and disaster recovery through instant offsite backups, with no performance penalty.[1] Some file server and database products support data replication across the network, providing good protection, at somewhat lower levels of performance.

1. Just to make sure that you don't think I'm just making that scenario up, we actually implemented a system exactly like that in 1996 at the University of Waterloo.

## 2.9 RAID Implementation

RAID systems can be implemented through operating system software, add-on software products, or through special purpose "hardware" RAID implementations. Most current operating systems offer some form of "software" RAID, providing various RAID configurations at the cost of (typically) a small system performance penalty.

The alternative to software RAID is hardware RAID through the use of stand-alone disk subsystems, which provide the ability to join multiple disks together in various RAID configurations, and which appear to the host OS simply as larger than normal disks or partitions. Hardware RAID subsystems avoid the operating system overhead of managing a RAID implementation.

Different situations call for the use of different approaches to RAID—I'll claim that the software implementations are often more flexible (in that the disks are just regular OS disks, rather than disks dedicated to hardware RAID), but the hardware implementations are often easier to manage, and are especially useful when your OS doesn't provide the best software RAID support.

One last aspect of a good RAID system is the ability to make use of "hot spare" disks, where otherwise unused disks are configured into a RAID setup and are used to automatically replace any disk that fails. This provides higher reliability, quicker recovery, and simpler support, at the cost of an additional disk or two. And I should mention that external disks are much easier to deal with than internal disks when something goes wrong. Internal disks are convenient in some ways, but it's usually much more complicated to find a spare internal disk (with the right mounting hardware!) and replace the failed one than it is to simply grab another external disk subsystem and plug it in.

## 2.10 Other Disk Notes

One final approach to disk reliability is through the use of dedicated network file servers, such as those sold by Network Appliance, Auspex, and others. These are special-purpose machines, running customized code to provide very fast, very reliable network file service. The server software is designed to be very reliable, and these file servers typically use RAID internally to guard against disk failures. As mentioned above, some network file servers support data replication across a backbone or wide-area network. These aren't appropriate for all applications, however—it may not be a good idea to run your production database over NFS.

One final note about disks. All the mirroring in the world won't protect you against OS or application software failure that trashes a disk partition or directory. See chapter 7 for a discussion of backups and restores.

## 2.11 Other Internal Hardware

The non-disk portions of computer systems are usually somewhat more reliable than disks are, but they can still fail in annoying and unpredictable ways. I will mention two aspects of increased reliability in this area—increased fault tolerance and easier recovery from failures. These two areas tend to be "internal" and "external" aspects, respectively.

Modern computer systems can often deal with certain kinds of internal hardware failures in reasonable ways. Systems with multiple CPUs can sometimes just stop using a particular CPU if one fails—either while still running or by ignoring the failed component after a crash and reboot; memory failures can sometimes be handled the same way. In these cases, your system will still run, albeit at lower levels of performance, until you can arrange to replace the failed parts.

Failures of other components, such as network, terminal, or disk interfaces, will often just cause the component to be ignored when the machine reboots, but this can cause important things to stop working. If you only have one network interface on your network file server, and the interface breaks, you may find it somewhat inconvenient or unsettling when people from all over your organization start phoning and visiting, with varying degrees of cordiality. It's usually possible to configure multiple network interfaces into a computer, so, if one fails, you can continue to communicate over the other (redundant) interface.

Disk subsystems are sometimes "dual-ported," so you can have two physical connections to your disks. Be careful not to try to use both interfaces to talk to the same disks at the same time—that would usually be an invitation to disaster. Note that if your disks are mirrored and the mirrors are on different disk controllers, you may be thought of as a hero when one of your disk interfaces dies unexpectedly.

If you have serial connections to your machine, for modems or dumb terminals, you can either configure in more ports than you will ever need (so when one set fails, you just move all the serial connections over to your spare ports), or you can take a different approach and use network terminal servers instead, thereby shifting the blame elsewhere when something goes wrong. Terminal servers are of course also subject to failures—just make sure you have a spare or two around.

The other primary internal components are things like the power supplies, console terminals and interfaces, floppies, CD-ROMs, and fans. Some machines will let you configure in multiple, redundant power supplies, so that if one blows up, the machine will keep on running, using the other power supply. For server machines, it is often possible to run them without a console (though a console is often convenient to have)—if your console terminal, keyboard, or display fails, just unplug it and see what happens (you may need to reboot in some cases). For things like floppy disk drives, CD-ROMs, and those little tiny fans (blowing on things like CPUs and internal disk drives), you pretty much have to either live with the failure, replace the failed parts, or open the cover and point a nice big room fan at your computer.

## 2.12 Clustering and Redundancy

Depending on your requirements, operating system, and hardware, it may be possible to "cluster" multiple systems to provide higher reliability and more resilient services. A cluster is two or more machines, acting as one, configured so that a failure in one of the machines in the cluster does not interrupt the services provided by the cluster. There are a number of different implementations of clustering, some more and some less elegant and effective, depending on your software and hardware.

Redundancy for services across multiple servers can be implemented in a number of different ways, and often depends on which services are involved (see section 3.2). External load balancers, often used for larger Web sites, may provide some of the benefits of clustering without many of the complications.

## 2.13 External Considerations

The external aspects of hardware failure recovery primarily involve spare parts, service contracts, and courier services.

Depending on the size of your site, and how critical particular computer systems are to your organization's activities, it may be worthwhile to invest in a spare parts inventory, especially for those parts that are most likely to fail. (You should always have a spare disk or two around—even if nothing goes wrong, your disks are soon going to fill up anyway, so a spare disk can help you recover from a different kind of failure.)

Choose your service vendors carefully, and make sure that you have the service contract that is appropriate for your particular systems and needs. You may want to have a 24x7, two-hour onsite response contract for your central database server, but you may not even want a service contract on your desktop PCs (unless the desktop in question belongs to your company's CEO).

Be prepared—take training courses offered by your vendor, keep your spare parts near where they might be needed (not across town on another campus), know how your systems go together and come apart, and make sure that everything is neat, tidy, and well labeled.

Consider limiting the number of vendors you deal with. This has several nice results—you'll need to keep fewer spare parts on hand, you'll be more familiar with your equipment and better able to respond when something goes wrong, and the more you buy from a particular vendor, the better service and discounts you are likely to get!

And, especially when shopping for commodity systems (i.e., Intel-based PCs), consider whether the cost savings from a "no-name" vendor outweighs the potential costs and difficulties in dealing with unknown, one-of-a-kind, or less reliable parts. With no-name systems, your warranty may only be good as long as the particular vendor is still around and in business.

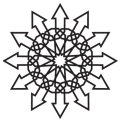
## 2.14 Be Prepared

Above all, don't forget that, sooner or later, anything can (and will) fail. Most of us have probably seen disk failures, or power supplies gone bad, or flaky memory chips. But even the simplest and most reliable components can fail (like serial or ethernet cables). I've even seen failures in small external AUI/10baseT ethernet converters—there aren't many system components simpler (or cheaper!) than those.

You may have noticed that I only briefly mentioned desktop, user machines. The answer to reliability for these kinds of machines is simple: just keep spares on hand, and make sure everyone stores their files on the central, backed up, file server and not on their desktop. When something goes wrong, just replace it with a spare, and your

user will be up and running again in no time. (You'll notice that I've ignored situations like financial trading or medical applications here.) And again, your task will be simpler if you have fewer vendors and fewer different types of desktop machines to deal with.

That (hopefully) covers the basics of higher reliability for typical computer systems. More demanding environments will demand more extreme approaches to reliable systems, but if you're in that kind of situation, you may know more about this topic than I do.



# 3 Networks and Network Services

This chapter is concerned with networks and network services planning and implementation. This includes such things as routers, switches, cabling, leased lines, and servers for such things as mail, DNS, and file service.

> Advance planning is very important for networks—it is almost always incredibly painful and expensive to rework your network topology if you have anything other than the most trivial of networks.

Many of the basics for network hardware are the same as those for computing hardware, as covered in chapter 2. In this chapter, I'll try to point out a few places that tend to be "single points of failure" and suggest ways to deal with them.

## 3.1 Network Topology and Components

Let's talk a little bit about network topology—the physical layout of the cables, routers, hubs, switches, and other devices that make up the "skeleton" of your network,[1] and the network hardware components themselves.

### *3.1.1 Local or Small Networks*

If you're a small organization, and everyone fits on one floor of a typical office building, your layout is likely going to be pretty straightforward. All the routing and repeating hardware will end up in your single wiring closet (which, for this size of network, may actually be a closet), with a single upstream connection to your Internet service provider (ISP). While you don't have a lot of reliability choices in this instance, there are a number of things that you can do to help you recover quickly when a failure happens, and many of these tactics can be applied to local wiring hubs in much larger networks.

#### 3.1.1.1 Office Wiring Drops

I have three suggestions for your office drops:

- Use quality components, i.e., category 5 or better wiring, terminations, punch downs, etc.
- Install them carefully and within specifications (e.g., length limits).
- Install spares, since some wires and connectors will eventually fail, and it's so much nicer when you can just switch someone to the next port to get them working again.

1. If the main part of a network is the "backbone," I figure that it's fair to call the whole shebang the "skeleton."

You should also be careful where cables are run in the offices or cubicles—it's not just a safety thing; you don't want people walking on or tripping over your nice, new cables either. You should also hire an outside contractor to do your wiring instead of trying to do it yourself—it will probably get done faster, better, and you'll have a written guarantee and someone else to point the finger at when things don't work.

#### 3.1.1.2 Local Hubs, Switches, and Repeaters

Whether you choose fancy-schmancy smart hubs and network switches with all sorts of whiz-bang remote management features, or the dumbest, plainest hubs you can find to minimize the number of things that can go wrong, try to standardize on one model or brand, maintain a good relationship with your vendor, and keep a spare or two on hand.

Make sure you have spare network ports available in case a single port or group of ports goes bad, and if your main file/mail/Web/Doom server is the only fast ethernet device you have, make sure that you have a spare fast ethernet port to use if the one in use goes bad. (If your organization is like most, you'll need extra network ports eventually anyway—just remember to buy more hubs or switches when you start running out of ports!)

Cascading hubs, where a number of slave modules are daisy-chained off a single master, can be attractive if you need managed hubs, but make sure that you're not left completely dead in the water when the master module fails. My personal preference is for stand-alone units—I'm terribly afraid of cascading failures.

#### 3.1.1.3 Local Routers

If you're a small office, you may have only a single network and your only router may be for your (single) connection to the outside world via your ISP. Routers tend to be relatively expensive, and so it may not be financially practical to keep a spare on hand. If you have only one or a few routers, see if you can strike a deal with your ISP or router vendor (and you do have only one router vendor, don't you?) for quick replacement in case of failure, or, at the very least, purchase a maintenance contract which guarantees you overnight (or faster) delivery of a replacement. If not, be prepared to be cut off from the rest of the world at some point.

#### 3.1.1.4 Miscellaneous

One final point about local network design: you should consider the security and safety of your wiring closet. Ideally, you would like one that is 100% secure from prying eyes and fingers, is air-conditioned and rodent-free, has uninterruptible power, and is not located in a hurricane- or tornado-prone area, on a flood plain, or underneath a kitchen or washroom. Unless you're very lucky, you'll probably have to settle for something less than the ideal.

### *3.1.2 WAN/Backbone/Upstream*

The "upstream" or "backbone" portion of your network is where you will probably be more concerned about higher levels of reliability. It's one thing for a work-group LAN to go down and isolate 15 or 20 people, but it's a different matter entirely when your global corporate backbone melts down and thousands of employees are left idle.

This is where you should consider redundant paths and cables, uninterruptible power and good environmental controls, high reliability hardware, and very careful configuration and monitoring.

#### 3.1.2.1 Redundant Paths

One of the most obvious reliability approaches in network topology is the use of redundant communication paths to guard against natural or back-hoe related cable failures. This is often more important when your organization is spread across multiple buildings, cities, or continents than when you are within a single building—cable or fiber failures within a building are usually easier to find and deal with than a failure on a leased fiber somewhere between Chicago and New York.

If you build some sort of "looping" into your network (by, for example, connecting your three different buildings so that each building has a direct connection to each of the others), you can live with the failure of any one wide-area link, albeit at a reduced total bandwidth.

When possible, you should consider the use of multiple access points to your building and the use of different wide-area communication carriers to reduce the risk of a single accident (fire, back-hoe, disgruntled telco employee) taking out both of your redundant links. For example, a power outage at a communication carrier can be a big problem for your network if your only connection goes through that switching office.

Networks with multiple routes require increased routing complexity—you won't be able to rely solely on default or static routing in a redundant network. You'll need to make use of a more sophisticated routing protocol, often something like OSPF, BGP, or HSRP. If you don't have the necessary routing expertise in-house, your ISP may be able to provide the required skills more cost-effectively than a third-party consultant.

As an alternative to private leased lines between your offices, you should consider the use of the Internet for wide-area communication links, using virtual private networks. VPNs can be implemented using software or hardware encryption to provide secure communication over public paths. An advantage of using a VPN is that you can use the multiple redundant links of your ISP (and the rest of the Internet) to provide a more reliable communications path. This, of course, can also be a disadvantage, as you're depending on someone else to provide appropriately reliable service for your network.

#### 3.1.2.2 WAN Hardware

The routing and switching hardware that you use on your WAN is also a very important part of your overall network reliability. One unfortunate thing is that, as your network grows larger, it often makes the most sense (from a bandwidth and management point of view) to use larger routers rather than collections of smaller routers. This means that the cost per router goes up, as does the cost of keeping a spare available. But fortunately, the larger routers tend to be more reliable thanks to such things as redundant power supplies (make sure they're on different power circuits!) and multiple, independent interfaces. In practice, you're more likely to have a cabling or WAN circuit problem than a router problem (or at least than a router problem that can't be fixed with a more current software release or a reboot).

### *3.1.3 Be Prepared*

As always, label, document, and be ready for problems:

- Make sure that each cable is labeled (with proper labels or tags that won't fall off).
- Make maps of floor plans and network drops, map your network links, and keep your vendor support numbers handy, with a list of part numbers, options, and wide-area communication circuit numbers.
- Consider collecting and documenting every MAC address on your network—having this data available can make troubleshooting much easier, as it will be much easier to determine the physical location of a misbehaving device.
- Make sure you have more than one copy of your documentation, including a paper copy, in different locations, so that you won't be unable to reach the copy you need to recover from a fire or natural (or network) disaster. This of course includes the configurations for your routers—don't keep your only copy in the router's memory!

Remember—plan ahead to limit problems in the first place, and make it easy to recover when you've had a failure.

## 3.2 Network Servers

Once you have your physical network in place, you may actually want to put it to use by connecting some machines. I'll claim that, other than the networking infrastructure, you can pretty much split network-connected machines into clients and servers, though it is never actually that clear-cut in practice.

I'll define a "client" as a machine that no other machines or services rely on. A client machine is one that can disappear off the network and the only people inconvenienced will be those who want to sign on to that machine directly—personal workstations are the canonical example. I'm going to ignore client systems in this chapter—see chapter 2 for a discussion of computing hardware reliability.

It's worthwhile to differentiate between "servers" and "services"—the server is the hardware, the service is the protocol or information that the server provides or records. Increased reliability is easier if your service can be decoupled from your server—if the service in question does not require special purpose hardware and replicates easily, you're in luck. Examples of services that decouple and replicate well are DNS name service, DHCP and BOOTP service, and (most) Web servers; services that don't typically fare as well are file and database servers and mail servers, since you usually want to have one "authoritative" server for these purposes.

For those services that don't decouple or replicate well, you have two basic approaches to reliability: make the server machine itself as reliable as possible (again, see chapter 2), and/or make it as easy as possible to move the service to a different machine in the event of a failure. The latter approach is more or less practical depending on the service and the size of the data and client population. But planning, record keeping, and preparation will make service movement much easier (am I starting to sound like a broken record here?).

One technique I mentioned briefly in chapter 2 that is even more useful when you have multiple networks is the use of multiple network interfaces on your servers. For example, if you have a file server with a separate network interface for each network that it serves, router failures won't interrupt your file service. This technique can, of course, be applied to just about any service, and will usually provide better performance as well as better reliability.

For those services that can be replicated, the obvious approach to reliability (and often performance) is to replicate. For example, DNS service is usually best provided by a primary server and multiple secondaries, geographically dispersed throughout your company—that way, a failure (power, network, or human) that isolates the part of your network containing your primary name server won't disrupt your other networks (unless the failure continues long enough that the DNS data starts to time out). Topological dispersion, both physical and logical, is a very important technique on non-trivial networks. Other services that benefit from replication and dispersion are security servers (such as Kerberos or SecurID), relatively static file servers (such as software servers for your workstations), and so on.

