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.

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.