How To Accelerate Your Internet

A practical guide to Bandwidth Management and Optimisation using Open Source Software

How To Accelerate Your Internet

For more information about this project, visit us online at *http://bwmo.net/*

Editor: Flickenger R.
Associate Editors: Belcher M., Canessa E., Zennaro M.
Publishers: INASP/ICTP

© 2006, BMO Book Sprint Team
First edition: October 2006

ISBN: 0-9778093-1-5

Contents

Preface ix

About This Book xi

Introduction 15

Bandwidth, throughput, latency, and speed.....16
Not enough to go around.....17
Where to begin.....19

Policy 23

The importance of policy.....24
Explosive network growth at Havensburg.....24
Bandwidth as a public good.....25
Desperate measures.....26
Policy, strategy, rules and regulations.....27
Real policy development at Havensburg.....28
Characteristics of good policy.....29
The new Havensburg network policy.....30
The policy development process.....31
Policy is needed in all environments.....33
Policy pitfalls.....34
Example policies.....34
Policy checklist.....35
References.....36

Monitoring & Analysis 39

Networking 101 40
Introduction 40
Cooperative communications 42
The OSI model 42
The TCP/IP model 45
The Internet protocols 46
Networking hardware 58
Physical connectivity 63
Virtual connectivity 72
What is network monitoring? 76
An effective network monitoring example 77
Monitoring your network 80
The dedicated monitoring server 81
What to monitor 84
How to select tools to monitor the network 85
Types of monitoring tools 86
Walking around the lab 87
Spot check tools 88
Log analysers 94
Trending tools 97
Realtime tools 101
Benchmarking 103
What is normal? 105
How do I interpret the traffic graph? 109
Monitoring RAM and CPU usage 111
Resources 113

Implementation 115

The importance of user education 116
The 5/50 rule 116
Providing feedback to users about network load 117
General good practices 119
Essential services 126
Firewall 128
Caching 148
Mirroring 158
Email 162
Resources 170

Troubleshooting 173

Proper troubleshooting technique 173

Preparing for problems......174
Responding to a problem......174
A basic approach to a broken network......175
Common symptoms......178
Automatic updates......178
Spyware......179
P2P......179
Email......179
Open email relay hosts......180
Email forwarding loops......181
Open proxies......181
Programs that install themselves......181
Programs that assume a high bandwidth link......181
Windows traffic on the Internet link......182
Streaming media / Voice over IP......183
Denial of Service......184
Rogue DHCP servers......184
Port analysis......185
Browser prefetch......186
Benchmark your ISP......186
Large downloads......186
Large uploads......187
Users sending each other files......187
Viruses and worms......188

Performance Tuning 191

Squid cache optimisation......192
Cache server hardware......193
Tuning the disk cache......194
Memory utilisation......195
Tuning the hot memory cache......196
Cacheable content limits......196
Access Control List (ACL) optimisation......197
Redirectors......198
DansGuardian......199
Authentication helpers......200
Hierarchical caches......201
Configuring delay pools......203
More information......205
Monitoring your Squid performance......206
Graphing Squid metrics......209
Traffic shaping......210
Linux traffic control and QoS tools......210
Traffic shaping with BSD......217
Farside colocation......219

Choosing a colo or ISP 222
Billing considerations 222
Protocol tuning 223
TCP window sizes 223
Link aggregation 224
Bonding 225
Aggregate routing 225
DNS optimisation 226
Web access via email 228
www4mail 229
web2mail 229
PageGetter.com 230
GetWeb 230
Time Equals Knowledge (TEK) 230
Other useful web-to-email applications 231
loband.org 231
High Frequency (HF) networks 232
Modem optimisation 233
Hardware compression 233
Software compression 234
Bandwidth accounting 235
Squid bandwidth accounting 235
Bandwidth accounting with BWM tools 236
Linux interface bandwidth accounting with RRDtool 237
VSAT optimisation 237
Use of inclined orbit satellite 238
C band, Ku band, and Ka band 238
Shared vs. dedicated bandwidth 240
Resources 246

Case Studies 249

KENET, Kenya 249
Problems 250
Analysis 250
Solutions 250
Site One: firewall & proxy server 251
Site Two: proxy & mail server 251
Site Three: FOSS traffic shaper 252
Aidworld in Accra, Ghana 253
BMO in the UK 255
JANET, UK 255
Blackburn College, UK 257
Malawi 259
One Bellevue Center 261
Carnegie Mellon University 262

Workaround #1: Best effort rate limiting....262
Getting more than you paid for....262
Workaround #2: Fun with rate limiting....263
More problems with packet drops....263
Requirements and considerations....264
Researching hardware rate limiters....264
Final solution or new workaround?....264
Application layer analysis to the rescue....265
Social engineering....265
The campus bandwidth usage guidelines....266
Human effort....267
Positive results....267
Conclusion....267

The Future 269

Bandwidth consuming technologies....269
Trends in developing countries....270
New software....271
In closing....272

Resources 273

Links....273
Wikipedia entries....281
Relevant RFCs....281

Squid ACL Primer 283

ACL elements....283
ACL rules....285
Examples....286
Allow only local clients....286
Deny a list of sites....287
Block a few clients by IP address....287
Allow access to the bad sites only after hours....287
Block certain users regardless of their IP address....287
Direct certain users to a delay pool....287

Glossary 289

Preface

One measure of the growing disparity between the developed and developing worlds is the speed of the Internet. For example, the speeds of connections from North America to Africa are slower than those to Europe by a factor of 50 or so. Such assessments have been made by measuring the round trip time that it takes for a digital pulse sent over the Internet to return to the sender.