For other services, such as mail and USENET news, replication usually isn't appropriate (or just plain doesn't work). However, depending on the size of your organization, you should consider having multiple servers, with people assigned to servers based on location or some arbitrary differentiation such as userid. This is the "don't put all your eggs in one basket" reliability guideline—it doesn't help those people assigned to the failed server, but at least the people on other servers can keep on working.

One alternative that can be especially appropriate for Web or other external servers is to co-locate your server with your ISP or some other service provider. This means that your server is no longer limited by the bandwidth or reliability of your network link to your ISP, and you can benefit from the UPS, air-conditioning, and 24x7 services of your provider, without having to do it all yourself.

## 3.3 Software and Configuration

In addition to the network and system hardware and planning, proper configuration and software support is essential for a reliable network and services. Some useful techniques are:

**Configuration Automation**

Automate your configurations. This makes it easier to maintain and replicate your servers and services, and makes it much less likely that a finger slip will cause your servers to stop working. One of the most obvious places for automation is when configuring your routers, especially security-related routers (see [*Calabrese96*] for an example).

**Dynamic DNS**

Make use of a dynamic or load balancing nameserver (such as "lbnamed" [*Schemers95*] or "eddie" [*eddie*]) so that DNS lookups will ignore redundant servers that are down or otherwise unavailable. There are also hardware devices, primarily sold

as gateways to multiple WWW servers that serve the same URLs, that act as gateways to the private server network and which choose the fastest or currently available server.

**BOOTP and DHCP**

Configure your client machines (when possible) using protocols like BOOTP or DHCP. Properly configured, a simple config file change can cause all the client machines in your organization to choose different servers (e.g., DNS, time, gateways, etc.) the next time they boot, which makes it much easier to reconfigure things in case of a failure, or ordinary network evolution.

**DNS MX Records**

Use DNS MX records for your mail machines to cause incoming mail to collect on another one of your servers when a mail machine is unavailable. This is much more convenient than having your mail pile up on the sending system, as it allows you to adjust the expiry time, or manually redirect the mail elsewhere to accommodate the failure and provide alternative recovery methods.

**DNS CNAME Records**

And finally, use DNS CNAME (alias) records to attach service names to server machines. Doing this will make it easy to move the service to another machine, when necessary, without forcing all your users to change their habits or reconfiguring all your client machines.

## 3.4 And Finally . . .

Make sure that you have monitoring software and systems in place, so that you can detect failures as soon as (or before!) they happen—see chapter 6 for more about monitoring.

# III Software and Operations

## 4 System Administration

Once you have your hardware installed, and your network working, there are still all the day-to-day tasks of installing and upgrading software, maintaining user accounts and mailing lists, monitoring for system or network problems, and repairing all those little things that go wrong. The software- and configuration-related tasks of system administration are what I'm going to attempt to cover in this chapter. For lack of a better word or phrase, I'm going to call those tasks "system administration," but we all know that it's just one part of the total system administration work load.

So far in this booklet, we've focused on hardware and wiring, and touched on tasks and processes only superficially. Reliable "system administration" (at least as I have defined it here) tends to be much less related to hardware and much more related to the tasks themselves and the performance of those tasks.

In this chapter, I'm going to outline a few key words for reliable system administration, a few examples of how to apply them, and then mention a few topics that don't fit neatly into those categories. (This matches nicely with system administration in general, since nothing ever fits together quite as well as you hope it will.)

### 4.1 Key Words

Allow me to propose a short list of important key words for reliable system administration:

**Consistency**

Set your standards and procedures, and stick to them, and all sorts of things will be easier and more reliable.

**Documentation**

If it's not properly documented, it won't pass either the "run over by a bus" test or the "won the lottery" test—if all the documentation is in your head or in your mailbox, the system is unlikely to survive your sudden disappearance.

**Automation**

Never do something twice by hand if you can write a program to do it for you. If tasks are (properly and carefully) automated, they're less likely to fail due to distraction, being rushed, or having fumbling fingers.

**Repeatability**

This is often a combination of the previous three. Proper documentation and automation will help you set up a new machine (or recover or upgrade an existing one) far more effectively and reliably.

Let's talk about these in a little more detail with a few examples.

### *4.1.1 Consistency*

This really is one of the cornerstones of system administration. If you're not consistent in how you do things, you're going to be much less effective, much less able to delegate tasks, much less able to scale your installation past a small number of systems, and much less able to put together systems that run reliably and are easily recoverable in the event of a failure.

Let's pick on everybody's favorite topic for an example: backups. (We'll be able to use this same example later for the other key words as well.) If you have inconsistent backup methods, using different commands or schedules on all your machines, I would venture to guess that you're going to find doing backups a terrible, onerous chore and not do it as well as you would hope to. And if doing backups is a complicated and convoluted task, it will be hard for you to pawn the job off on some underworked clerk (we all know, of course, that the only truly overworked people in a company are the system administrators), giving you less free time. Different backup methods for each system mean that once you get beyond a small number of machines, you will likely find it simply too complicated to do proper backups. Lastly, if you don't have one standard way of doing backups and restores, of cycling your tapes offsite, and of keeping track of which backups are on which tapes, you're going to have a heck of a time putting things back together when your company Web site goes boom, and your boss, the head of sales, the head of marketing, and the CEO are all standing around breathing down your neck.

Have I convinced you yet?

OK, then how about another perennial favorite: software installation, configuration, and distribution. If you don't believe me that it's a favorite topic, have a look at the proceedings of any of the LISA conferences and you're bound to see at least one or two related papers each year. A simple example here is the fabulous GNU autoconf [*Autoconf*] system, which is used to generate those nice configure scripts that you see in so many software distributions. In the olden days, when you grabbed some software from the Net, you had to read the READMEs, examine the makefiles and .h's, and customize everything to suit your environment manually. Nowadays, if there's a configure script included, you can just about always rely on

```
% ./configure --prefix=/local
% make install
```

to do just what you would hope it would do. A concrete example of the time- and effort-saving benefits of consistency.

If you look back at the LISA software installation and distribution papers, one thing that they virtually all have in common is a set of rules defining how and where to install your software "packages." Decide on one place to put your source, one structure in which to install your binaries and supporting files, and follow your rules religiously. It's much easier to be able to tell someone that the source is under /local/src/pkg and the installation is under /local/dist/pkg, than to have to special case everything you do.

Consistency applies to almost everything: day-to-day procedures, how user accounts get authorized and created, where your backup tapes are stored, how IP addresses are assigned, and on and on. Whether you call them rules, policies, procedures, or practices, once you set 'em, don't forget 'em. I spent quite a few years doing centralized system administration and software support for hundreds of different machines, with different operating systems and revisions, for thousands of users, and I'm convinced that one of the most effective tools in a system administrator's arsenal is consistency. Your users will thank you, your friends and family will thank you, and your evenings and weekends will thank you too.

### *4.1.2 Documentation*

If you don't document everything, you're not doing your job. You're making work harder for the users (is there anyone who doesn't get frustrated when the "man" page and user documentation are missing and not to be found?). You're making life harder for yourself and your co-workers (since you'll end up answering the same questions over and over). Finally, if all the documentation is locked up inside your head, you're making yourself un-promotable.

There always seems to be a strong perception that "documentation is hard," or "I'm not a good writer," or "I don't have time." I'll tell you, from experience, that man pages for most programs get to be trivial to write after you've knocked off a dozen (or a gross) or so. And who hasn't had the experience of returning to a program or system after a few months, and being unable to remember or determine what your beautiful, "self-documenting" code is supposed to be doing?

As I mentioned above, the favorite metric for how much documentation is enough is the "if you got hit by a bus" rule. The question being, would your systems keep on running if you got hit by a bus tomorrow? (I prefer to use the "if you won the lottery and suddenly quit your job to move to Tahiti" rule, since that seems much more cheerful.)

Make a man page (if you're a UNIX administrator) or a Web page (for anyone else), for every command or process that you install. Document each "process" or "job step," document your cron jobs and mail aliases, document your installation standards, your policies and procedures, your vendor service contract information, and make your life easier. Document your backup and recovery practices and processes, so your minions can take care of things while you sleep.

While you're documenting, don't forget to document all the passwords needed for privileged access to your systems. Password escrow software and services do exist, but the most common practice involves passwords written in sealed envelopes, kept under lock and key in the corporate safe, with some form of defined procedure or authorization mechanism to allow access to the passwords when necessary (and appropriate). And make sure you don't forget to update the stored copies when you change passwords or add new systems.

Once you have an online copy, don't forget to make your all-important paper copy, so you won't be completely stranded when your root partition dies. Set up a cron job or something to automatically (oops, that's the next key word) print an updated copy

every few months; obsolete emergency recovery documentation isn't much fun. And finally, don't forget to write down the root or administrator password somewhere, so that your systems can survive a bus crash, even if you don't (even though it's not likely to be a significant legacy handed down through the generations).

### *4.1.3 Automation*

Automation is your friend. Get it working right once, on one machine, and it will (more than likely) keep right on working, and you can usually use the same method on all your other machines, old and new.

Let's look at backups again. No one wants to type the runbackup command every day. And it's a sure thing that no one wants to have to type nonsense like

```
% dump 3bfu 32 /dev/rmt2 / /usr /var
```

day after day after day. Get the best backup hardware and software your budget allows. If your budget is zero, at least use something like the (terrific) "Amanda" backup software from the University of Maryland [*Amanda*]. If your budget is non-zero, buy expensive backup software, a giant jukebox, install fiber to a suitable offsite location, set it and forget it (well, not really, but you get the idea). The idea being, of course, make computers do the repetitive tasks—computers are good at mindless repetition, and humans generally aren't.

We've all seen makefiles (or at least, most of us have). If you think about it, make [*Feldman78*] is one of the earliest automation tools for UNIX, and can be used for all sorts of things, in addition to building programs from source.[1] Shell scripts combined with cron or at are another easy way to automate repetitive tasks.

My final example in my argument in support of automation is its use in software distribution. One of the few recurring themes in the myriad of software distribution papers in past LISA conferences is that of automation. Updates almost always get distributed automatically, usually fired off early in the morning from a cron job. People can be far more effective when they have the same software environment everywhere, and, conversely, system administrators can be far less effective if they need to copy, compile, and install the same software on each machine any time something new or updated is installed.

### *4.1.4 Repeatability*

There are two ways to think of repeatability—how to redo or repeat tasks on a single machine, and how to apply (or repeat) the same actions across multiple machines.

Using our favorite example (backups), repeatability means that your backups will run the same way, day after day, with as little manual intervention as possible. This includes little things like making sure that your log files get rolled from time to time, so that they don't just grow without bound and fill your disk. Repeatability also means that you can install the software and get it working again after an OS upgrade (or recovery). It also means that once you have your backups working on one machine, you can easily get them working on all your other machines.

1. See, for example, [*Satdeva94*].

Making tasks repeatable usually requires a larger up-front investment of time and energy. You need to think about the process and write things like install scripts, makefiles, setup scripts, and so on. In software development, we talk about the development cost being only a small part of the total cost, and the bulk of the cost being in the ongoing support and maintenance of the software. I'll claim that system administration often has an analogous rule—that the bulk of the cost of system administration is often the ongoing and repetitive tasks that stretch forward through the years. But I'll also claim that the future costs can be reduced (and ofttimes reduced substantially) by making a modest up-front investment in ensuring repeatability.

Two more quick examples of tasks that really benefit from repeatability: account creation (especially in an educational setting, or anywhere else with high user turnover) and PC and/or workstation setup (typically through software automation, disk cloning, or custom boot disks). Very few of us look forward to creating the 10,000th user account, or setting up the 1500th "wintel" PC by hand. (Yes, repeatability is often very closely tied to automation.)

## 4.2 Odds and Sods

Now that we've gone over what I claim are the key words and basic ideas, I'll quickly mention a few more items that don't seem to fall obviously under one of the key words.

- Use BOOTP and/or DHCP servers to dole out IP addresses and distribute DNS, gateway, and other configuration information, even if you statically allocate IP addresses to all your PCs, X terminals, etc. The ease of setup, and the ability to renumber or reconfigure easily and automatically when necessary are great ways to increase reliability (and your free time) by avoiding having to visit each desktop or having to track down which two machines are trying to use the same IP address.
- Avoid manual or custom setups whenever possible. Use a standard workstation software setup, and discourage (or prevent) users from changing or adding things.
- Use tools like rdist [*Cooper92*] and track [*Nachbar86*] to automate your file distribution—they're a great way to automatically replace or repair files that have become corrupted or gone missing (as long as it's not your master host that has problems!).
- Use cfengine [*Burgess95*] to automate all those fussy little configuration tasks on your machines.
- Use a change control system, like RCS, SCCS, or CVS [*Berliner90*], whenever possible, and always put something meaningful in the change logs. It's a great way to keep track of what you've changed, an easy way to keep old versions available, and far more reliable that a series of ".bak" files.
- Use a proper change management system (section 5.3) to improve and formalize your processes. This leads to a more reliable and supportable environment.

- Set up as much automatic monitoring as possible. Use a log file watcher to monitor syslog and other log files and dispatch warning messages in the appropriate ways (email, broadcast messages, pagers, voice modems, cell phones, and so on). Or even better, have your log file watcher fire off repair scripts to fix things automatically. Periodically run commands or scripts to watch things like mail and printer queues, free disk space, load average, and so on. It's always nice to notice and repair a problem before the users notice and start calling.
- Use a cron job to save copies of important files (e.g., /etc/passwd) to help guard against errors, accidental deletions, and file system corruption.
- Training and knowledge—the more you know about the software and hardware you're working with, the more reliable and effective you will be in maintaining and enhancing your systems. Get training, read books and articles, attend conferences, participate in newsgroups and mailing lists.
- And to help you react most effectively, make sure that you have the set of tools that help you do your job in the most appropriate way. Tools like laptops, network connections at home, cell phones, and pagers can be annoying, but if they save you a late-night trip into work, you might be better off.

## 4.3 Covered Elsewhere

There are some topics that could have been included or covered further in this chapter, but which are important or large enough to merit separate discussion.

- Chapter 7 covers backups in much deeper detail.
- Disaster recovery is covered in chapter 8.
- Chapter 6 covers system and network monitoring.
- Discussions of communication and "people practices" and how they relate to reliability are in chapters 9 and 10.
- And, finally, there's a discussion of security in chapter 5.

# 5 Security and Reliability

In this chapter, I will attempt to provide an overview of how "security" contributes to the reliability of your systems and, hopefully, show how a lack of security decreases reliability. The underlying premise is that an insecure system is far more susceptible to inappropriate use and unauthorized changes, which can have a considerable impact on how reliable the system is.

Some of the topics covered in this chapter will be related to discussions in the previous chapters, but this is an attempt to gather everything related to security and reliability in one place. This should, I hope, make it easier to see the big picture.

Security is a wide-ranging and sometimes poorly defined topic. It's probably worth noting that "computer people" often (incorrectly) think of security as being only related to things that you can do with a keyboard, a computer, random bits and bytes, and someone else's password. Accordingly, I will attempt to summarize what security is, or at least what it is in the context of this chapter.

We will cover the following security-related elements:

**Access Control**
: Passwords and other mechanisms that attempt to require some level of authentication and authorization for access to your networks and systems, i.e., how to know when to open the barn door.

**Physical Security**
: Protection against physical attacks and "acts of God."

**Intrusion Detection**
: How to detect when someone unexpected has entered through an open or insufficiently closed barn door.

**Correction**
: Fixing things when they break, or get broken, which includes "re-motivating" individuals when they act inconsistently with what is expected.

**Change Management**
: To ensure that changes are appropriate and have been subjected to the appropriate review and approval.

Security is not just "prevention," it's

- Prevention
- Detection
- Control
- Correction

When you have all those, you have (of course) a more reliable system. Let's review each of the five elements in turn.

## 5.1 Access Control—Authentication and Authorization

Please allow me to be ridiculous for a moment: if you have no access control, anyone can do anything to your systems, and so they are almost by definition unreliable. And you're similarly exposed even if you have good access control but have no authorization mechanism which limits what different users can do.

I'm going to discuss access control in two parts, authentication and authorization. I'm going to further subdivide the discussion of authorization into logical access restrictions, physical access restrictions, and activity restrictions.

### *5.1.1 Authentication and Access Control*

Authentication is the process of determining "who" someone is, in order to decide whether or not they should be allowed access to a system or service. Authentication is usually implemented by pairing some form of "identifier" (typically a userid) with one or more of:

- something you know: a password, secret phrase, or secret lodge handshake
- something you have: a key, smart card, token, or magic decoder ring
- who you are: a recognizable face, fingerprint, retinal scan, or DNA test

The last form, "who you are," is often more complicated to implement and is somewhat hard to use when connecting remotely over a network.

The most common form of authentication, and one we are all familiar with, is "something you know": some form of userid and password pair that "proves" that the user actually is who he or she claims to be.

In theory, both the userid and password could change with time, but the most common implementations involve a publicly known userid and a static password.[1] Static passwords are most commonly stored on the destination machine (or network of machines), typically in an encrypted or obscured form to prevent the casual browsing of passwords. Passwords are sometimes shared across groups of machines with the use of such things as NIS, Windows NT domains, or various password file distribution schemes.

The most commonly used "more secure" static password mechanism is Kerberos [*Steiner88*], where the passwords are stored on a "secure" server, and the protocol protects the passwords as they are passed back and forth between various clients and servers to authenticate users. One problem with Kerberos is the weakest link problem—you need to have Kerberos locally available on every device, or you risk sending your password over some connection in clear text, which limits the effectiveness of Kerberos in some environments.[2]

1. I'll define a static password as one that stays the same until it is explicitly changed, typically by the user.

2. You can sometimes secure otherwise clear text links with ssh [*Ylonen96*] encryption, but then you need to have ssh on the local box, which is the same problem but with a different piece of software.

More advanced (obfuscated or annoying) systems use some form of one-time password (OTP) system to guard against password eavesdropping (over the shoulder or over the network) and password sharing. Some OTP systems are software only (such as S/KEY [*SKEY*]) but the more common approach requires the use of some form of token ("something you have"), which computes or reports the next password in the series. The most commonly used token is probably SecurID from Security Dynamics (now RSA), but other types of token card are available from a variety of vendors. Some tokens use a challenge-response method, others require only the current number or phrase from the token. OTP systems have some mechanism to guard against password re-use, and token-based systems typically also require some sort of secret PIN in addition to the token in order to authenticate.

The reason for an authentication mechanism is to reliably identify the user. The more reliable the authentication mechanism, the more reliable your overall system is going to be—a better "front door" defence will better protect you from the unreliable among us. I'm a big fan of one-time passwords—to my mind only the most rudimentary of systems would not benefit from the use of OTPs, even if used only for authenticating the more privileged users or for granting root or other privileged access.

### *5.1.2 Authorization and Access Control*

Authorization is the explicit or implicit granting of access to particular files, devices, systems, or applications. In many cases, little or no distinction is made between different users on a system; users are often given equal access to everything. Authorization is a policy or process decision; access controls and restrictions are used to implement the authorization decisions.

#### 5.1.2.1 Logical Access Restrictions

The next element in an access control system is what I will refer to as "logical access restrictions." Logical access restrictions are those which allow or disallow access based on the attributes of a user or a connection. The most common logical access restrictions are restrictions based on a person's userid or group memberships, but they can also be based on such things as the originating network address of a connection, the time of day, or current usage rules.

The most common way to implement originating network restrictions (in the UNIX world at least) is through the use of the "TCP Wrapper" [*Venema*] package. This package makes it easy to "wrap" certain services (such as telnet) with an access control program which can restrict based on originating network address or other similar conditions. Other logical restrictions are more commonly implemented with certain operating system configurations, custom shells, or commercial software.

Some of the logical access restrictions that you might want or need to implement include:

**No Multiple Logins**

> You may wish to limit concurrent access to particular applications or systems for reasons of system load, security, or (business) process control.

**No Logins From Multiple (Apparent) Locations**

You may wish to prevent users from being in two places at once, primarily for security reasons. If a user is in your office, working away, it might be safe to conclude that a login attempt from halfway around the world was not actually the same person that is authorized to use your system.

Of course, a connection from two different locations doesn't automatically mean that it's two different people, since the person could have connected to the remote machine over the network before connecting back, but it would mean your traffic might be taking a long, possibly exposed route, which probably isn't a good thing either.

**No Off-Hours Connections**

It's probably not reasonable (or expected) for an ordinary accounts payable clerk to be doing a check printing run at 2 a.m. on Sunday—you might want to use "off-hours" restrictions to prevent that before it happens.

**No Connections During Maintenance Periods**

Sometimes you need the machine to be up and running, but don't want to allow (ordinary) users to sign on—allowing users on during system maintenance can sometimes just make things more complicated. One classic example is doing full backups to a network backup server in preparation for an OS upgrade. On UNIX systems, user logins can sometimes be prevented with the /etc/nologin file, but not usually very successfully.

**Peak-Load Restrictions**

You might wish to refuse additional logins if the system load (however that is calculated) is above some threshold. This can help avoid making an already bad situation even worse.

As a more concrete example, I'll mention that I once implemented an access control system that made use of almost all those restrictions. The system was a job database for university students, serving several thousand students each term. The usage on the system was very "peaky"—there were a few periods of very high demand and long periods of almost no use. We wanted to make sure that each student was using only one session at a time, that we didn't let on too many simultaneous users (because otherwise the system would crawl to a halt), and we wanted to be able to prevent user sign ons during the daily update window and during periods of "emergency" maintenance. We did all that using a simple shell script, 50 or so lines long. This script was used as each student's login shell, proving that (once again) complicated solutions are not always required.

It's probably worth mentioning that mechanisms and policies like this have a long history in the mainframe world.

#### 5.1.2.2 Physical Access Restrictions

Complementary to logical access restrictions are (of course) physical access restrictions. Sometimes you may wish to only allow access to a particular system, application, or function if the user is (thought to be) in a "secure" location.

For UNIX systems the most common example of this kind of restriction is to only allow direct root logins from the system console. Other examples include only allowing connections from within your building—enforced either through the use of hardwired connections (almost unheard of in these days of networked workstations), subnets and firewalls—or simply not allowing any connections (network or dialup) to or from the outside.

Locked machine rooms are another physical access restriction—see section 5.2, below.

#### 5.1.2.3 Activity Restrictions

The final component of access control is what I'm going to call "activity restrictions." These are restrictions or limits on the commands and functions that a user can invoke. These come into effect after your authentication system has identified a particular user, and the user (or the user's connection) has passed any logical or physical access restrictions that have been implemented.

One of the most common (UNIX) examples of activity restrictions is the common requirement that a user be a member of a certain group, often "wheel" or group 0, in order to "su" to root. Lots of other examples of user group or ACL- (Access Control List) based restrictions exist. Other restrictions can be implemented by applications, using compiled-in information (bad), or files or database entries with restriction or permission information.

I'm going to suggest dividing activity restrictions into three types:

**Static Activity Restrictions**

These are yes/no restrictions, independent of other considerations, such as date or time, other users, etc. Some examples are:

- group membership requirements, as mentioned above
- the prevention of access to a general purpose environment by such mechanisms as menuing systems, restricted shells, etc.

**Variable Activity Restrictions**

Restrictions that are based on straightforward but varying information, such as date or time, connection origin, etc. Some examples are:

- no recreational Web sites during business hours
- not allowing check printing outside regular hours
- no root or privileged access if you're not on the local network

**Complex Activity Restrictions**

Restrictions that are controlled by other events, situations, status, etc. Some examples are:

- permission granted or denied based on file or database contents
- task allowed only at certain steps in a business process
- operation allowed only when an operator is on duty

#### 5.1.2.4 Access Restriction Summary

You will have noticed that the line between activity restrictions and logical and physical access restrictions gets a little blurry sometimes.

I haven't covered all situations here. One obvious situation that's not covered is multiple authentication, where two or more people must agree and authenticate before a task is executed.[3] Another is the use of biometric information for authentication. And I haven't mentioned the need for proper logging, which is a necessity for tracking, troubleshooting, and change control.

Some of these access control mechanisms can be quite inconvenient and/or obtrusive. As in most other discussions of reliability, there's a tradeoff between reliability and control on the one hand and cost and inconvenience on the other, and each organization must strike the most appropriate balance for its needs.

And to tie this discussion back to reliability, good access control means that you limit, control, or track, who did (or could do) what, when, and under what circumstances. This means that when you determine that certain controls or limits are required to help your systems, networks, and business processes function reliably, you've got (at least part of) the mechanism to help you implement them.

## 5.2 Physical Security

The preceding discussion has focused primarily on electronic access to systems and networks, which is the traditional area of concern for computer oriented people. But it's just as important to consider the physical security aspects of your system and networks, and again balance the costs (monetary and otherwise) against the expected risks and/or advantages. Note that this discussion is not about disaster recovery planning or high availability hardware—it is about preventing people (or things) from getting physical access to your premises or equipment.

Why is physical security important? In most cases, physical access to a machine is tantamount to administrator access. In the most extreme cases, a machine (or parts of it) may be stolen and attacked at the thief's leisure, whim, or screwdriver. Physical security is intended to guard against those kinds of threats. Physical security can also help to guard against so-called acts of God—a more secure building is likely to be stronger and more appropriately located.

What kinds of things should physical security guard against, and how do they contribute to reliability?

**Equipment Theft**

> If your computers, disks, or network hubs are missing, it's hard to offer a reliable service. It's also getting fairly common for people to steal internal components, such as processors, memory, or disks, as they're easier to carry, fit nicely into a pocket, and are often not immediately obviously missing.

3. Recall those action movies featuring nuclear missile silos where two people have to turn two different keys on opposite sides of the room at the same time. And they're both carrying guns.

**Media Theft**

Removable media (disks, tapes, cartridges, and the like) can contain very useful information (consider the backups of your customer or personnel database) and are often easy to carry and not likely to be missed very quickly. The business risks here are obvious, but the impact on reliability is less so. Reliability can be reduced by the possibility of the use of confidential information to attack the organization at a later date and by the lack of backups, which could make recovery a very painful process.

**Console or Network Access**

Unauthorized access to console ports or network connections or devices can open you up to all sorts of attacks (such as password sniffing and bug exploitation) that can cause your systems to start behaving unreliably, unintentionally or otherwise.

**Physical Destruction**

A fire ax can render even the best systems and equipment somewhat unreliable.

What sort of physical security controls should be considered?

**Locks**

Of varying and appropriate levels of sophistication. It's not unusual to start making the locks more difficult and complicated the closer one gets to the important stuff. Consider and contrast the use of ordinary keys that can be copied at the local all night convenience store, high security keys that require special blanks and machines to duplicate them, magnetic or other types of access control cards, combination locks, biometric scans, etc. And remember the hardware that the lock mechanisms are attached to—a high security key cylinder on a two-dollar latch secured with one-inch wood screws might not be the most prudent way of securing your machine room.

**Access Controls and Logging**

Electronic logs of who went where and when are handy after the fact and can act as a convenient deterrent. Time of day restrictions that can be used to prevent (or limit) physical access in the middle of the night when no one else is around are often useful, as is requiring the cooperation of two people to open a particular door. These all have their place in an access control plan.

**Structural**

Check your walls, ceilings, and floors for ease of access and/or destruction. Are your high security walls constructed of a single layer of wallboard? Does your nice concrete block wall only extend to just above the ceiling tiles? Do you have nice hollow wooden doors rather than proper steel fire-rated doors and frames with effective latches?

**Monitoring**

Consider the use of fire and burglar alarms, video monitoring, motion detectors, and other physical monitoring techniques. Make sure that it's hard for an intruder to get at the video tapes and destroy them and the evidence that they contain. And don't forget to make sure that your tape rotation, storage, and retention practices guard against inadvertent destruction or theft of the tapes.

**More Extreme**

Some organizations will find it worthwhile to go to greater lengths to secure their premises, with such things as armed guards, "man traps" (small rooms with two independently locking doors that you must pass through when entering or leaving), attack dogs, and the like.

## 5.3 Change Management

The best laid plans can be all for naught if there are no controls around them, and one of the most important controls is change management. Change management is intended to provide a standard process for documentation and approval of all changes to systems and networks.

The primary components of a proper change management system are as follows:

**Review**

Proper review and testing of any proposed changes will greatly increase the likelihood of a successful, non-disruptive change, and will help prevent intentionally or unintentionally malicious changes from being undertaken. Note that a proper review also ensures that there is proper documentation, including how, why, what, when, and a description of the expected impact and results of the change.

**Authorization**

Ensures that changes (to, say, the payroll system) have been properly authorized according to the policies of the organization. Some systems might require only a very low level of authorization to change, but you might want to make sure that any changes to the CEO's laptop are only made by a limited set of people.

**Proper Implementation**

A standard and documented implementation process will help avoid mistakes, will keep downtime to a minimum, and will make your maintenance windows much more bearable.

**Ability to Back Out**

This is often overlooked, but a proper change management system is prepared to deal with changes that fail (in whatever way) once they are put into production, and ensures that there is a way to get back to the pre-change working, reliable system. There are some situations in which

backing out a change is really not feasible. Make sure that your change management documentation and approval process can properly handle that kind of change.

Change management processes can be implemented in many different ways, from pencil and paper, to email, to database backed forms and workflow systems. Choose a method that is right for your organization.

## 5.4 Vulnerability Detection

Vulnerability detection is the process of searching for problems and potential problems so that they can be corrected or protected before they can be exploited by an attacker. By detecting vulnerabilities as early as possible, you can reduce the risk of an attack, a compromise, and a failure.

There are many software packages and services that will perform some level of vulnerability scan. Two of the best known are COPS [*Farmer90*] and Crack [*Muffett*].

## 5.5 Intrusion Detection

The best security system in the world is reduced in its effectiveness if it's not properly monitored. You must have some mechanisms and processes that are designed to detect any intrusions that do take place and, optimally, any attempted intrusions that were blocked by the systems. Proper intrusion detection systems will alert you when you're under attack and will give you time to increase your awareness or monitoring to fend off any further attacks.

Consider what might happen if the file containing your encrypted passwords was stolen. If you can detect that kind of theft, you have a better chance of changing all the passwords before they get cracked and exploited, and blocking the access used to intrude. Quite simply, if you can't detect when something has gone awry, you've got much less chance of protecting yourself and your systems. And if you can't protect the systems, it's going to be harder to keep them working reliably.

Techniques and mechanisms for intrusion detection include:

**Log Review**

Review your log files, either manually or using some (well protected) automatic tools, so that you'll have a better chance of noticing the unexpected when it happens.

**Real Time Network, System, and Premises Alarm Monitoring**

Make sure you have monitors and alarms in place, on your systems, networks, and your premises itself. On your systems and networks, look for unexpected changes, unusual traffic patterns, or specific kinds of traffic or commands. Page yourself or the security company when alarms get triggered, or have someone on duty onsite. A quick response to an alarm or alert can limit the amount of damage that happens, and can also serve as a deterrent if the attacker is simply looking for a fertile playground and not targeting you specifically.

**Periodic Analysis**

Tools such as tripwire [*Kim94*] can advise you when files change unexpectedly. Review historical trends and changes in such things as disk usage, network traffic, system load, and so on. MRTG [*Oetiker98*] or Cricket [*Allen99*] can be very useful for visual analysis of trends and activity.

There are a number of commercial intrusion detection systems available, and the ambitious or budget-limited can put together various combinations of freely available and home-grown tools for various levels of intrusion detection ability.

## 5.6 Correction

Once you've detected an intrusion or attack (or attempted attack), you need a mechanism and process by which you can put things right again, and hopefully a way to prevent it from happening again.

It's worth mentioning that having appropriate policies and procedures in place before an attack is detected is very important—it reduces the feeling of panic during an event, and helps ensure the most effective response possible.

Keep good backups, know where your distribution media are, and have documented procedures and mechanisms to get in touch with the necessary people. Keep current on vendor updates, notices, and security alerts in the community at large. Be ready to disconnect machines or networks that are under attack or need repair while you investigate and undertake repairs, should your process determine that that is the best or most appropriate action to take.

The impact on reliability should be clear—a modified machine or system is at risk, and the sooner you can get things back together, the sooner normal, reliable operation can resume.

The other side of correction is "re-motivating" individuals who are acting contrary to policy and reasonable standards of behavior. A user or system administrator who behaves incorrectly (let's say by choosing trivial passwords and writing them on note paper stuck to his or her monitor) can be putting other users, systems, and information at risk. If you're expecting people to act appropriately, you had better define and publish what the standards of behavior are, and be prepared to enforce or explain them. See chapter 10 for more on interactions with your user community.

## 5.7 In Summary

Security is a wide-ranging topic and has an impact on many areas of an organization's activities. Proper security systems, mechanisms, policies, and practices sometimes augment reliability, but in many ways their primary reliability benefit is in preventing the intentional or unintentional reduction of current reliability levels.