The reasons for this disparity include the availability of Internet access only via slow satellite connections, and the lack of communications infrastructure in the remote parts of the world. Bandwidth and computing equipment are expensive as a result of weak currencies, high transport costs, small budgets and unreasonable tariffs. Bandwidth in some developing countries can be so costly that even their prime universities cannot afford speeds equivalent to the average western household with an ADSL connection. Thus universities and other institutions cannot afford a decent link, or are simply unaware of existing alternatives.

This book attempts to provide practical information on how to gain the largest benefit from existing connections to the Internet, by exposing readers to the latest techniques to optimise the use of low-bandwidth network connections. By applying optimisation techniques based on open source technologies discussed here, the effectiveness of available connections can be significantly improved. Access to more bandwidth will facilitate better exchange of scientific information, data and literature among researchers all over the world. One hopes that the process will enable every scientist to become part of the scientific enterprise no matter where geographically she is located with respect to the main centers of modern science.

While the Internet has helped global communication, and its use is rising everywhere, the fraction of people with access to it is far higher in rich countries than in poor countries. The average per capita income in industrialised nations is about $27,000 per year, compared with barely $2,000 or so in the developing world. Literacy rates approach 100% of the adult population in developed countries, but the figure falls to below 50% in developing nations. Even as the world is becoming more interconnected, it is becoming increasingly divided in these regards.

This book is a collaborative effort enabled by the support of INASP (UK) and ICTP. The effort that has gone into its preparation will be rewarded if the book can reach large audiences of interested readers and assist them in improving the quality of service of the bandwidth available to them. The authors of the book realise that it is a small drop in the huge ocean of bits and bytes, but the value of their service is not in any doubt. I congratulate them on their work and their decision to make the book freely available both in print and on the Internet.

K.R. Sreenivasan
Abdus Salam Professor
Director, ICTP

Trieste
October 2006

About This Book

Credits

This book was started as a BookSprint project at the ICTP in Trieste, Italy, in May of 2006. A core team of ten experts in the field of bandwidth management built the initial outline, and developed the book over the course the following months. Throughout the project, the core group has actively solicited contributions and feedback from the Internet community, particularly those who work in the area of bandwidth optimisation in the developing world. The final manuscript was produced by Hacker Friendly LLC in Seattle, WA (USA).

Contributors

- **Aidworld** (*http://www.aidworld.org/)* is a not-for-profit organisation focussed on information technology for international development. Aidworld's mission is to effectively support the Millennium Development Goals with appropriate ICTs. Aidworld builds bandwidth management solutions and helps NGOs and others make their online services accessible in the developing world. Aidworld has also created an on-line tool (*http://www.loband.org/*) that shrinks web pages so they are accessible over poor internet connections. Aidworld contributors include **Nishant Bhaskar**, **Hamish Downer**, **Alan Jackson**, **Simon Liu**, **Tom Lord**, **Jon Stafford**, **Nick Street**, **Tom Taylor,** and **Chris Wilson**.

- **Martin Belcher** is the Senior Programme Manager for the International Network for the Availability of Scientific Publications (INASP), Lund, Sweden. He can be reached at *mbelcher@inasp.info* .
- **Enrique Canessa** is a PhD Physicist working at the ICTP in Trieste, Italy. His areas of interest are scientific software applications, ICT training, and dissemination of science to/from and within developing countries using open source technologies.
- **Kevin Chege** is the Senior Network Administrator at the Kenya Education Network (KENET). He is an avid user of FreeBSD and an open source enthusiast focusing on improving ICT reach in education using FOSS tools. He can be contacted at *kevin@kenet.or.ke*.
- **Rob Flickenger** was the lead editor of this project, and is the founder of Hacker Friendly LLC. Rob is a long-time supporter of the use of wireless networking to extend the reach of the Internet. He can be reached at *rob@hackerfriendly.com*.
- **Carlo Fonda** is a member of the Radio Communications Unit at the Abdus Salam International Centre for Theoretical Physics in Trieste, Italy.
- **Duncan Greaves** is an Executive Officer at the Tertiary Education Network (TENET), a not-for-profit company supporting higher education in South Africa. Duncan oversees TENET's capacity development programs. He can be contacted at dbg@tenet.ac.za.
- **Casey Halverson** is a Network Engineer at Infospace Inc. in Seattle, Washington, USA. He has a broad experience in WAN optimisation, traffic shaping, and other bandwidth management techniques. He is also a member of the SeattleWireless network project. *http://seattlewireless.net/*
- **Peter Hill** is a self-titled "Holistic Network Engineer" for the University of Washington. He previously worked in the trenches keeping Amazon's network afloat, and still has fond memories of Carnegie Mellon's network and awesome Network Development team.
- **Nigel Kukard** has a PhD in Computer Science, and has been a passionate supporter of open source (GPL) software for over ten years. He is the founder of LinuxRulz (*www.linuxrulz.org*) and the Linux Based Systems Design group of companies. Can be reached at *nkukard@lbsd.net* .
- **Richard Stubbs** is a technical evangelist who works for the University of KwaZulu-Natal in South Africa. He has been involved with the Internet and associated activities at the Institution for the past 15 years. He can be contacted at stubbs@ukzn.ac.za
- **Marco Zennaro** is an electronic engineer working at the ICTP in Trieste, Italy. He has been using BBSes and ham radios since he was a teenager, and is happy to have merged the two together working in the field of wireless networking.