# 6 System and Network Monitoring

No matter how carefully constructed and maintained your environment, sooner or later, something will go wrong. A disk will break, a cable will fail, an unexpected process will go haywire. Something, somewhere will break, no matter how perfect your environment is.

Before you can correct a fault, you have to be aware of it. You basically have two choices:

1. Wait until some user brings it to your attention, and hope that it's not your boss who first mentions it.
2. Have sufficient system and network monitoring systems in place and in use that you're the first person to know.

The choice is obvious. Or at least it is in my book(-let).

You will have noticed that I've mentioned monitoring elsewhere in this booklet. But it's important enough, and too often insufficiently implemented and used, that it deserves a chapter of its own.

## 6.1 What Kind of Monitoring?

In this chapter we're concerned primarily with monitoring systems and networks for proper operation and activity, rather than for intrusion detection (which is reviewed briefly in section 5.5). While some intrusions come to light in the normal course of monitoring, that's typically more a by-product than an intended result.

Monitoring is done primarily for three purposes:

**Exceptions**

Exception monitoring is the most obvious reason to monitor systems and networks. Exception monitoring is intended to identify and report unusual conditions, failures, and possible problems that may need attention. For example, a monitoring system should notice if a machine is down or unreachable and generate some form of alert or notification to bring it to the attention of the administrators.

Exception monitoring is usually concerned with specific indicators, such as reachability, available disk space, or network traffic levels, and it usually compares them to specific states (on/off, up/down, and so on) or thresholds (percentage free, bytes per second, or similar measures).

Jargon: An unusual or unexpected condition or a problem is an "exception." An exception that is identified and is to be reported generates an "alert" (or alarm). An alert is "dispatched," by some mechanism, to either a monitoring system or a human.

**History**

Monitoring for historical analysis involves the collection and retention of specific time-stamped data, primarily for record keeping and troubleshooting. Historical traffic and usage data is often used for charging customers, internal or external, based on their activity, or for measuring compliance with service level agreements. Troubleshooting and problem resolution can be aided by having state data available from the relevant time periods.

**Trends**

Trend monitoring is intended to provide information for forecasting future usage and to summarize past experience. This is similar to monitoring for history, but trend analysis doesn't require that all historical data be retained. Old data can be averaged or summarized to retain the trend information while throwing away the specific data points, thereby saving data storage space and reducing the calculation time required to perform analyses.

## 6.2 What Should You Monitor?

What you should monitor depends on your environment, your needs, and the goals you've set for your systems and networks.

There are many, many things that you could monitor, and you could easily overwhelm yourself with data, graphs, alarms, and summary messages if you chose to. Many organizations are short-staffed these days, and the to-do lists are often longer than can be reasonably handled. Too much monitoring could result in even more things to worry about.

That said, some of the most common things to monitor are:

> You need to strike a balance between the monitoring that you need, the monitoring that will make your life easier, and the monitoring that will just overwhelm you.

- utilization of disk space, CPU, and memory
- system uptime and availability
- network connectivity, interface activity
- bandwidth utilization and errors
- queue sizes—mail, news, printing
- Web pages served
- system errors and events
- service availability, system processes
- users and usage
- environmental statistics such as temperature, humidity

If you're an ISP, network utilization and dialup use will probably be very important to you, and you'll probably want to collect and retain all that data (and more). If you're a university, you're probably going to want to track things like disk utilization, some

forms of user activity, and CPU utilization. If you're a Web-based business, then you'll be interested in service availability, and utilization statistics.

## 6.3 Where to Monitor

Examine your environment, determine what elements are most important and effective for you, and start there.

Monitoring can take place in two places: on the machine you're monitoring or from a remote machine.

### *6.3.1 Local or On-Machine Monitoring*

Local monitoring is usually the simplest to implement if you're dealing with only a few machines—a few shell scripts, a crontab entry or two, and you're set. In reality, it's not usually quite that simple, but it's close.

Local monitoring is typically used to check the condition or state of the local machine, and log or email the results or alert messages. Some problems and situations make it possible for the monitoring script to correct problems that it discovers. For example, a script that checks whether sendmail is running could restart sendmail if necessary.

For exception monitoring, some of the things that you may wish to do are:

- Use commands like lpq, lpstat, and mailq to check queue status.
- Use df and mount to check disk space.
- Use commands like uptime, vmstat, and iostat to check activity levels.
- Use ping, nslookup or lynx to check connectivity to remote systems.
- Look for old files in spool directories using find.
- Use grep or awk to look for unexpected events in system log files.
- Check for process existence with ps.

For history or trend monitoring,

- The who command can (more or less) tell you how many people are using a machine. The idle time shown by w can suggest whether they are active or not.
- ps, cut, sort -u and wc -l can tell you how many different users have processes running at a particular time.
- uptime gives you the load average.

There are, of course, many other possibilities. Use the logger command or a log file of your own to record your observations.

Experience suggests that most systems aren't delivered with all the local monitoring tools that you would like them to have. This means that implementing comprehensive local monitoring and reporting on any more than a small number of systems can be a complicated and time-consuming task. But an integrated combination of local and remote monitoring can be very effective.

### *6.3.2 Remote Monitoring*

Remote monitoring is typically coordinated or performed by a central system—a "network management station"—running software that generates, collects, and records, displays, or dispatches monitoring information.

The management station typically collects data in two ways:

- by sending probes to remote systems or devices and recording the results
- by receiving alerts or "traps" from other devices that have identified something worth reporting

It then typically either generates alerts of its own (via mail, pager, or telegram), logs the data, or displays it on or makes it available through the management system's "console" application.

Some of the following discussion implies that the remote device is a general purpose computer, running a variety of services. The discussion also applies to simpler network devices, such as ethernet switches or routers, virtually all of which have a variety of standard services built in. If you're monitoring a special purpose device, adapt your monitoring methods to match the services that the device supports.

#### 6.3.2.1 Simple Network Management Protocol (SNMP)

The most common mechanism for querying and controlling network connected devices is the Simple Network Management Protocol (SNMP). SNMP is implemented (or available) in virtually every reasonably current network-connected device, from hubs and power switches, to high-end routers, to almost every general purpose computer.

A basic SNMP implementation provides an ordered collection of data elements, such as system uptime and description, network interface information and statistics, and routing and protocol information, that can be queried and manipulated over the network. The SNMP protocol uses UDP to send and receive information between an SNMP manager running on a network management station and an SNMP agent running on a remote system or device. The protocol defines five packet types: GetRequest, GetNextRequest, SetRequest, GetResponse, and Trap.

In normal operation, a management station configured to monitor a device would send a variety of SNMP GetRequest packets, requesting specific information, and the device would reply with GetResponse response packets containing the requested information. An SNMP agent running on a device can be configured to send alerts to notify a management station of exceptions, by sending the management station SNMP Trap packets that contain the relevant information about the exception.

Many current SNMP agents used on computing systems are "extensible." They can often be configured to run local commands or tests in response to specific SNMP get requests, returning an integer or string result code. This typically makes it possible to implement virtually arbitrary local monitoring that is managed and tracked remotely, which will usually result in a monitoring infrastructure that is both simpler overall and more effective.

SNMP is a little obscure when you're just getting started with it, but it is very powerful, and well worth being familiar with. There is a wide variety of SNMP tools and applications available, of widely varying sophistication and price, most of which can be installed in or adapted to suit just about any environment. A little effort can provide you with some very useful tools for monitoring and problem solving.

See [*SNMP*] and [*Zwicky99*] for more information and references on SNMP.

#### 6.3.2.2 Other Probes

There are three other obvious mechanisms for remote probing of systems or devices:

**Pings**

The ping command can be used to implement a simpleminded test of network connectivity. If you can ping a remote server or router, chances are that it's on the network, and at least partially functional. There are modified high-performance ping commands available written specifically for use in monitoring systems, that can do multiple hosts in parallel and so provide better overall throughput.

It should be pointed out that ping monitoring is not highly reliable. Busy routers will often ignore ping packets, some firewalls and gateways are configured to block ICMP packets (which are used by ping), and general network congestion can prevent ping packets from getting through. A ping failure doesn't necessarily mean that there is anything wrong, but it may indicate congestion or high levels of activity somewhere on the network if pings suddenly stop being returned.
That said, ping probes are still a useful, low-overhead monitoring tool.

**Port Probes**

Port probes are used to determine whether particular network services are available and working. The basic port probe tries only to open a connection to a particular port and assumes that a connection indicates a working service. The better port probe opens the port and actually engages in a protocol exchange with the remote machine to ensure that the service is actually talking on its port.

A word of warning—make sure that a transparent proxy or Web cache isn't located between your management station and the remote system or device, fooling your management station into thinking that all is well.

Port probes are probably most commonly used to check the accessibility of Web servers but can be used for all sorts of services. A successful connection to port 80 on a remote machine tells you that the machine is reachable and at least in a basic operational state. A better probe would be to actually engage in a simple protocol exchange, by sending an HTTP GET or HEAD request, and then examining the output for a successful status code or a regular expression match on the page content

itself. That would prove that the Web server was actually serving up data, in addition to listening to the port.

**Remote Execution**

A remote execution probe is when the management station causes a command to run on the remote host and then examines the output for the expected results. For example,

```
% ssh otherhost uptime
```

is a simple check that the remote machine is accessible, up, running (and running a UNIX-like system), and can perform some form of basic authentication. Remote execution probes are sometimes difficult to deal with effectively, as it's often difficult to separate error and command output, the interaction isn't always well defined, and various unpleasant hangs and complications aren't uncommon. It's more commonly used as a special query tool or problem determination tool, rather than as a normal periodic system health probe.

One other category is worth mentioning: proprietary probe and analysis tools. The most common of these is likely the "Performance Monitor" that comes with Windows NT, which provides local or remote monitoring of a wide variety of system statistics and information. I won't say much about proprietary probes here, other than to point out that no matter the quality of the tool, a proprietary tool is going to be more difficult to modify, adapt, or integrate into a multi-vendor environment.

#### 6.3.2.3 Remote Traps and Alerts

The final element of remote monitoring to mention is the handling of traps and alerts generated locally on a device and dispatched to a network management system. A local monitor can dispatch an alert to a network management station via email, an SNMP trap, remote execution, syslog to a central host, or other purpose-specific methods. The key point is to make sure that your local monitoring meshes well with your primary network management system. It's almost always better to have one central system that deals with as many of the alerts and problem reports as possible, as it allows you (or your support staff) to stay focused and keep the relative priorities of the various alerts in mind.

### *6.4 Alerts, Notifications, and Observation*

Once you're generating all sorts of logs, alerts, status messages, problem reports and so on, don't forget to put them to use. Arrange to get paged for serious problems (software such as QuickPage [*QuickPage*] or HylaFAX [*HylaFAX*] can be very useful there), send email for the medium priority problems, and log the low-level noise in case you ever run out of emergencies to deal with.

If you're lucky enough to have an operations center, consider having a monitor always showing the top priority problems. Or, if you have customers regularly visit your operations center, get several giant projection screens so you can show problems, maps, pictures, and graphs to impress the customers while you watch CNN on cable.

If you're the operations center, choose a method that works for you, so that things don't get ignored too long, and so that you have an opportunity to assign some sort of priority to alerts and problems. A trouble ticket system can be useful in some cases for tracking problems, whether they're machine- or customer-generated complaints.

Depending on your business and your organization, you may also want to consider sending automated notifications to your customers for certain kinds of problems or outages. It's possible to generate email, pages, or faxes (depending on the customers' preferences) automatically, if you can reasonably classify your alerts and if you have confidence in your ability to avoid false alarms. Yes, there is the possibility of airing your dirty laundry in public, but it's sometimes the case that customers very much appreciate a flawed but open and honest provider more than they appreciate a closed and "perfect" provider.

## 6.5 Tools and Applications

As mentioned, there is a wide variety of tools and applications for system and network monitoring and management, of all complexities, sizes, intentions, and prices. Some are intended to be monolithic applications that solve all your problems (and wash your dishes in the built-in "kitchen sink" at the same time). Some are intended to be building block tools that you can combine in the best manner for your environment. Some work better on smaller networks, some work better on larger networks.

One specific distinction that I would like to make is the following. Some applications provide a default, detailed, "all is well" status screen, full of information at all times. Other applications provide a display of current alerts, and when all is well show only a blank screen. I'll claim that, in general, Big Brother [*BigBrother*] is an example of the former[1] and that NOCOL [*nocol*] and Netcool [*Netcool*] are examples of the latter. I think the distinction in philosophy is a worthwhile one to keep in mind, and that the size of your network or problem domain may cause you to prefer one or the other.

That said, I would like to mention the following tools and applications as being significant and worth reviewing when you're looking for your personal monitoring nirvana.[2]

### *6.5.1 Tools*

**UCD SNMP**

The University of California, Davis SNMP distribution [*UCDSNMP*] (originally based on the CMU SNMP implementation) is virtually a must-have for anyone working with monitoring to or from UNIX systems. It's a great tool set of both agent and management tools, nicely extensible, and easy to deploy and understand. It's the UNIX SNMP agent that I use and recommend.

1. Big Brother does have a number of summarization mechanisms, to limit its output to only those items that are "interesting," but it still gives me the feeling that it really would rather show me everything.

2. See [*Monitoring*], [*SNMP*], and [*Zwicky99*] for additional monitoring and SNMP references.

**Scotty and Tkined**

Scotty is a Tcl (Tool Control Language) shell with the Tnm network management extension, and Tkined is an interactive editor for creating and maintaining network maps. Both are terrific, and great starting points for all sorts of applications and uses. If you're thinking of doing any SNMP programming or application development, look seriously at scotty. To my mind, Perl or C are painful alternatives in this realm. See [*Scotty*] for more information.

### *6.5.2 Open Source or Freely Available Applications*

Some of these applications have both freely distributable and commercial versions available—check the references for more information.

**MRTG**

MRTG, the Multi Router Traffic Grapher [*Oetiker98*], is used by virtually every ISP and network provider that I've ever heard of, and is a fantastic tool for visual trend analysis (i.e., graphs) of all sorts of useful things. It's very adaptable and very flexible. If you've got a network, get MRTG.

**Cricket**

Cricket [*Allen99*] is essentially a follow-on to MRTG that scales much more effectively and addresses a few of the problems that arose in MRTG. If your network is non-trivial, get Cricket too.

Compare Cricket with MRTG and choose the one that's best for your environment. But no matter what, get one or the other, or both.

**Big Brother**

Big Brother [*BigBrother*] is a Web-based system and network monitor, and has achieved wide adoption in a relatively short time. Primarily for status and outage monitoring, rather than for trends or history, it comes with all sorts of monitors and probes. Definitely worth a look, especially if you have a small- to medium-sized network to worry about.

**NOCOL/SNIPS**

NOCOL (Network Operations Center On-Line) [*nocol*] has a long history and is a solid application, with good probing and monitoring and reasonable scalability. The standard interface typically lists only current problems and provides several levels of detail. Check it out if you have a medium-sized network, or if your preference is for exception-based reporting.

### *6.5.3 Commercial Applications*

There are all sorts of commercial monitoring and SNMP applications, some of which are vendor or OS specific, others which are extensible and have wide-ranging support. Three of the best-known are:

**Netcool**

High-end monitoring and status for large networks. Very scalable, extensible, and customizable. Used by a number of large ISPs and network providers. See [*Netcool*].

**OpenView**

The best-known of the centralized network management systems. Lots of plug-ins and extensions for management and configuration of a large variety of devices. Not trivial, but worth a look to see what's possible. See [*OpenView*].

**Spectrum**

To my simpleminded mind, primarily an alternative to OpenView, but it has advantages and disadvantages of its own. See [*Spectrum*].

# 7 Backups, Restores, and Data Recovery

In chapter 1, I outlined some "general guidelines" (section 1.5) for reliability. One of those general guidelines was "plan for failure and recovery," and that's going to be the primary focus of this chapter. More specifically, this chapter covers backups, restores, and data recovery. The focus will be on how you deal with your data, rather than on which data you need to back up.

Let me remind you of the basic tenets of reliability: service levels, risk evaluation, costs of failures, appropriateness for your environment, and so on. One nice thing about this chapter's focus is that almost everyone will agree on the necessity for proper backups, so your justification document/business case may be a much easier sell this time.

Let's define what we mean by backups, restores, and recovery. In this chapter we're going to be concentrating on data backups to some secondary storage medium.

A "backup" is a duplicate copy of programs or data, usually kept offline and on different (usually slower) media (such as magnetic tape). Backups can be "full" (which include everything), or "incremental" (only the changes since a previous backup). A "restore" is the process of retrieving a copy of one or more files from the backup media. And "recovery" is what happens when something goes very wrong (fire, flood, earthquake, theft, presidential scandal) and you need to put everything back in order.

I'll also mention "archival storage." An archive is very much like a backup, except that it's intended to be kept for the long term, perhaps forever. Again, the media used varies depending on cost, security, and retrieval considerations. I've mentioned "archival storage" primarily so that you'll know what I'm not talking about.

## 7.1 Why Backups Are Important

I should mention, just for the record, why it's important to do backups. In a very real sense, it's not the backups that are important, it's the restores and recoveries. Most people have had the experience of accidentally deleting or corrupting an important file, and most system administrators have also been faced with users who have done just that. And it's not just human error that can corrupt files—as hard as it may be to believe, some software actually does have bugs. And while disks are more reliable now than in the past, sooner or later a disk is going to fail, burn, or get stolen, and you're going to need to undertake a disk recovery. Think of backups as insurance—the mere presence of reliable backups not only prevents your disks from failing, it also keeps you from getting fired the next time there's a flood in your machine room.

What's important about backups? Backups should be

- Current
- Consistent
- Complete

Current and complete are easy to define—you need to do backups on a regular and reliable schedule (usually daily), and you need to back up all the files and directories that you intended to back up.

Consistency is a little less obvious—your backup system should be able to deal with files that are changing during a backup. An easy example of a changing file is a large file used for database storage, which could easily change between the time you start reading the file to back it up and the time you've finished reading the file, thereby giving you a nice-looking, but useless backup. Backups also need to be available when you need them—more on that later.[1]

One of our reliability key words is "automation" (section 4.1.3), and backups are a perfect task to be automated.

- They're done regularly.
- They're boring (but necessary).
- They're very important.

Whether you use a home-brewed shell script, some freely available software, or an expensive commercial product, your goals are the same: get the backup done, and get it done right.

Let's examine the different components of a backup system—software, hardware, physical location, and media handling—and consider how each contributes to reliability.[2]

## 7.2 What and What Not to Backup

I mentioned above that your backups should be "complete." I didn't mention that your backups should not be "overly complete."

The choice of which systems, file systems, directories, and files to back up is often more complicated than one would hope. Backing up too much data isn't anywhere near as bad as not backing up enough, but backing up too much data can have significant implications on backup speed, network bandwidth, and media use.

What should you back up? Consider:

- data and documents used in or generated by your organization
- locally generated, modified, or maintained software
- password files and other authentication and authorization information
- system configuration files and information e.g., the /etc directory on UNIX systems

1. For a good discussion of various backup-related issues, see [*Shumway91*] in the LISA '91 proceedings, which also contains a number of other backup-related papers.

2. For a discussion of backups in greater depth, you may wish to consult [*Preston99*] or [*Leber98*].

- user home directories
- email spool directories

What might you not want to back up? Consider:

- email—some companies avoid backing up email explicitly, so that they won't have to search for incriminating (or other) evidence in old mail spool directories
- stock vendor directories, such as usr, that can be "easily" replicated from installation media or another similar machine (which assumes that you always have the original media and a "spare" machine handy)
- user desktops or laptops, regardless of what valuable (or incriminating) documents might reside there, in order to save a lot of time, media
- USENET news, print, or other "spool," temporary, or scratch directories
- multiple "replicated" machines, such as Web or proxy server farms, or workstations in a student computing lab

> Weigh the time and cost of doing the backups against the value of the data and the cost of reproducing or regenerating it in case of a failure, and then throw in a measure of legal liability, responsibility, and the ability to retroactively track suspicious or suspect activity.

The primary point here being that you should approach the "what" question from both directions (both what you need and what you don't need) and ensure that your practices and policies reflect both your intentions and your underlying needs.

## 7.3 Choice of Media Type

The most common medium for doing backups is magnetic tape. Magnetic tape comes in a wide variety of sizes and types, some developed specifically for data storage, and some adapted from the entertainment world. Two examples of the latter are 4mm and 8mm tape cartridges. In the early days of the PC, audiocassette tapes and VHS videotapes were used for some low-end applications, but they have pretty much disappeared (thankfully).

Tape formats and tape drives are (obviously) closely tied together. I'll cover them both in section 7.5.

Some situations call for alternatives to magnetic tape, such as regular old hard disk (or DASD for you big iron fans), floppy disks (and their removable media cousins, such as Iomega's Zip and Jaz units), CD-ROMs, optical disks, paper tape, and punched cards (though the latter two have fallen somewhat out of favor in recent years).

In practice, almost all backups involve magnetic tape of some form, so I won't discuss alternative media further.

## 7.4 Software

Most operating systems these days come with some form of backup software, some of it more suitable for the purpose than others (see [*Zwicky91*] for a review of the good

and the bad). The best of the "stock software" lot is typically the dump/restore combination—they're not perfect, but they tend to do a "reasonable" job.

### *7.4.1 Home-Grown Software*

Many sites have written home-grown wrappers around the stock commands, with varying degrees of complexity. You can try to deal with labeled tapes, tape switching, unattended execution, tape cataloging, and all the other things that you really should do, but if your disks are small enough (or your tapes big enough) your backup script can be very simple.

I once set up a backup "system" for a small, isolated UNIX machine which involved a clerk putting the day's tape in the tape drive, signing on as "backup" (which ran a backup script as its shell), dropping the previous day's tape in the campus mail to an offsite location, and going home, leaving the backup running. Dead simple, but appropriate for the situation.

At the other extreme of home-grown software, I've seen systems that track tape numbers, maintain a flat-file index of which file systems from which hosts are on which tapes, and provide various query and operator-notification mechanisms.

### *7.4.2 Packaged Software*

But, unless your needs are very simple, I'd recommend against rolling your own and re-inventing the wheel. You're almost certainly better off using existing software.

If cost is a concern, I would recommend that you investigate the Amanda Backup Manager, available from the University of Maryland [*Amanda*]. It's built around standard utilities (such as dump/restore, GNU tar, and others), does labeled tapes, has some support for jukeboxes and tape changers, maintains a database, and works on a wide variety of systems and across the network. Very good software and definitely worth a look. There are other freely available backup systems that you might want to have a look at, including the "Ohio State" backup software [*Ohio*].

And as for commercial software, there is a wide variety to choose from—pick up any industry magazine and check the ads, or look for a recent article comparing different backup software. The commercial products typically provide a GUI management interface, online indexing of files, user-initiated restores, support for more types of hardware, etc. The commercial software seems to vary in features, with different OSes that are supported, hardware support, security, polish, and price. Most commercial packages have reasonable scheduling capabilities. Some use "standard" tape formats (such as tar- or dump-compatible formats), and some use proprietary formats. Have a look and see what fits your needs best, considering such factors as:

- hardware and operating systems supported
- media and jukebox types supported
- ability to handle your organization's expected special needs, such as restrictive backup time windows, expected number of restore requests, tape duplication requirements, and so on
- ease of use, for backup, recovery, restore, and for user-initiated restore
- perceived or expected security of the implementation and how closely it matches your environment

- scalability to large numbers of machines or sites
- ability to "stage" backups to disk, or multiplex multiple backup streams, in order to keep your tape media "streaming"[3] without stopping and starting
- vendor reputation and availability
- and, of course, last but almost never least, price

### *7.4.3 Live Versus Offline*

And while we're talking about software, let's talk about "live" versus "offline" backups. By that I mean, do you run your backups against a machine running in normal "time sharing" mode, or do you shut down to "single user" mode, and kill off unneeded processes to ensure that nothing changes on a file system while you're backing it up? The answer is, of course, that "it depends" (and yes, I know that that's a cop out). If you can identify a time of day when your systems are likely to be lightly loaded, and your backup software can handle file system changes in a "reasonable" fashion, you'll likely want to do live backups and leave your systems up and running while they're being backed up.[4]

Even if you're doing live backups, you may also be faced with limited time to complete your backups (your "backup window"). Time limitations can have substantial impact on your software (and hardware) limitations, as well as on your backup schedules and on media handling and retention.

### *7.4.4 Special Needs and Abilities*

There are some special backup situations that I'm not going to cover here, or even mention in any detail. These include:

- large data collections, such as data warehouses
- windows machines and other desktops
- special purpose environments, such as student labs or server farms

I will mention two more complicated backup situations in a little more detail: databases and file systems or servers with special abilities.

Database backups are often more interesting to deal with for a few reasons:

- They are often large, or at least non-trivial in size.
- Databases often need to be available virtually non-stop, 24x7.
- They often have many large, constantly changing, and interdependent files.
- They may make use of special operating system abilities, such as special files or advanced file systems, or they may use raw disk devices or partitions rather than files in a file system.

If you can't shut down or interrupt your database to do backups, or if you can't dump or export your running database into an internally consistent set of operating system files, or if your database uses raw devices or partitions, you may be best served

3. Streaming is when your tape devices are kept running with a full input stream while writing. If the input stream is not kept full, most tape devices will start and stop every little while, wasting both tape capacity and speed, and usually providing more wear and tear on your tape drives too.

4. You may wish to consider a full, offline backup before you do OS upgrades or hardware changes; if something goes wrong, it can be very comforting to know that you've got a nice safe backup nearby.

by commercial backup software that integrates and interacts directly with your database software. Most commercial backup software vendors offer software modules to deal directly with most major database software.

Some file system software, some RAID subsystems, and some network file servers offer special features designed to ease the pain of backups. Some of these features are quite advanced and more than a little complicated, but if your needs are non-trivial, you may find that to be very helpful.

> Final note on database backups: if you're not the database administrator (DBA), get him or her involved in your backup system selection and testing. You and your organization will be glad that you did.

A very useful feature is what is known as "snapshots." A snapshot, as the name implies, is a "copy" of a file system or directory at a particular point in time. A snapshot is usually created almost instantly, with minimal disruption to the programs, users or systems that happen to be using it at that point. I put quotes around the word "copy" above to suggest that a snapshot is not usually implemented as a separate copy of all the files in a file system or directory—it just appears that way. Manufacturers make use of techniques such as "copy on write" and inode redirection to implement snapshots—ask your vendor for details of their implementation if you're interested.

The benefit of snapshots for backups is that, properly configured and used, they allow the backup software to see what appears to be a pristine, unchanging, self-consistent file system, just waiting to be backed up at the software's leisure.

One last file- or disk-related feature that can be useful for backups, if sometimes cumbersome, is mirrored disks or directories (section 2.8). One can sometimes "break" a mirror, thereby leaving one side of the mirror intact, online, and updatable and the other unchanging but available for the backup software to read and back up. Once the backup is complete, one would simply arrange for the mirror to be "re-silvered" and put back into production use. The re-silvering process can take anywhere from a few minutes to a few days, depending on how much data changed while the mirror was broken and how smart the mirroring software is. Note that this can also be accomplished with "three-way mirrors," which contain three copies of that data, so that when you take one set offline for backups, you still have the benefit of mirroring for your data's safety.

### *7.4.5 Software Reliability*

As for software reliability, it's pretty much the automation, ease of use, and the robustness of the software that you choose that are the relevant issues. You'll want to automate as much as possible, but with something as important as backups, you'll want to have positive confirmation that your backups are running properly (by reviewing logs, status mail messages, and so on) on a daily basis.

## 7.5 Hardware

The hardware that you choose for doing your backups affects your ease of use, media cost and reliability, and ease of replacement in case of failure.

Most backups are made to some form of magnetic tape. In the old days, we relied on good old nine track reel-to-reel tape, but there aren't many people investing in that technology these days. In the UNIX environment, two of the most common tape formats are 8mm (popularized by Exabyte) and DLT, with a variety of other types (such as DAT and AIT) also in use. A comparison of tape formats is more than I want to get into here, but I will mention a few things for you to consider when choosing a tape format:

**Reliability and Intended Use**
: Was the tape format developed primarily for data use, or is it a format that was originally developed for audio or video use? How will it stand up to your expected usage?

**Capacity**
: Do you need a high capacity format, or is a lower capacity (e.g., only a few gigabytes) enough to meet your needs?

**Speed**
: Some tape formats are inherently faster or slower than others, or optimized for different types of use. And a tape technology that will stream on your system is usually better than one that won't.

**Media Cost and Availability**
: How much does each tape cost, and how easy are they to obtain? Is it important to be able to go down to the local consumer electronics store when you run out of tapes some evening?

**Drive Durability and Availability**
: How well will the drives stand up to your expected duty cycle? Will you wear out the drives on a regular basis? Will you need quick replacements? And what's the warranty and service contract like?

**Compatibility**
: How does the format fit in with what's already in use in your organization? If everyone else is using a particular format, you might be better off to conform, so that you can borrow media, drives, and expertise if and when you need them.

For more information on tape formats, see [*Preston99*] or [*BackupCentral*].

The next consideration is aggregation and automation. By this I mean the choice between single tape drives, and tape jukeboxes or auto-changers (of small or large capacity). This depends a lot on how much data you need to back up, how long you want to keep it available, how often you expect to need to restore data, and how you expect your needs to change in the future. In many cases, the extra cost of a small jukebox (10 tapes or so) will be well worth it in terms of ease of use.

Closely related to aggregation and automation is the question of how many drives you need (or want) to have. How much capacity do you need? Do you need multiple drives running in parallel in order to get all your backups done in the time available? Do you need to be able to duplicate your tapes to guard against media failure or to take offsite? Do you want to be able to do restores on one drive while doing backups on the other? And finally, if you do choose a jukebox, will the tape drives still be usable when the tape changing mechanism breaks (as it almost certainly will sometime)?

When the University of Waterloo was looking for a new backup system in 1996, it ended up choosing a DLT jukebox with two drives and room for 250 tapes, with the ability to expand both the number of drives (to 10) and the number of tapes (to 600) (the university actually bought two of them). This may seem like a giant system to some, but it's actually only a mid-size in the world of backups, and in retrospect, it seems to have been a good choice for the university.

And finally, you'll need a machine to run your tape drives. Choose a machine that fits with your other machines, and consider dedicating it to the task. It may seem a waste for a nice machine to be sitting idle all day, just to wake up and write a few tapes in the middle of the night, but for something as important as backups, it's often nice to have a secure, limited access machine that you can dedicate to the process. It can also sometimes make restores much easier. Make sure that your backup machine has enough local disk space to handle staged backups or restores, and that it has enough power and I/O bandwidth to keep the tape drives streaming while writing backups.

Those of you who have been paying attention will have noticed that I didn't mention other media, such as magnetic or optical disk. I'll contend that those alternatives are only appropriate for backups in very special situations, and that you'll already know if they are something that you should consider in your environment.

## 7.6 Physical Location

Where are you going to locate your backup server? Does it need to be physically close to your desk? Near your servers? In a nice locked room with fire suppression gear? Do you need easy access to it to swap out tapes? What does your network look like—do you have adequate bandwidth for your backups in more than one place?

I mention these questions to get you thinking about the physical and network security of your backups and backup system. Remember that a backup system makes a nice attack target, since it will contain all your data. What happens when you have a fire in your machine room and all your servers, including the backup server (and the tapes), melt? And don't forget about a nice UPS for your backup system—you may not be able to do any backups during a power failure (since your other machines or networks may very well be unavailable), but at least your backup server won't get corrupted by a sudden power outage.

When we bought those jukeboxes in 1996, we were able to put one in a building across campus that didn't already contain any machines of interest. We dedicated a pair of fibers to a fast ethernet connection, built a small air-conditioned room, installed an intruder alarm, and locked the backup server and jukebox in there. That way, we ended up with offsite backups, without having to remember to move tapes about.

## 7.7 Media Handling

The main consideration for media handling is how to get your tapes offsite, and how to get your duplicated/cloned tapes into a different location than the originals. Many people overlook the need to get their backups physically away from the original disks: a fire, flood, or fire-ax-wielding computer hater, and you could be out of business.

I'll make my point with a short story, wherein I learned the necessity of offsite backups. I used to do some programming for a university professor, on a PC in his office, as part of a major, multi-year, externally funded research project. I came in one day, and the IBM PC AT (it was a long time ago) was gone, along with every 5 1/4-inch diskette in the office, including the backups. Fortunately, we had another set of backup diskettes that were fairly recent, offsite at the professor's home. Without those, we would have been in major trouble. (Most people learn a "backup lesson" at some point—I'm just lucky that mine was more abstract than most.)

In some areas, there are companies that provide services for transferring your backup tapes offsite to secure storage facilities. Regardless of the cost of such a service, it's probably better to pay someone than to continue cycling the backup tapes out of the machine room and into the secure offsite storage facility that is really just the back seat of your car.