Additional material

Portions of this work were adapted from:

- Network traffic monitoring and analysis workshop (INASP) by Dick Elleray, AfriConnect, 2006 *http://www.inasp.info/training/bandwidth/bmo-ntmw/*
- Optimising Internet Bandwidth (INASP) by Gerhard Venter, AfriConnect, 2003 *http://www.inasp.info/pubs/bandwidth/index.html*
- The VSAT Buyer's Guide, IDRC, 2005 *http://ictinafrica.com/vsat/*
- Wireless Networking in the Developing World, *http://wndw.net/*

Funding

The development and publication of this book have been made possible with funding support from:

- Department for International Development, UK (DfID); through funding to INASP for the "Programme for the Enhancement of Research Information (PERI)."
- Flemish Interuniversity Council (VLIR); through funding to INASP for the programme "Optimization of the use and management of bandwidth at university level."
- International Network for the Availability of Scientific Publications (INASP).
- International Development Research Centre (IDRC): through funding to INASP for the programme "Supporting training for the optimization of university bandwidth in Africa" undertaken with financial support from the Canada Fund for Africa.

Special thanks

The production team would like to thank the ICTP (*http://www.ictp.it/*), INASP (*http://www.inasp.info/*), and everyone else who has made this project possible.

The Abdus Salam International Centre for Theoretical Physics

International Network for the Availability of Scientific Publications INASP

1
Introduction

The Internet has irrevocably invaded many aspects of daily life. What was once an obscure scientific research tool has blossomed into a communications platform used by hundreds of millions of people. Telecom providers use the Internet to carry critical voice communications. Banking institutions use it to provide access to account services and market trading. Airline tickets, hotel reservations, and car rentals can all be booked with a click of the mouse. Whole industries have sprung into existence with business models that depend entirely on Internet infrastructure to reach their customers. More users than ever depend on the Internet to connect with family and colleagues using email, instant messaging, Voice over IP, photo and video sharing services, and online journals. Children born in the last ten years have grown up in a time when the Internet has always been available.

This point of view is popular among Internet users, but it does not necessarily reflect the experience of all, or even most, of the rest of the world. According to the ITU*, more than half of the users on the Internet are concentrated in the G8 countries (Canada, France, Germany, Italy, Japan, Russia, the UK, and the US). In 2004, less than 3% of Africans used the Internet, compared with an average of one 50% of the inhabitants of the G8 countries. The entire African continent accounts for about 13% of the total world population, yet in 2004 it had fewer Internet users than France alone.

Fortunately, in places where the Internet has not yet penetrated, it is all but certain to happen soon. There is a global push to bridge the so-called digital divide by bringing modern telecommunications to the developing world. State and private investment in public infrastructure, in the form of fibre optic backbones, wireless networks, and satellite connectivity are bringing the Internet to the most remote locations at a pace that is accelerating over time. People all

* Source: http://www.itu.int/ITU-D/ict/statistics/ict/

over the globe are beginning to realise that in order to effectively participate in the global marketplace, they need access to the global communications network.

But superhighways aren't built overnight. As with any major undertaking to build infrastructure, extending fast network connections to all of the ends of the earth takes time. Technologies such as VSAT make it possible to install an Internet connection just about anywhere, particularly in the absence of existing wired infrastructure. While this does extend the footprint of the Internet to otherwise unreachable places, the capacity of the connection provided is far from infinite. The cost of these connections is also quite high for many organisations. This often leads to the practice of stretching an insufficient network connection to serve many users simultaneously.

Bandwidth, throughput, latency, and speed

There are a few technical words used to describe how fast an Internet connection may go. Users often find these terms confusing, so it's best to be clear about their definitions from the beginning.

- ***Bandwidth*** refers to a measure of frequency ranges, typically used for digital communications. The "band" part of broadband is short for bandwidth, meaning that the device uses a relatively wide range of frequencies. In recent years, the term bandwidth has been popularly used to refer to the ***capacity*** of a digital communications line, typically measured in some number of bits per second. In its popular usage, you might read that a T1 provides a theoretical maximum "bandwidth" of 1.544 Mbps.

  While some purists insist that we should speak of capacity when talking about data transfer speeds and bandwidth when talking about frequency ranges, the popular usage of the term "bandwidth" has been reinforced by years of product marketing and misleading documentation. There simply is no going back now. Therefore, we will use the terms bandwidth and capacity interchangeably in this book.

- ***Throughput*** describes the actual amount of information flowing through a connection, disregarding protocol overhead. Like bandwidth, it is expressed in some number of bits per second. While a T1 may provide 1.544 Mbps between the endpoints, the protocol spoken on the physical line reduces the effective throughput to about 1.3 Mbps. When you factor in the additional overhead of Internet protocols, the available throughput is even less. When you measure the actual usage of a connection or perform a "speed test" on a line, you are measuring throughput.