## 7.8 Testing, and Restore and Recovery Practice

The classic UNIX backup horror story involves multiple file systems being backed up to a single tape, a hapless system administrator who accidentally specified the rewind instead of the non-rewind tape device, and a company president who just accidentally deleted a very important file.

There are two main reasons for testing your backup system.

The first is to ensure that you're actually creating good backups, and that you can restore from them, and that you're backing up the files and directories that you actually intended to back up. Write a script to step through your tapes to check the dump headers on each file and generate a report. Pick some files at random from various machines and make sure that you can find them on your backups.

The second reason for testing the restore and recovery practice is to ensure that, when the emergency comes, you will know what to do and how to do it. When the root disk on your main central server gives up the ghost, make sure that you can rebuild it from your backups. This also helps ensure that your standardized backup process leaves your backups available when they're needed.

This is a convenient place to note that sometimes commercial products that generate backups in "native" formats are a real blessing. Many operating systems let you easily restore a dump file onto a new blank disk. But if you're using backup software with a proprietary tape format, you may have to do a complete OS installation, install the backup software, and only then start doing the actual restore. (I'll point out that it's convenient to be able to attach a new disk to some other running machine, do the restore there, and then install the new disk in the broken machine.)

# 8 Disaster Recovery Planning

Let's discuss some aspects of disasters, how to avoid becoming too much of a victim, and how to put things back together should your avoidance measures prove to be inadequate. I'm going to review things primarily from a system administration standpoint, but it's important to remember that disaster recovery and avoidance is a far larger topic.

Computing professionals like us typically look at disaster recovery planning (DRP) primarily from a computing systems point of view, which is only natural when you consider which budget pool we're paid out of. Let's look at the big picture and hope that that will help put the system administration issues into perspective.

What kinds of disasters might befall an organization?

- the obvious ones: hurricane, flood, earthquake, explosion, and so on
- a tragic plane crash while all key staff are flying to a much-deserved offsite tropical "retreat"
- primary product found to cause cancer in every living mammal except laboratory mice
- armed insurrection
- complete breakdown of municipal transit systems, preventing staff from getting to the office
- massive chemical spill and fire with toxic fumes at the company's plant, resulting in mass evacuations, health and environmental concerns, and virtually unlimited personal liability for the senior management and directors
- disgruntled key employees start competing company, lure away every employee with a non-zero IQ
- keys to accounts receivable filing cabinets lost, leading to billion-dollar write-offs

See? There's a lot more to DRP than making sure that the computers are running and the printers are printing. That said, let's concentrate on DRP for computing systems.

As I've tried to get across in other chapters, the key to appropriate levels of reliability in system administration is the balancing of the exposure to and costs of risks with the costs of avoiding those risks. What's the worst that can happen (typically)? The company goes out of business, everyone is unemployed and without a pension, and the boss goes to jail. There's a story, which is probably apocryphal, of a mid-level executive who was charged with disaster recovery planning for his organization. His DRP? Keep an up-to-date copy of his resume at home. Most of us, however, would probably prefer

to have at least something in place to provide some level of protection and recoverability.

Let's try and divide the problem space into three major areas:

- major physical damage to computing hardware or communications infrastructure
- utility (power, HVAC, telecommunications) failures
- physical inaccessibility due to weather, structural damage to the building, civil unrest, or evacuation (due to chemical spill, fire, etc.)

For each of these three areas, I'll try to outline some of the reasons you might want to protect yourself against the risks, ways in which you might try to avoid the problems, and some ways you might be able to recover if you should be affected by this kind of problem. Following that we'll briefly consider some general methods for recovery.

Remember—the key, as always, is planning and documentation (on paper, both on- and offsite). Leaving your disaster recovery planning until disaster strikes only increases the severity of your disaster.

You'll note that redundant remote systems or locations can help deal with these kinds of problems—I'll cover some of the issues that raises a little bit later.

## 8.1 Physical Damage

If your entire computing infrastructure consists of a single clone PC, a modem, and a cheap printer, it's probably not worthwhile worrying too much about protecting your equipment from loss or damage—if something gets damaged, just go to any of the consumer electronics stores in your area and get a replacement off the shelf.[1] If, however, your equipment is not typically available at the mall on a Saturday afternoon, you probably want to consider how to limit your potential damage and how to get access to replacement equipment in a timely fashion.

Physical damage to your computing and communications equipment can happen in a number of ways. Two of the most obvious ways in which equipment can be damaged are fire and water, but there are a number of other possibilities that you might consider protecting against. Let's review some of the possibilities for damage.

**Fire and Smoke**

The best protection against fire and smoke is a safe, fire code compliant building, and an appropriate fire detection and suppression system. In past years, Halon was widely used as a fire suppression agent in computer rooms, but it was not environmentally friendly. Fire suppression systems are currently available based on carbon dioxide and other chemicals, but water-based sprinkler systems are still the most common suppression method. Note that "pre-action" sprinkler systems are available in many areas, which provide a remote sprinkler valve and normally empty sprinkler pipes, protecting you from plumbing failures in the sprinkler system.

You should also consider the use of an emergency power-off system, to power down

1. But make sure that your backups are up-to-date and protected offsite!

your systems in the event of an alarm. Among other things, this will help avoid damage to your equipment from smoke and residue being drawn into the chassis through the cooling fans.

**Water**

There are a few ways for water to attack your equipment—plumbing failure is probably the most common, but you should also worry about water damage from fire suppression systems or flooding. You should consider two primary attacks from water—falling down from above, and seeping up from below.

From above, there are burstable pipes (both in your premises, and feeding the bathtub or dishwasher in the unit above) and fire hoses. For pipes in your computer room, I've seen some installations with drainage trays mounted under the pipes, draining off to the side of the room. And if you have control (or knowledge) of whatever it is in the rooms above you, you might want to worry about whatever plumbing there is up there. Otherwise, your best protection is to keep your equipment in racks or cabinets with a roof over them (and make sure that the ventilation fan outlet isn't in the middle of the top of the cabinet).

From below, consider a raised floor, with in-floor drains (including backflow prevention valves), pedestals or some other device to keep your electrical connections off a potentially wet floor, and an alarm system to warn you when it gets wet. And if your computer room is below grade, you may wish to reconsider—the farther you are above the water table, the safer you are. Unfortunately, raised floors are not always a good idea in earthquake zones, due to the risk that the floor might collapse during an earthquake.

**Earthquake, Tornado**

There are two things to consider for these problems: the integrity of your building, and equipment safety. If you are located in an area that is at risk of earthquakes or tornados, consider how your building would be affected if an earthquake or tornado hits. Consider which parts of your building are most likely to be damaged—large plate glass windows, overhangs, trailer parks—and try to locate your equipment as far away from there as possible.

For your equipment, consider bolting it down in some appropriate fashion, and don't forget to fasten your rolling equipment racks down too. No sense having your equipment bounce across the room or fall out the window every time there's a tremor.

**Vandalism**

Depending on your industry and location, you may wish to consider the threat to your equipment posed by vandalism, looting, revenge, or a disgruntled employee. Is your computer room unlocked? Do you have big glass display windows to impress random strangers? Do you store a selection of fire axes in and near your computer room?

Alternatively, are you careful to collect keys and change security codes when an employee leaves (or is pushed)? Can employees enter on their own, or are two people required to act together to gain access to the computer room? Do you have 24x7 physical security monitoring?

If you have a non-trivial computer room, one of the most important things you should do is consult a local expert who can advise you on the most appropriate protection in your area and for your situation.

How can you attempt to recover from physical damage? The classic answer is to have a redundant offsite installation that can be used for recovery (section 8.4). Alternatively, consider such options as:

- emergency recovery agreements with key suppliers
- strategically selected and located spares
- planning for what processes and activities (if any) can be performed manually while computing system recovery is underway

Understand your situation, review your exposures, and make and implement your plans to ensure that you're ready if and when disaster strikes.

## 8.2 Utility Failures

Just about everyone is in a position to be affected by some form of public utility failure—the most obvious being electrical power failure. Even if you generate your own electricity with wind turbines and backup batteries, you're still at risk of extended calm or physical failure of your generating equipment. Most people can survive short outages on an occasional basis.

But if you're in an area where utilities can be unreliable (poor infrastructure, frequent thunderstorms, etc.), or if you worry about extended outages such as those suffered in Quebec and New England due to the 1998 ice storms (some places were without electricity for several weeks), you may wish to consider some suitable form of backup or redundancy for your utilities.

**Electricity**

Most of us rely on electricity from the local power company. The obvious way to protect yourself against outages is through the use of an uninterruptible power supply (UPS) with a diesel generator for backup and extended outages.

But it's important to remember that in a disaster, you won't just be worried about your central computing systems. You'll need power to run heating or cooling equipment, ventilation, at least some room lighting, your telephone switch, and so on. This isn't just a system administration issue, it's a facilities-wide issue.

In any case, don't skimp on maintenance. Test your backup power systems regularly, under load, keep your fuel tanks full, and consult an expert. If it's important enough to you, look to an "N+1" system to guard against crucial "single point of failure" equipment failures.

**Water**

From a system administration perspective, the primary use for water is in air-conditioning equipment. In some situations a reservoir or cistern could provide spare water during an outage, and there are water tanker trucks sometimes available. Otherwise, hope for a cool spell.

**Gas, Oil, Propane**

Again, primarily for environmental control. And, fortunately, alternate heat sources are often available, even if you have to resort to electric space heaters.

**Communication Links**

Most of us rely on some form of communications, whether it's ordinary telephone connections, leased lines, fiber, or various forms of wireless communications (though it's probably safe to say that wireless use is in the minority). Most of us rely on these links as a regular part of our work day, and for some of us, the business stops when the communication links go down.

The best way to protect your communication links is through the use of redundant connections. For Internet connectivity, many organizations are "dual homed" to two providers, and prudent organizations make a point of ordering links from multiple carriers (and ensure that the carriers don't simply buy capacity from each other). Even if your redundant links leave your premises through different paths over different carriers, it's still possible for them to terminate or pass through the same carrier central office, which does limit your redundancy.

If you're provisioning multiple links, try to get specific physical routes from your carriers so that you'll have a better idea of whether or not you have route diversity and where your exposures are.

> Don't just blindly accept what your carriers tell you about circuit routing—get whatever assurances you can, by whatever means you can manage. Carriers, intentionally or not, don't always tell you the exact truth about your circuits, and they tend to change things as time goes by, so that previously diverse connections can easily end up in the same cable bundles.

One alternative for redundancy or backup that is becoming more common and more feasible is the use of metropolitan area wireless communications and/or the use of satellite links. A satellite link, while typically slower and more expensive (or, at least, not cheap) provides nice redundancy as it can allow you to isolate your communications from any local problems. Of course, if all your connections are to systems in the same area as your office, remote satellite connectivity might not help too much. (Of course, this is where I remind you all of the 1998 satellite outage which disabled huge numbers of pagers, and the difficulty of dealing with failures in your backup communication systems.)

## 8.3 Physical Inaccessibility

I'll again dredge up the ice storms in Quebec and New England in the winter of 1998 as an example of how your office can be just fine, but it's just not possible to get there. Other examples are the result of earthquake, flood, trucker blockades on European highways, bombs in the World Trade Center, and major parades.

If your business relies on physical access (e.g., a printing or a warehousing company), you could be in trouble. If you're an organization that deals in knowledge or computing, you may be better off. An easy way to deal with the latter is to ensure that your

staff has home computing and an account on a reliable ISP (or run your own remote access servers, with lots of capacity for emergencies), and just have them dial in for the duration. Voice mail, remote phone forwarding, cell phones, and pagers all help limit (if you're lucky) the impact of this kind of "disaster." But make sure that your security infrastructure can provide appropriate security for all those remote users.

## 8.4 Redundant Premises

The classic disaster recovery plan (for computing and communications at least) involves a redundant recovery site just sitting and waiting for something to go wrong—this is still common in the mainframe world, where large data processing capacity is needed on an ongoing basis. If you absolutely need ongoing computing, an alternate site is likely going to be part of your plan. Even if your needs are much simpler, you can benefit from some forms of offsite redundancy.

### Standby Sites

In the traditional case, a large, climate controlled, raised floor computing center is loaded up with millions of dollars of equipment that sits there idle waiting for something to go wrong somewhere else. These sites have taken a number of forms—one of the most common is to be run by a service company, providing backup services to a number of clients, but some large organizations have backup computing centers that are dedicated to them. It is also not unheard of for system vendors to offer recovery services for their customers, and some co-operative ventures also exist for their members' mutual benefit. This typically isn't a cheap method of protection, but if you need it, you need it.

### Distributed Sites

What is much more feasible, and much more practical in these days of high-speed Internet connectivity, is the use of distributed computing sites that can provide backup for each other in the event of a "disaster." An obvious example is the use of Web server hosting at service providers using some form of load sharing across multiple servers.

This idea can be expanded by distributing your primary computing resources across multiple sites, taking care that you have similar equipment at each site. This kind of distribution also makes it possible to automatically store your backups offsite (given large enough bandwidth). However, it is harder to justify distributing your computing when your staff is all located in one place.

## 8.5 Recovering

I mentioned it before, as many others have before me, but I'll reiterate that the key to successful recovery is a proper plan and proper documentation. Space limits restrict how much I can say here, and I will refer you to the bookstores for DRP books, but I will mention a few points.

### Hardware

Is your recovery hardware compatible? Do you have documentation of what makes your systems and installations unique?

**Backups**

You have them, of course, and they are offsite, of course, but do you have the index to the media that will allow you to find the necessary backups when you need them? Which are the most important and which need to be restored first?

**Names and Addresses**

Do you know what your machines should be named and numbered? Do you know what your network topology should look like?

**People**

Do you know how to get in touch with your staff at home, or is your only phone list the office extension numbers on your desktop machine that was just destroyed in the fire? Do you have a list of who to contact first and of who should do what? And in what order?

**Communications and Connectivity**

Do you know who to call at your service providers and carriers to get your connectivity adjusted for your new or temporary location?

**Status**

Do you know who is in charge of the entire recovery operation for your organization, and how to get in touch with them and when? And what to do if you can't get in touch with them?

**Food**

And finally, do you have a list of food delivery places at your recovery site so that you won't pass out from lack of sustenance while working feverishly to put things back together?

## 8.6 In Summary

This chapter has provided an overview of disaster recovery planning, covering some of the things you need to consider and some of the problems you need to be prepared to address. The underlying message is: plan, prepare, practice, and pray you never need it.

# IV *People*

## 9 What About Yourself?

In other chapters, I've talked about general principles of reliability, computing hardware, networking, and some aspects of system administration. Most of those things are really quite tangible—if you can't put your hands on them physically, you can at least copy them to a printer or a tape drive, and hold them in your hands that way.

When I was at the 11th LISA Systems Administration conference in late 1997, I noticed that we spent a lot of time (more so than usual) talking about management, motivation, and people issues. Since that time, I've found myself doing much more reading on management and people, and thinking more about the people issues that we face in our jobs (and other activities). And in the past few years, I've spent a heck of a lot of time in meetings, and working with people, and thinking about motivation, and coordination, and how people can really enjoy their work.

And so, at one point, I found myself sitting with my laptop on my daily commute on the inter-city bus, with the Christmas holidays and a new year looming up before me, writing about reliability of a different flavour. I was compelled to consider, from a purely amateur point of view, personal reliability. In this chapter I mean to consider how we interact with our co-workers, vendors, and customers, and, to a lesser extent, our friends and families. How do we act "reliably"?

### 9.1 System Administrator Reliability

Now, I'll pause here, and beg your indulgence. How is personal reliability relevant to system administrators and computing professionals in general? How does it help to make our computer systems and networks run better and more effectively? System administration is very closely tied to personal interaction, with individuals and with groups, and sometimes with people that you will never see or talk to directly. I'll try to give a few examples of why I think that that is the case and why reliability and trust are important.

System administration is a service activity—we supply the computing resources so that other people can do their work (or play). We solve problems for people, we design systems and software to serve people, and we help people learn to accomplish their computing tasks in the most effective ways. Any time we install a new command, send out a notice or advisory message, or answer the phone on the help desk, the underlying end product is (almost always) a service for some person or group.

When we take a system down for maintenance, submit a request for more funding for yet more equipment, design a mission critical computing environment, start fixing a computer or network problem, or propose a solution to suit someone's needs, we're asking for trust. Trust that we are using good judgment, trust that we are knowledge-

able and competent, and trust that our intentions are good. In short (and I'm sure you've been waiting for this), we are asking others to rely on us. And that's where personal reliability enters into the equation.