- ***Latency*** refers to the amount of time it takes for a packet to travel from one point on a network to another. A closely related concept is ***Round Trip Time*** (***RTT***), which is the amount of time it takes for a packet to be acknowledged from the remote end of a connection. Latency is measured as some amount of time, usually in milliseconds. The latency of Ethernet is about 0.3 ms. A T1 connection has a latency of 2 to 5 ms, while a VSAT connection requires at least 500 ms before an acknowledgment can be received, due to the speed of light and the large distances involved. Some factors that contribute to latency are network congestion, overutilised servers, and the distance between the two points.

- ***Speed*** is an ambiguous term that refers to some combination of these other terms. An Internet connection may "feel slow" when using an interactive service (such as Voice over IP or gaming) on a line with high latency, even if there is sufficient bandwidth. Users will also complain when transferring large files on a connection with insufficient capacity, even if the latency is very low.

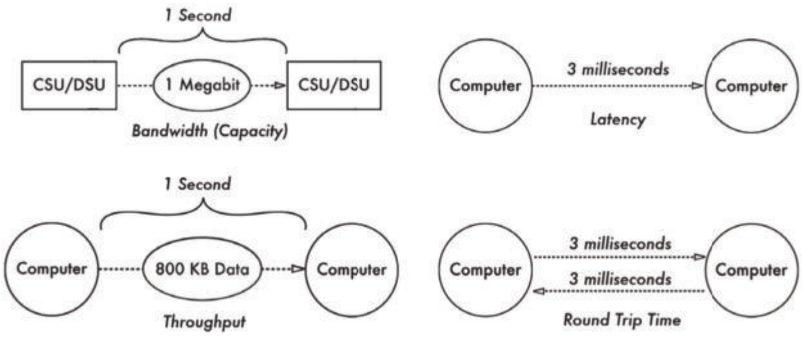


Figure 1.1: Bandwidth, Capacity, Throughput, Latency, and Round Trip Time.

The goal of this book is to show you how to optimise your Internet connection so that it provides the greatest possible throughput and lowest possible latency. By eliminating wasted bandwidth, the cost of operating your network connection will be reduced, and the usability of the network will be improved.

Not enough to go around

What actually causes a slow Internet connection? Obviously, the capacity of a given connection is finite, so if too many people request information at once, then someone will have to wait. In an ideal world, organisations would simply order more bandwidth to accommodate the increased traffic. But as we all know, Internet access costs money, and most organisations do not have infinite budgets.

It is an interesting fact of online life that users tend to consume more bandwidth over time. It is very rare to find a user who, once they have had access to a broadband connection, is satisfied with going back to a low speed dialup line. As users are exposed to Internet services earlier in life and in a variety of venues (for example at home, at work, at University, or at a cyber-cafe), they become accustomed to using it in a certain way. They are increasingly unlikely to know or care about the bandwidth required to listen to Internet radio, or to download the latest video game, or to watch funny movies on a video sharing service. They "just want it to work," and may complain when the Internet "is slow." Users often have no idea that they can single-handedly bring an organisation's Internet connection to a halt by running a simple file sharing program on their computer.

User education is obviously critical to every stage of implementing a plan to manage your bandwidth. While users can be forced to adhere to certain behaviour patterns, it is always far easier to implement a plan with their voluntary compliance. But how does such a plan come into being? If you simply order people to change their behaviour, little is likely to change. If you install technical hurdles to try to force them to change, they will simply find a way around the obstacles.

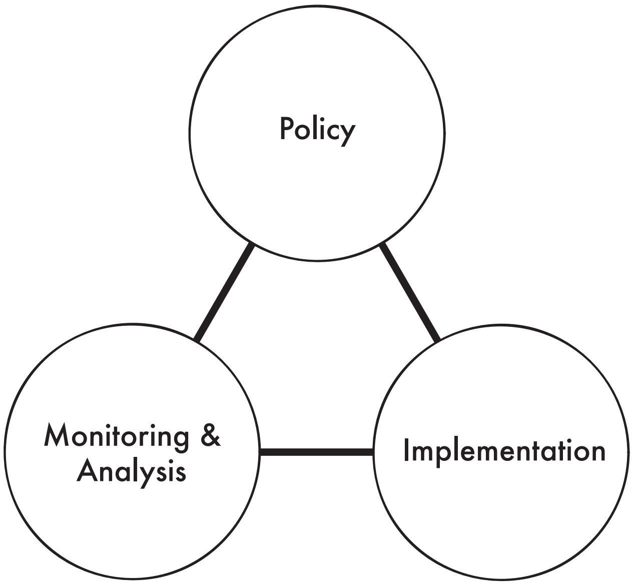


Figure 1.2: Policy, Monitoring & Analysis, and Implementation are three critical (and interdependent) components of bandwidth management.

In order to effectively manage a network connection of any size, you will need to take a multifaceted approach that includes effective ***network monitoring***, a sensible ***policy*** that defines acceptable behaviour, and a solid ***implementation*** that enforces these rules. Each component is important for effective bandwidth management in any network that consists of more than a few users. This book includes chapters devoted to each of these three major areas.

A **policy** is a statement of opinions, intentions, actions and procedures that guide the overall use of the network. An **acceptable use policy** is a subset of this, setting out in technical detail what uses of the network are believed by the network operators to be acceptable, and what they intend to do to anyone who uses it in a manner that they consider unacceptable. It should be a written document that defines acceptable forms of network access, as well as guidelines for how network problems are dealt with, definitions of abuse, and other operational details. The policy also typically includes definitions of legal constraints for network users (such as the exchange of copyrighted material, requesting inappropriate materials, etc.). Having a policy makes it much easier to enforce certain types of network behaviour, as you will be able to hold people to a set of agreed rules.

Network monitoring is the ongoing process of collecting information about various aspects of your network operations. By carefully analysing this data, you can identify faults, find cases of waste and unauthorised access, and spot trends that may indicate future problems.

Implementation is the step of implementing traffic shaping, filtering, caching, and other technologies within your network to help bring actual usage in line with policy. The actions you need to take are indicated by the data collected through monitoring and analysis, and are constrained by the network policy. Many people expect to begin the task of bandwidth management by starting with this step. But without good monitoring techniques, you are effectively blind to the problem. Without a policy, your users will not understand what you are doing or why, and will complain or subvert your actions instead of helping you to achieve your goal.

Don't underestimate the value of personally interacting with your network users, even at a very large institution. At Carnegie Mellon University (page **248**), social interactions made a far greater impact on bandwidth consumption than did technical constraints. But at an organisation as large as CMU, personal attention could only have had this effect by operating within a well-defined policy, with the support of a good network implementation and watched by careful network monitoring.

Where to begin

Effective bandwidth management can only happen by applying a combination of technical computer skills, effective network monitoring, and a sensible policy that is understood by all users. If your organisation has a small network, one person may need to work on all of these areas. Larger organisations will likely require a team of people to effectively manage busy networks, with each person specialising in a particular area.

This book is designed to be used as both a guide and a reference to anyone who needs to tackle this difficult problem. While you may read it cover-to-cover, each chapter is designed to stand on its own and address a particular aspect of bandwidth management. If you don't know where to begin, these guidelines should help you find a good starting place.

Do you need to fix your network immediately?

- Is something wrong with your computers or Internet access?
- Do the problems get in the way of people getting legitimate work done?
- Is your job at risk if you don't do something now?

If you answered **yes** to any of these questions, go to the **Troubleshooting** chapter (page **159**). When you've solved the immediate problem, continue with the steps below.

Do you know what's happening on your network?

- Do you monitor your network?
- Do you know what your bandwidth usage is, on average?
- Do you know who is using your bandwidth?
- Do you know how your bandwidth is being used? How much bandwidth is used for email, as compared to web traffic and peer-to-peer applications?
- Do you know about network outages before your users complain?
- Are you certain that your network only being used for appropriate services, and has not been compromised by a malicious user?

If you answered **no** to any of these questions, take a look at the **Monitoring & Analysis** chapter on page **25**. When you have a clear idea of what's happening on your network, continue with the steps below.

Do you want to change how users behave on your network?

- Is inappropriate user behaviour (e.g. peer-to-peer file sharing or excessive downloads) causing problems on your network?
- Do you need to create a written policy on network usage?
- Do you need to update an existing policy?
- Are your users largely unaware of what the network policy is, and why it is important?
- Do you need to guarantee the availability of certain services on your network?

If you answered **yes** to any of these questions, you will want to start with the **Policy** chapter (page **9**). When you have established a policy, please continue with the steps below.

Are you using basic optimisation techniques?

- Do you operate your network without a site-wide web cache?
- Do responses to DNS requests seem sluggish?
- Are spam and viruses wasting a significant amount of your bandwidth?
- Do your users make extensive use web mail services, such as Hotmail or Yahoo! Mail?

If you answered **yes** to any of these questions, you should start with the **Implementation** chapter on page **101**. Please be aware that technical solutions, while important, are unlikely to help unless you already have a well-defined and well-known network usage policy, and have already implemented good network monitoring.

Do you need to enforce further technical constraints on the network?

- Do you need to reduce the bandwidth used by certain services?
- Do you need to guarantee bandwidth for certain services (such as email) at the expense of others (such as web browsing)?
- Do you need to block some kinds of traffic entirely?
- Are some users able to monopolise the available bandwidth, effectively blocking access for all other users?
- Does your network usage exceed the available capacity of a single line, requiring you to make use of multiple Internet connections?

If you answered **yes** to any of these questions, you will want to start with the **Performance Tuning** chapter on page **177**. These steps should only be taken after basic optimisation methods have been implemented.

Do you need to convince someone else of the importance of bandwidth management?

Go to the **Case Studies** chapter (page **235**) to see examples of how bandwidth management is used in real organisations.

Do you want to know how to reduce your personal bandwidth use?

See the **General Good Practices** section on page **105**.

2
Policy

This is a story about Havensburg University, which doesn't exist. The elements of its story are taken from those of many different institutions and organisations, and are assembled to illustrate the scope and limits of policy in managing bandwidth.

Havensburg first connected to the Internet in 1988, with a circuit initially of 64 kbps, rising to 192 kbps by 1992. During these years the dominant protocols on the Internet were email, ftp, gopher, and nntp. Users were mostly in the scientific community, and they generally used one of three shared Sun computers. Almost every member of the Internet user community on the campus knew every other.