Why is it important to be reliable? Quite simply, if we are to call ourselves "professionals," we (and others) must be able to rely on our reputations, and the most important part of a (positive) reputation is the trust that people can place in us, our judgment, and our abilities. If we cannot be relied upon, all of our experience and abilities will be far less valuable to our customers and co-workers. The ability of others to rely on us is the foundation of the value that we bring to the profession of system administration.

## 9.2 Personal Reliability

How do you demonstrate your reliability? How do you earn the trust of your constituents?

I think the most important piece of advice that I can offer is to avoid the "us versus them" mentality that we see (or hear about) all too often. Recognize that you and your users are (or should be) working towards the same goals, and towards the success of your enterprise or activity. While the goals and needs of different groups often seem to be at odds, a little goodwill and effort to understand will make it far easier to work together towards the best solutions. Additionally, try not to think of the people that use your systems only as "users"—think of them as your customers, your clients, and the reason you have a job.

Consider the other people in your organization and consider your customers—work to understand their concerns and needs. System administration is not done in a vacuum—a system is only as worthwhile as the services and solutions that it provides. A beautiful, carefully designed, "perfect" computing system is useless if it is conceptually pure but unsuited to solving the problems at hand.

When interacting with customers or others in your organization, be honest and open. If there's a problem, admit it, and if it's as a result of something you did (or didn't), own up to it and take responsibility. Any short-term pain will be far outweighed by the long-term gain, as your users trust and rely on you. Say what the problem is (or was) and what you've done to keep it from happening again. Give advance warning when you're about to change something, and be realistic about expected downtimes. Remember to follow through—do what you said you would do, when you said you would do it. And finally, be proactive, talk with your users, solicit their feedback and concerns, and act on them. Earn their trust, and you'll be far better off in the long run.

> If the word "lusers" is a part of your vocabulary, you might want to reconsider your use of it.

## 9.3 Reliability as a Leader

On the other hand, if you're a manager or leader of system administrators, how well can the people in your group rely on you?

Are you supportive, understanding, fair? Do you send people home when they are sick, or do you tell them to "tough it out"? Are you consistent? Do you keep your

word? Do you reward and praise significant accomplishments? Do you pitch in when the crunch comes?

Are you an advocate for your co-workers? Do you defend them if they're being attacked (deservedly or not)? Do you champion them in interactions with other groups and higher-ups? Do you fight for appropriate conference and training budgets, and for extra pay or compensatory time when they work overtime?

Can the people in your group rely on you?

Please allow me to offer some words from Dee Hock, Founder and CEO Emeritus of Visa: "If you don't understand that you work for your mislabeled 'subordinates,' then you know nothing of leadership. You know only tyranny."

## 9.4 Relationships

When I started in system administration many years ago, I spent a lot more time concentrating on my "relationship" with the machines than I did on my relationships with the people who used them. I eventually found out that the machines really didn't care whether I was reliable or not, just so long as I kept the AC power coming, the backup tapes loaded, and the ethernet link light lit.

These days, I spend a lot less time dealing with machines and a lot more time dealing with the people who surround those machines. I've learned which of those relationships is the more complicated and the more rewarding, and I've learned where the true value and the true satisfaction lie.

# 10 You and Your Users

In this chapter, let's discuss user community interaction and how it relates to reliability. As we discussed in chapter 9, as system administrators we are providing services, and users are, of course, the reason we provide those services. How can we use our user interaction to improve reliability, communicate that increased reliability is one of our goals, and help our users become part of the reliability equation? Let's look at the question three ways: communication to users; communication from users; and education, training, and publications.

## 10.1 Communication to Users

A long time ago, I learned that there are a number of situations in which it is not sufficient to simply do your job—you must also be seen to be doing your job. By that I mean that sometimes you must be obvious about what you are doing (and why) while you're doing it. Consider, for example, a security guard—the fact of being visible can in itself act as a deterrent and reduce the likelihood of "an incident."

I'll claim that reliable system administration is another one of those tasks that is enhanced by visibility. For example, if a system is obviously being run in an organized and disciplined manner, are users more likely to act that way themselves, and thereby bring us closer to our goals? How can we be obvious about what we're doing, and why, and how can users help? And I'll claim that a documented, predictable computing environment will be thought of as far more reliable than it might otherwise.

The standard method for successful presentations is to tell your audience what you're about to tell them, tell them, and then tell them what you've told them. I think there's a convenient parallel for communication to your user community:

- Tell them what you expect to do for them and what you expect from them.
- Follow through and work towards your commitments, keeping your user community informed of your progress (or lack thereof) and any failures or incidents that might occur.
- Provide statistics, incident reports, and plans for improvement as you progress.

### *10.1.1 Tell Ya What I'm Gonna Do*

Advance communications are often going to be the largest component of your "formal" user communication, and are likely to tend towards the "static" rather than "dynamic."

#### 10.1.1.1 Service Offerings

You should outline (in greater or lesser detail as your situation demands) what services you will (plan to) provide, such as centralized file service, printing, user consulting,

authentication services, and so on. The list of services will presumably have been arrived at through a process of user consultation, executive fiat, or divine inspiration (or a combination of all three), tempered by your experience and expertise and suggestions on what might be most appropriate and useful in your organization. These are your "service offerings."

#### 10.1.1.2 Service Level Agreements

Next, you should document your goals for performance and availability, both in terms of machine and network performance and availability and in terms of guaranteed response and repair times. These are your "service level agreements."

For machines and networks, you would typically look to such metrics as percentage uptime, response times, network latency guarantees, and other similar measures. For example, you could state that your network file server will be unavailable less than an hour a month (99.9% uptime),[1] that round-trip packet times between major points on your network will be less than 10ms, or that there will always be at least 5GB of available disk space (e.g., for image processing).

For response and repair times, some examples might be next-business-day response for installing new personal network connections, accounts created within three hours, file restores within eight hours, first response to problem reports within two hours, and so on.

The important thing here is to make sure that your goals and guarantees are aligned with the business needs of your organization—figure out what services and activities are most important to your users, and then determine how you can organize your resources (both human and machine) to best balance those needs in delivering your services.

#### 10.1.1.3 Policies and Practices

The last component in setting expectations is policies and practices. There are several reasons to establish service and usage policies, including conformance with and the ability to serve organizational goals (by reserving CPU capacity for product developers working to get the next release out the door, for example), making it possible to meet service level agreements (by, say, reserving two hours on Sunday so that maintenance doesn't interrupt activities during the week), and ensuring that users aren't interfering with services to others (for example, no network-based Quake playing during office hours).

As a service provider, you will likely be better off if you proactively define the polices and practices you will use to deliver your services. That is, it's better to define your own goals before some other less "reasonable" goals are imposed on you. You will typically want to consider such things as:

- standard change/maintenance windows (e.g., Saturday mornings, Tuesday nights from 11 p.m. until 3 a.m., etc.)
- emergency repair procedures, and what constitutes an "emergency"
- standard operating hours for services such as the help desk, hardware repairs, and so on

1. Scheduled maintenance windows and other planned outages are usually excluded from "availability" calculations.

- a policy for off-hours response, and a method of deciding what can wait and what has to be fixed immediately
- a method for responding to unexpected "incidents," security-related and otherwise
- escalation procedures, in case goals aren't met or problems are more substantial than they first appeared

You will also likely want to outline possible remedies or repercussions for those times when you fail to meet your stated goals and guarantees. For an internal service organization, these are more likely to involve public "humiliation," bad performance reviews, or lousy parking spots in the company lot. For an external service provider (such as an ISP or a computing service bureau), the result of a failure to meet the goals and guarantees is likely to involve money.

Finally, you will need to outline the policies that your user community is expected to follow. These policies contribute to reliability by (hopefully) freeing you from having to deal with malicious or unexpected acts by your users and establish a base for understanding between the two groups. The better the understanding between users and service providers, the easier it is to provide a reliable and understood service. You will likely want to cover such areas as:

- security, password sharing, snooping, sniffing, hacking, and other similar activities
- virus protection, and under what circumstances it is acceptable to add software to your systems and networks
- protection of the organization's equipment (such as no latte with more than three sugars is allowed within two feet of a keyboard)
- disk space limits, CPU hogging, and other resource consumptive areas of contention
- what may or may not be connected to the network, and where (inside or outside the firewall, which subnet, and so on)

### *10.1.2 Change, Status, and Failure Reports*

These reports should be part of your ongoing, day-to-day communications with your user community. The two most important attributes of these reports are timeliness and completeness—you need to provide the information that your users want, need, or deserve at the most appropriate time.

#### 10.1.2.1 Change Reports

Change reports are those that outline a planned change to your systems or networks—its impact (or even better, lack of impact) on your users—and that summarize or confirm its implementation. These reports are important in allowing your users to plan their activities around expected service disruptions and to understand and prepare for changes to software or interfaces.

Reports would typically outline the expected date and time of the change, its impact, and how to obtain additional information. Internally, you should also docu-

ment (in advance) how to implement that change, how to test the implementation, and most importantly, how to back out the change if necessary. Change reports should be issued far enough in advance that users have adequate preparation and warning time, but not so far in advance that they get forgotten.

#### 10.1.2.2 Status Reports

Status reports are the ongoing mechanism by which your services can be measured for trend analysis and to allow for evaluation against your service level agreements. You would typically track as many of your defined metrics as possible, in log files, graphs, printed reports, Web pages, summary numbers, etc.

I'll recommend MRTG [*Oetiker98*] and its successor Cricket [*Allen99*] as terrific ways to graphically track virtually anything against the calendar—network traffic, uptime, users, disk space, news or mail traffic, routing table size, numbers of outstanding trouble tickets, and on and on. If you've got the time and you can automate the collection and reporting (and it won't adversely affect your systems), track as many metrics as you can think of, even if you don't publish them all. The more information you have, the easier your troubleshooting and capacity planning will be.

It's common in customer service organizations to track various service metrics (time to resolution, phone queue time, calls per hour, etc.). Many (most?) system administration organizations could benefit from more proactive status reporting.

#### 10.1.2.3 Failure Reports

Failure reports are, obviously, something that we would all like to keep to a minimum. They are the method by which you report system or network "incidents" to your customers and (hopefully) document your plans to ensure that such failures are reduced or eliminated in the future. They can also serve as a symbolic way for you to "take responsibility" for an outage and demonstrate your commitment to improving your service.

Don't feel you need to wait until an outage is resolved to issue a failure report—your users will appreciate your openness and consideration, and will also be less likely to contact the help desk about expected up times if you've already posted bulletins. The latter can be quite useful if you're a small organization and the people responsible for answering the phones are the same people that are busily trying to fix the problem.

### *10.1.3 Statistics, History, and Revisiting the Future*

This is where analysis and prediction come into play. Using the information that you have gathered, and projected changes in usage, information on new projects, and normal usage increases—you can generate current and historical statistics, identify trends, and predict the future. And when you've done it once, you will also be able to use it to re-predict the future, and compare your predictions to what actually happened.

The capacity planning uses of this information are obvious, but these reports are also a good way to demonstrate your professionalism to your users (and your management!).

## 10.2 Communication Initiated by Users

In order to be able to correct problems, deal with "incidents," and allow for some ongoing improvement of your processes, you need to provide ways for your user community to get you the information you need (and the information that they need you to have). I've defined this communication as user initiated in an attempt to gently remind you that communication must be two-way—you (of course) have a duty to respond (hopefully in a timely and effective way).

I'll claim that it typically makes sense to think of two sets of user-initiated communication:

- the ongoing day-to-day problem reports, help requests, and requests for enhancement (and, if you're very good, thank you notes for a job well done)
- the periodic status reports, general reviews, direction or policy guidelines, and overall satisfaction indicators that help determine your direction and activities

The former can be expected to come from almost anyone within your organization (and sometimes from outside your organization too), while the latter are more likely to come from organizational management, steering committees, user groups, and the like, as well as your own surveys and inquiries.

The most common communication method is (of course) the telephone—most of us have probably experienced those calls out of the blue reporting some major problem, with an expectation that we will drop everything and solve it immediately. But as we review the communication process, we'll also review the alternative communication methods that you should consider and/or support.

Let's assume that you have some number of people responsible for dealing with user or customer queries (the "help desk"), and divide the communication process into three stages: the request, tracking and resolution, and your response. (I'll claim that these three stages apply to both sets of user-initiated communication, but you will of course adjust your reaction and process depending on the type of communication.)

### *10.2.1 The Request*

When someone needs help, or wishes to report a problem or request an enhancement, they need some "reporting method." Consider these alternatives as ways in which problems can be reported:

**Phone**

A "well-known" generic problem reporting phone number, which is distributed among your help desk staff in some reasonable fashion (e.g., queue, dedicated "on-call" person, round robin, whoever isn't busy). It is, of course, convenient if there is mnemonic value to the number (perhaps company extension 4357—HELP), but the important things are that it exists and that it's publicized. Note that some telephone systems can provide call statistics for your help desk calls, which is handy when you're trying to prove that you need more staff.

**Email**

A well-known email alias, such as help@company.com—most of the same considerations apply here as for telephone contacts.

**Newsgroup Postings**

A local newsgroup can be used to report problems or request enhancements in larger environments (I've seen this used in a university). This can cause the users to feel a little bit as if they're sending their problem floating off like a message in a bottle, but with prompt response and tracking it can be very effective.

A useful side effect of the newsgroup method is that it makes it possible for other users to reply, which can lessen your overall support load, at no additional cost. This is likely most effective in an environment like a university where large numbers of people (the students) tend to be enthusiastic generalists with a low per-hour cost and some amount of free time, or at least some time available for procrastination.

**Office**

A consulting office or physical help desk, where users or customers can walk in and (hopefully) get helped while they wait—this is also a great place for distributing printed documentation.

**Web Form**

The obvious method for the early 21st century, which could generate email, trouble tickets, or (potentially) even voice mail (but why would you bother?), or just about anything else. But make sure that the form is easily findable on (or from) your organization's internal Web site. And make sure that the form provides positive confirmation to the user that the report has been recorded, and a way for the user to follow up on the report if necessary.

**Hallway Chat**

Try to avoid this one, because you'll never remember all the details, you'll forget about it entirely, or something else will go wrong and get in the way of addressing the original request. (The even more problematic versions of this include the bar stool chat, the running out the door by the way, and the offhand comment while in the washroom.) I always make it a practice to say something like, "Sure, we'll get right on it—can you send some mail to 'request' summarizing the situation?"

Regardless of the method used to make the request, the better and more complete the information that you get with the request, the easier it will be to solve the problem, and the better you are at solving problems, the more reliable your systems and services will be. Consider the use of some form of problem reporting form or checklist to help improve the quality of your initial data collection.

### *10.2.2 Tracking and Resolution*

Once you receive a request, you must use some method to keep track of it to make sure that it doesn't get forgotten. Regardless of how simple or how sophisticated your tracking system is, it will probably involve the following six activities:

**Recording**

The initial information received from the user

**Delegation**

Assigning the task to someone

**Tracking and Note Taking**

During the investigation of the problem

**Resolution**
An indication that the problem or request has been fixed or addressed
**Reporting**
Notification to the user of resolution and ongoing efforts if still in progress
**Archiving**
A work record and (buzzword alert) "knowledge base" for future reference

We often don't analyze the process in such depth, but even those little pink telephone message slips can be used as the mechanism to implement those six activities. You might, however, appreciate the added features offered by even the most rudimentary automated tracking systems.[2]

The key benefits of a problem and request tracking system include:

- The user or customer will be reassured by being assigned a ticket number, and if query and status tools and messages are available, so much the better. It provides an indication that you take user/customer response seriously.
- An ongoing work record, which is useful for keeping track of changes, standard answers and fixes, and as a customer service log. It's far easier to convince management that you need more people if you have reliable statistics to back up your claim of being overworked.
- Provides a to-do list and a mechanism to ensure that nothing gets lost. But make sure that you review the list—it won't help if you just record the request and then forget all about it.
- Allows outstanding requests to be escalated for more focused attention (even if it's just your boss reviewing the two-month-old requests and asking you pointed questions).
- A conversation/interaction history, which makes it far easier to pass a task on to someone else and get them up to speed.
- Helps ensure a measure of consistency in the way you respond to requests, which will lead to more effective and efficient support.
- A training tool for staff—if they can review what others have done before them, they can do better themselves.

All of which improve your reliability.

### *10.2.3 Your Response*

I mentioned the need to report and reply to the requester, but it's worth repeating—the purpose of all this effort is to reply to the requester, providing advice, a fixed bug, a plan, a report, or even a statement or apology that you're unable to help (due to resource constraints, different areas of responsibility, and sometimes impossible problems). Your response should be timely, complete, and correct, and it's almost always worthwhile to provide interim status reports if it will take a non-trivial amount of time to respond to a request.