In 1992, things had started to change. Ethernet networks had started to become common on the campus. With some difficulty, users of these networks could get a TCP/IP stack on their PC and a connection to the Internet. Email had come into increasing use in the non-scientific community. Windows 3.0 began to appear on PCs. Its graphical user interface made the PC attractive to non-technical users. In 1993 the NCSA Mosaic browser was released; later that year, the first commercial websites appeared. By 1994 the web was clearly the dominant Internet service. Havensburg's academic community clamoured for access to it; in response, the University approved plans for increased expenditure on the campus network, and doubled the capacity of the Internet connection to 512 kbps.

By 1996, enterprising academics were demanding Internet access for students, and the first large student computer labs began to appear. In the space of two years, the number of hosts connecting to the Internet had risen **tenfold**. Despite the increase in bandwidth, response times had fallen dramatically. Academics were starting to complain aggressively about poor performance, and the University Budget Committee had started to balk at the cost of Internet access. Despite this, the build-out of student computer laboratories continued, and many academic departments were insisting on a PC for every member of staff. Non-academic departments were beginning to demand the same.

The importance of policy

An abundance of bandwidth enables electronic collaboration, access to informational resources, rapid and effective communication, and grants membership to a global community. An absence of bandwidth prevents access to the aforementioned global community, restricts communications, and slows the speed at which information travels across the network. Therefore, bandwidth is probably the single most critical resource at the disposal of a modern organisation.

Because bandwidth is a valuable and costly resource, demand usually exceeds supply. In many environments, unrestrained access and usage of bandwidth results in degraded service for all users. This is partly a supply problem (not enough bandwidth is available to meet demand), partly a demand problem (too many demands are being made on the limited resource), and partly a technical problem (little or no technical management and optimisation of the resource is happening). The end result is a poor user experience when trying to use resources and tools that rely on bandwidth (e.g., browsing the web, sending emails, using network applications, etc.).

Bandwidth management and optimisation are often seen as technical issues. However, policy is an essential component of any bandwidth management strategy. Without it, technical solutions will be difficult to implement and much less effective. Policies are essential, in that they provide the framework for defining how a network is to be used and detail how technical solutions should be implemented.

Policy should be thought of as guidelines concerning network usage for both the users and those responsible for maintaining the network itself. In the case of Havensburg University, these guidelines were not developed to match the growth of the network. Without a plan, unrestricted access to the campus network would push its management into total chaos.

Explosive network growth at Havensburg

By early 1997, demand for Internet access had far outstripped supply and the Internet was effectively unusable on campus. The Computer Services Management Committee then stepped in and appointed a task team to analyse the problem and make recommendations. The team recommended doubling the available bandwidth, implementing NNTP and web caching, and aggressive pruning of the Usenet groups carried by the University's news server.

With some difficulty, the University Budget Committee was persuaded to approve the increase in bandwidth, believing that the new measures would bring an improvement in service. There was indeed a brief improvement, but by 1999 demand was again rising sharply, and the emergence of peer-to-peer networks - beginning with Napster in that year - was threatening a crisis. Academics were demanding a tenfold increase in bandwidth and were threatening to install independent connections to the Internet. Many began to use dial-up connections from their offices rather than tolerate the abysmal performance of the campus network. It became obvious that unrestricted network access could simply no longer be supported.

Bandwidth as a public good

In many institutions, bandwidth can be thought of as a ***public good***. By "public goods," economists generally mean a resource that can be consumed by an individual in arbitrarily large amounts, irrespective of the contribution made by that individual to conserving or renewing that resource. (The technical definition is a good deal more complex, but this is sufficient for our purposes.) Public goods are notorious for being liable to over consumption, and it can be shown that the rational, self-interested individual will almost always choose to over consume – even though this leads to a collective outcome that is bad for everyone. A "public goods problem" is any problem that arises out of this paradoxical tendency. Public goods problems can be managed in a number of ways: for example, by rationing the good, by converting it from a public good into a private good, by coercing appropriate behaviour, by educating consumers, and by fostering community spirit.

Those concerned with managing bandwidth need to be informed of this dimension regarding public goods. In particular, they should be made aware that it only requires a small group of abusers to wreck the availability of 'the good' (or bandwidth) for the group at large. It is almost always the case that a small minority of (ab)users account for most of the consumption of an over consumed public good. Thus, **5-10% of users create 50-60% of the problems**.

Policy aims to manage the behaviour of this minority. If a majority are over-consuming bandwidth, then the problem is probably of a different kind: most likely of undersupply (i.e., not enough of the bandwidth is being provided to meet the reasonable needs of the users).

Good policy also has an ***enabling purpose***. Policy is not just a set of arbitrary restrictions about how a network may or may not be used. Its central purpose is to govern usage of a resource to provide equitable access to all of its users. By enacting policy, we limit the ability of the minority abusing the network to infringe on the majority who need to use the network.

At Havensburg, students were not aware of the criteria that constituted acceptable use, because no relevant policy was in place. IT staff could not solve network congestion issues because they were unable to decide which services deserved priority, and which should be cut off altogether. If Havensburg was going to continue to offer network services to faculty and students, something had to change.

Desperate measures

At this point, the Computer Services Management Committee decided to begin charging students for web access. The proposal was strongly resisted by students, who marched on the Computer Services Building in protest. Despite this, student charges for web access were eventually implemented in 2001, based on volumes of traffic downloaded. Surprisingly, this had very little effect on consumption. Some cash was generated, but university policy prevented it from being used to improve Internet access.

The Computer Services Management Committee then proposed to extend charging to staff, a proposal that was rejected by the University Executive. Instead, the Executive demanded an accounting of what the Internet access circuit was being used for, and by whom. Such an analysis had never been undertaken before, on the grounds that it would violate rights of privacy. A group of academics raised a formal protest in the University senate on precisely these grounds, but the senate finally decided that Internet access was a common good and that the rights of the community trumped the individual's right to privacy.

The University's lawyers advised that there was no inherent right of privacy when using a resource paid for by the University, provided that the University advised its members of this in advance. On this basis, the University took two decisions: first, that all users of the Internet would henceforth be authenticated, and second, that Internet usage would be analysed after a period of three months.

These announcements by themselves produced a drop in traffic, but not enough to make a major difference. After three months, log files were exhaustively analysed. The conclusions were, among other things, that:

- Not all accesses were being authenticated. Some users could not be identified by name because they were finding ways to circumvent the authentication.
- Even when users were being authenticated, the nature of their usage could not always be determined: inspection of both packet contents and source revealed no meaningful information, since the data was often tunneled and encrypted.

- A great deal of material that could be identified had no demonstrable relationship to the University's ordinary business.
- A small minority of users accounted for most of the traffic.

The IT department investigated the first issue and adopted measures to ensure strict authentication on all accesses. In the case of issues 2 and 3, attempts were made to interview users about their pattern of access. In case 2, most of the traffic was eventually identified as peer-to-peer file sharing. In case 3, responses from users were mixed. Some denied all knowledge of having generated the traffic, and claimed that their workstations had been used by others without their knowledge - or that their PCs had been hijacked by malicious software. In some cases users openly admitted to downloading content for private gratification, but objected that there was no university policy to prohibit it.

In many cases, users had no idea of how much traffic they were generating. When informed, some of them were shocked and agreed to desist. Others shrugged their shoulders and questioned the right of the University to prohibit such activity. Some students insisted that since they were paying fees they had the right to download material for private purposes.

Policy, strategy, rules and regulations

It is important to recognise that policy, strategy, and rules and regulations are all different issues. They should, wherever possible, be dealt with separately. Although related and often closely linked, they are different in important ways. **Policy is not regulation,** and these two areas should be dealt with separately. Regulations are defined from the policy, and policy is derived from the strategy.

The relationships between these different components are important when developing effective policy. Consider the following four levels:

1. **Mission, vision, and values are about objectives.** What do we want to achieve? What are the visions or dreams of the organisation?
2. **Strategy is about the acquisition, development, deployment, and renewal of resources in the pursuit of objectives.** How are we going to get there?
3. **Policy concerns directed behaviour.** We define behaviour as either acceptable or unacceptable. By connecting these interpretations to our high-level definitions (or policy), we make decisions concerning where we want to go and how we plan to get there.
4. **Regulations are the codes of behaviour that policy will mandate.** So policy might say “the IT department shall from time to time set limits on traffic volumes” and the regulation might say “nobody may send an email

attachment larger than 3 Megabytes." Regulations are always made within the mandate established by policy, the do's and don'ts.

Each of these levels are distinct, but support the others. Access to network resources should support the mission of the organisation. Policy makers should develop an explicit strategy to make the best possible use of resources in support of that mission. The strategy is embodied in a published policy that defines acceptable behaviour on the part of network users. The policy is actually implemented through specific regulations that enforce proper behaviour, and define actions to be taken against users who violate the policy.

Real policy development at Havensburg

The University had always had an acceptable use policy for computer access, but it had been drafted in the 1990s and reflected the concerns of a pre-Internet IT department. The policy did not give the network administrators enough flexibility to monitor and manage the Internet connection to prevent abuse, so they convinced the University management to modernise it.

A task team was appointed to consult within the University and to consider the acceptable use policies of other institutions. The task team decided, as a point of departure, that the principle objective of policy was to ensure that Internet resources were used for institutional purposes: that is to say, it began with the assumption that not only the volume of traffic, but also the type of traffic, was relevant to its mandate. With this objective in mind, it embarked on a series of discussions with all academic boards and other institutional committees.

The task team pressed one argument repeatedly: that a minority of people were using the Internet for purely personal ends, and were also responsible for most of the traffic. They illustrated the argument with charts developed from analysis of the log files. They didn't promise that eliminating this traffic would also eliminate the congestion, but they did make a crucial point here: that if an Internet access circuit is being used solely for institutional purposes, and if it is congested, then it must mean that the University is not buying sufficient bandwidth. Every group to which the task team spoke agreed with this analysis.

The task team then drafted a policy, asserting that bandwidth was reserved exclusively for institutional purposes and expressly prohibiting its use for private purposes, and reiterating the University's commitment to respecting intellectual property rights in digital content. The draft policy was eventually approved by the University's board of governors and came into effect in 2002. A copy of the new policy was sent electronically to every student and staff member, and copies were posted in all public access computer facilities.