2. There is a modest collection of links to problem and request tracking systems at http://www.generalconcepts.com/resources/tracking/

Consider also the method you use to deliver your response. Choose between email, paper, phone, or face to face, depending on the problem, the response, and the person who made the initial request.

## 10.3 Education, Training, and Publications

One of the best ways to make yourself (or your group) more effective is to help your users and customers to help themselves whenever possible. A one-time investment in training materials (with a small amount of ongoing maintenance and revision) can provide a lasting positive result on the effectiveness of all involved (service provider and service consumer).

To more clearly tie this to reliability, a more effective user community will make fewer errors and will lead to a more reliable organization; a system administrator who has fewer user requests (because of a better user education, documentation, and training program) will be better able to deal with the systems themselves, the long-term planning, and the problem avoidance that all contribute to higher reliability.

Consider some of the following mechanisms for getting the word out:

**Email or Newsgroup Postings**

These are probably most appropriate for notices, brief announcements, and similar messages, and usually are not very effective as an educational tool. One exception I have seen (as mentioned above) is the use of a local newsgroup for posting and resolving problems, which often provides a good learning environment for the innocent bystanders reading the group for entertainment.

**UNIX Man Pages and Other Traditional Help Systems**

As more computing activity happens in a GUI workstation environment, the traditional text-based help systems and man pages are losing some relevance for end users. But these methods are still very important for system administration and other behind-the-scenes activities, and also for people working in a command line environment. And, it is easy enough to put a Web front end on traditional help text.

**Web Pages**

It's probably a fair bet that the vast majority of internal systems documentation is being put on the Web now. For client-server, general process documentation, help desk, desktop support, and the like, there's probably no better alternative.

**Local Guides**

Local booklets and user guides can be very useful, as long as you have a reasonably sized topic area that doesn't need constant updating. An example might be a user and security policies document. While you would almost certainly want to make an online version available on the Web, there's still a lot of value in printed materials.

**One Page Guides**

A number of universities and large organizations have developed very effective single page topic guides, intended to be handed out from the help desk or consulting office. These are typically designed to be almost complete references to every (local) thing you need to know, on topics such as setting up PPP, introduction to email, how to print,

and so on. When someone comes in to ask a question, you can provide the answer, and also send them away with pre-written instructions.

**Classes and Tutorials**

Of course, sometimes there is no substitute for good old fashioned face-to-face learning in a classroom or workshop setting. These are probably more effective for larger, more involved topics, or for areas in which it's important to get a fast start. A common example is when major new applications are put into place (such as a new purchasing system) and people have to be up and running almost immediately.

Any training or information mechanism will take time to put together and get going. Resist the urge to put it off for another day—if you can't find the time yourself, hire a contractor or a third-party firm, or buy pre-packaged course materials. The time you save may be your own.

# V *And Finally*

## 11 In Summary

This booklet has touched on a wide range of topics, some in depth, and some only superficially. The topic and idea of "reliability" have implications for just about everything we do as system administrators.

Like the search for quality in *Zen and the Art of Motorcycle Maintenance*, reliability can be somewhat elusive, but the end result can be very satisfying.

I hope that the preceding chapters have given you something to think about.

# Bibliography


[*Allen99*]

Jeff R. Allen, *Driving by the Rear-View Mirror: Managing a Network with Cricket*, First USENIX Conference on Network Administration, pp 1-10, April 1999, Santa Clara, CA. Cricket is a tool that lets users visualize a set of measurements over time and is a successor to MRTG. See also http://cricket.sourceforge.net/

[*Amanda*]

The FTP site ftp://ftp.cs.umd.edu/pub/amanda/ contains everything about Amanda, including copies of *The Amanda Network Backup Manager*, by James da Silva and Ólafur Gudmundsson, Seventh Systems Administration Conference (LISA '93), and *Performance of a Parallel Network Backup Manager*, by da Silva, Gudmundsson, and Daniel Mossé, 1992 Summer USENIX Technical Conference.

[*Autoconf*]

Autoconf is an extensible package of m4 macros that produce shell scripts to automatically configure software source-code packages. See http://www.gnu.org/software/autoconf/

[*BackupCentral*]

Backup Central (http://www.backupcentral.com/) is a very useful collection of information all about backups, including a number of articles and white papers.

[*Berliner90*]

Brian Berliner, CVS II: *Parallelizing Software Development*, USENIX Conference Proceedings, pp 341-352, January 1990, Washington, DC. CVS (Concurrent Versions System) is a version-control system for software development, used for many large-scale cooperative development projects. See also http://www.cyclic.com/

[*BigBrother*]

Big Brother is a Web-based systems and network monitor. See http://www.bb4.com/

[*Burgess95*]

Mark Burgess, "Cfengine: A Site Configuration Engine," *Computing Systems* vol 8 no 3, pp 309-337, Summer 1995. Cfengine, or the configuration engine, is a very high level language for building expert systems which administer and configure large computer networks. See also http://www.iu.hioslo.no/cfengine/

[*Calabrese96*]

Christopher J. Calabrese, "A Tool for Building Firewall-Router Configurations," *Computing Systems* vol 9 no 3, pp 236–253, Summer 1996.

[*Cooper92*]
Michael A. Cooper, *Overhauling Rdist for the '90s*, Sixth Systems Administration Conference (LISA '92), pp 175-188, October 19-23, Long Beach, CA. A program to maintain identical copies of files over multiple hosts. See also http://www.magnicomp.com/

[*eddie*]
Eddie, software for advanced automatic traffic management and configuration of geographically distributed server sites. See http://eddie.sourceforge.net/

[*Farmer90*]
Dan Farmer and Eugene H. Spafford, *The Cops Security Checker System*, USENIX Conference Proceedings, pp 165-170, Summer 1990, Anaheim, CA. Also available as Purdue University Technical Report CSD-TR-993. See also http://www.fish.com/cops/

[*Feldman78*]
S.I. Feldman, Make—*A Program for Maintaining Computer Programs*, 1978. Available as http://plan9.bell-labs.com/7thEdMan/vol2/make

[*HylaFAX*]
HylaFAX is terrific faxing and paging software, originally by Sam Leffler. See http://www.hylafax.org/

[*Kim94*]
Gene H. Kim and Eugene H. Spafford, *Experiences with Tripwire: Using Integrity Checkers for Intrusion Detection*, Third International Conference on System Administration, Networking and Security (SANS '94), pp 89-101, April 4-8, Washington, DC. See also ftp://info.cert.org/pub/tools/tripwire/

[*Leber98*]
Jody Leber, *Windows NT Backup & Restore*, O'Reilly & Associates, Sebastopol, CA, 1999. Out of print.

[*Monitoring*]
See http://www.generalconcepts.com/resources/monitoring/ for some additional monitoring references.

[*Muffett*]
Alec Muffet, Crack 5.0. Password analysis software. Available through http://www.users.dircon.co.uk/~crypto/

[*Nachbar86*]
Daniel Nachbar, *When Network File Systems Aren't Enough: Automatic Software Distribution Revisited*, USENIX Conference Proceedings, pp 159-171, Summer 1986, Atlanta, GA. A librarian/subscriber software distribution tool. See also ftp://ftp.cs.utoronto.ca/pub/track.README

[*Netcool*]
Netcool is a suite of related products for system and network monitoring and reporting. See http://www.micromuse.com/

[*nocol*]
NOCOL (Network Operations Center On-Line), by Vikas Aggarwal, is system and network monitoring software, with a number of interfaces and quite a few event monitors. See http://www.netplex-tech.com/software/nocol/

[*Oetiker98*]
Tobias Oetiker, MRTG—*The Multi Router Traffic Grapher*, Twelfth Systems Administration Conference (LISA '98), December 6-11. See also http://ee-staff.ethz.ch/~oetiker/webtools/mrtg/

[*Ohio*]
See ftp://ftp.cis.ohio-state.edu/pub/backup/ for the "Ohio State" backup software, including Steve Romig's paper from LISA IV.

[*OpenView*]
HP OpenView is a family of system and network management and monitoring products. See http://www.openview.hp.com/

[*Patterson87*]
David A. Patterson, Garth A. Gibson, and Randy H. Katz, *A Case for Redundant Arrays of Inexpensive Disks (RAID)*, CSD-87-391, University of California, Berkeley, 1987. Available through http://sunsite.berkeley.edu/NCSTRL/.

[*Preston99*]
W. Curtis Preston, *UNIX Backup & Recovery*, O'Reilly & Associates, Sebastopol, CA, 1999.

[*QuickPage*]
QuickPage client/server paging software by Thomas Dwyer III. See http://www.qpage.org/

[*Satdeva94*]
Bjorn Satdeva, *'Make' as a System Administration Tool*, Third International Conference on System Administration, Networking and Security (SANS '94).

[*Schemers95*]
Roland J. Schemers, III, *lbnamed: A Load Balancing Name Server in Perl*, Ninth Systems Administration Conference (LISA '95), pp 1–11, September 17-22.

[*Scotty*]
Scotty is a software package written in the Tcl scripting language that provides a high-level SNMP API. It includes the Tnm Tcl extension for access to network management information, and the Tkined network editor. See http://wwwhome.cs.utwente.nl/~schoenw/scotty/

[*Shumway91*]
Steve Shumway, *Issues in On-line Backup*, Fifth Systems Administration Conference (LISA '91), pp 81-87, San Diego.

[*SKEY*]
Bellcore, The S/KEY One-Time Password System. See ftp://ftp.bellcore.com/pub/nmh/

[*SNMP*]
See http://www.generalconcepts.com/resources/snmp/ for a collection of pointers to SNMP-related information.

[*Spectrum*]
The Spectrum network monitoring and management system. See http://www.aprisma.com/

[*Steiner88*]
Jennifer G. Steiner, Clifford Neuman, Jeffrey I. Schiller, Kerberos: *An Authentication Service for Open Network Systems*, USENIX Conference Proceedings, pp 191-202, Winter 1988, Dallas.

[*UCDSNMP*]
The UCD-SNMP tools for SNMP, from the University of California, Davis. See http://net-snmp.sourceforge.net/

[*Venema*]
Wietse Venema (wietse@porcupine.org), TCP Wrapper. ftp://ftp.porcupine.org/pub/security/index.html

[*Ylonen96*]
Tatu Ylönen, *SSH - Secure Login Connections Over the Internet*, Sixth USENIX Security Symposium, pp 37-42, July 22-25, 1996, San Jose, CA. See also http://www.ssh.fi/ and http://www.employees.org/~satch/ssh/faq/

[*Zwicky91*]
Elizabeth Zwicky, *Torture-testing Backup and Archive Programs: Things You Ought to Know but Probably Would Rather Not*, Fifth Systems Administration Conference (LISA '91), pp 181-185, San Diego.

[*Zwicky99*]
Elizabeth Zwicky, "Enough SNMP to Be Dangerous, parts 1, 2 and 3," *;login:*, vol 23 no 6, December 1998; vol 24 no 4, August 1999; vol 24 no 6, October 1999.

# Glossary

**Access restrictions**
Mechanisms or processes which serve to limit physical or logical access to systems, services, or networks.

**Alert**
An exception identified by a monitoring system that is to be reported.

**Authentication**
The process of identifying a particular user, typically with some form of userid and password pair.

**Authorization**
The explicit or implicit granting of access to particular files, devices, systems, or applications.

**Backup**
A duplicate copy of programs or data, usually kept offline and on different media.

**Border Gateway Protocol**
An exterior network routing protocol, used between independent networks. BGP is the primary routing protocol used between different networks on the Internet.

**BGP**
See Border Gateway Protocol.

**BOOTP**
Boot Protocol—used to configure diskless workstations or other network devices.

**Cluster**
An integrated collection of two or more servers, with shared peripherals, storage, and/or IP addresses, intended to provide more reliable or scalable services.

**DHCP**
Dynamic Host Configuration Protocol—a superset of BOOTP that can distribute dynamic addresses and other additional information.

**Disaster recovery planning**
Preparing for the worst, and how to get operational again after some form of disaster.

**Dispatch**
To send an alert, by some appropriate mechanism, to a system administrator or a management console.

**DNS**
Domain Name System—the naming system used on the Internet.

**DRP**
See disaster recovery planning.

**Exception**
An unusual or unexpected condition identified by a monitoring system.

**Field replaceable unit**
A device that can be easily replaced by a simple swap, with minimal manual (re-)configuration required.

**FRU**
See field replaceable unit.

**Hot spare**
A device or component that is running and ready to replace a failed component. Often seen in the context of otherwise unused disks in a disk array.

**Hot Standby Router Protocol**
A protocol used with multiple cooperating routers to provide network redundancy, most often used at the edge of a LAN.

**HSRP**
See Hot Standby Router Protocol.

**IP**
Internet Protocol—the primary transport layer networking protocol used on the Internet.

**ISP**
Internet service provider—provides IP connectivity to the Internet.

**LAN**
Local area network, typically within an office area or building

**MAC address**
Media Access Control address—a unique hardware address that identifies a network node. An ethernet MAC address is usually expressed as a sequence of six pairs of hexadecimal digits.

**Mission critical**
When your service just has to be there overnight.

**N+1**
A system in which every critical component has at least one spare. If N devices are required in normal operation, N+1 are installed and configured. An "N+1 data center" is one in which every critical device is protected against failure by redundant spares.

**One-Time Password**
A password that may only be used once, usually implemented as a computed series of single-use passwords, or through some form of token-based or challenge-response mechanism.

**Open Shortest Path First**
An interior network routing protocol, most often used within a single administrative domain, or among closely cooperating groups.

**OSPF**
See Open Shortest Path First.

**OTP**
See one-time password.

**Paperwork**
Appropriate documentation of all planning and execution, including such things as part and serial numbers, configuration parameters, locations, connections, etc.

**Performance**
Correctly executing according to plan and correctly following established procedures and guidelines in a professional, efficient, and effective manner.

**Planning**
The appropriate identification of tasks, processes, and materials in advance of starting work, including the identification of how to "back out" changes if necessary or appropriate.

**Pre-Action sprinklers**
A sprinkler system for fire suppression in which the water pipes are empty until needed, and an external valve controls the water flow rather than the sprinkler heads themselves. This lessens the ramifications of a plumbing failure.

**Preparation**
Ensuring that all necessary materials are available, appropriate training has been completed and/or knowledge is available, and appropriate notice has been given, in advance of starting work.

**RAID**
Redundant Array of Inexpensive Disks, as defined in [*Patterson87*], used to provide disk storage that can continue operating in the event of a disk failure.

**Recovery**
The process of retrieving everything from a backup, following some form of disaster.

**Redundancy**
The replication of systems or system components to allow for continued operation in the event of a failure.

**Repairability**
The ability to quickly and effectively identify and repair or replace failed components or systems.

**Repeatability**
The ability to reliably and consistently repeat installation, configuration, and maintenance tasks.

**Restore**
The process of retrieving a copy of one or more files from a backup.

**Server**
A computer or other device that can be configured to provide one or more services.

**Service**
A function or process implemented on one or more servers. Typical examples of services are file storage, email transport and mailbox access, Web sites, and database access.

**Service level agreement**
A commitment by a service provider to a specific minimum level of service availability and performance.

**Simple Network Management Protocol**
A widely implemented mechanism for remotely monitoring and managing network-attached systems and devices.

**Single point of failure**
Any point in a system or network where a single device or equipment failure can make the larger system effectively inoperable.

**SLA**
See service level agreement.

**SNMP**
See Simple Network Management Protocol.

**System**
A collection of one or more servers, working in concert to provide a service. A system can include independent, redundant, or clustered servers.

**Uninterruptible power supply**
A battery-backed power supply that allows systems to keep running during power failures.

**UPS**
See uninterruptible power supply.

**VPN**
Virtual private network—an encrypted link joining two (or more) "private" networks, tunnelled over a public network link.

**WAN**
Wide area network; links multiple offices, sites, or cities

# Index

***S***



***T***



***U***



***V***



***W***

## About the Author

John Sellens is the General Manager for Certainty Solutions in Canada, based in Toronto (Certainty Solutions was formerly GNAC—Global Networking and Computing). Before joining Certainty Solutions, he was Director of Network Engineering at UUNET Canada, and was employed for 11 years at the University of Waterloo in a variety of computing-related positions. He has also worked as a public accountant and a music store clerk. John has a master's degree in Computer Science from the University of Waterloo, and is a Chartered Accountant.

John has written a number of articles and papers for USENIX over the years, including the articles on which this booklet is based. John enjoys teaching and talking, so much so that sometimes he just won't shut up.

John, Joanne, and their delightful children live in Unionville, Ontario. John would like to express his heartfelt thanks to his entire family for their loving forbearance of all his uncommon quirks.