Characteristics of good policy

When developing a policy, it is worth considering the characteristics that differentiate good policy from bad. Below are details of such characteristics, they are generally policy independent and so are useful guidelines for the development of any policy.

- **Good policy has an enabling purpose.** The aims of the policy should be clear and apply to all users**.** If it is going to restrict user behaviour, then all users need to know why that is. This needs to be clearly stated and easily understood, as all users of your network need to understand this in order for the policy to be effective.

  The aims outlined in the the policy should not be a technical statement (e.g., "this policy exists to optimise the flow of data essential for our core business objectives over our network circuit."). Rather, it should be easy to understand and attempt to foster a collective responsibility towards creating positive network performance. For example:

  > "*Internet access is provided to achieve or sustain our business purpose. Using it for personal reasons compromises that goal by potentially slowing or halting important network services . This is why we have chosen to prohibit personal Internet use, except for the limited use described in [section y].*"

- **Good policy is linked to a wider objective.** Why is the policy trying to enable the above? The wider objective should relate to the bottom-line of the organisation. For example, a university might want to encourage education, teaching, and research. A human rights NGO's purpose might be about achieving their mission and objectives. These wider objectives should help focus people's attention on why network access is being provided. For example:

  > "*Internet service is being provided to allow human rights activists to consult appropriate online literature and not to download personal music collections.*"

- **Good policy has clear ownership.** Ownership of the policy should be clear and mandated from an appropriate level within the organisation. Ideally, this level will be that which is representative of all members of the organisation and not be seen as being imposed upon users by one part of the organisation. Wherever possible, the policy should be seen to be the will of the most senior management of the organisation, rather than the IT department, to increase its authority and effectiveness.

- **Good policy is short and clear.** If we want our users to abide by the policy, then they need to be able to read it. If we want them to buy into the policy (e.g., have all new students sign an agreement to abide by the ***Acceptable Use Policy*** (***AUP***)), then it must be easy for them to read and understand. The document should be clearly written and laid out. It should also avoid technical or legal jargon wherever possible.

- **Good policy arises from a valid process.** The process of how the policy was developed and put in place needs to be clear and easily understood by all members of the community it will affect. If it is seen as being imposed by the IT department without consultation, then will it be supported? The process should be clear and ideally show that opportunities for input and comment have been provided. A democratic process is more likely to achieve buy-in from all users.
- **Good policy works within the confines of a given authority.** Without the authority to make policy, it will be difficult to achieve buy-in from users and convince them to submit to the regulations. It is unlikely that a single network administrator can effectively set a policy for an entire university. But if the policy comes from the senate or university council, it is much more likely to be taken seriously. The authority should be above all users at whom the policy is aimed. In most cases, this should include all members of the community. In the case of a university, this includes faculty, staff, and administrators in addition to the student body.
- **Good policy is enforced.** The policy must be enforced and enforceable. If you do not consistently enforce it, then what happens when you do? Can a user claim unfair discrimination? Remember that enforcement is usually only an issue for a very small number of users who are disproportionately using your bandwidth. Evidence shows that enforcement can be achieved at both a technical level (e.g., blocking users or traffic) and a human level (sending a warning email). The simple human level warning is often effective.
- **Good policy is adaptable.** No policy is perfect; it may need revisions, particularly as the network grows. It is also important to provide clear information regarding how it can be changed or questioned. This need not be done in great detail, but it should be clear that the policy is not written in stone.

The new Havensburg network policy

The initial effect of the new policy was to reduce bandwidth consumption dramatically. Within a year, however, utilisation had begun to creep up again and response times were increasing. At this point the IT department was instructed to conduct another exhaustive analysis of log files. It identified six postgraduate students who were generating large volumes of traffic, the character of which was not apparent from the log files. The IT department lodged a formal complaint with the proctor, who instructed that the offending PCs be seized and their contents analysed. This demonstrated conclusively that the machines were being used to download pirated movies from a file sharing network. The students were charged with violation of university policy; two of them were eventually acquitted for insufficient evidence, and the other four were expelled. The findings of the disciplinary court were posted on the University's electronic notice board and prominently displayed in all public access computer facilities. The result was a sharp drop in circuit utilisation and a dramatic improvement in response times.

This respite was temporary, however: within eight months, utilisation was consistently above 95% during office hours, sometimes at 100%, and another investigation was undertaken. To the surprise of the investigators, there was no real evidence of abuse. A minority of users were still responsible for a majority of the traffic, but the material being transferred was large data sets that were integral to ongoing research. Coincidentally, a benchmarking exercise found that the University was purchasing only 60% of the bandwidth (adjusted for size) that equivalent peer institutions were purchasing. In light of this, The University Budget Committee agreed to release funds to increase the available capacity - but it also made it clear that it never would have made such an agreement unless it were also convinced that the University was no longer funding abuse.

Later that same year, researchers interviewing students and staff at Havensburg discovered that most members of the University community were satisfied with the speed of Internet access; most agreed with the University's acceptable user policy; most believed that they, as individuals, had a role to play in conserving bandwidth; most made a conscious effort to limit their own use of the Internet for private purposes. Most believed that any significant or sustained abuse would result in discovery, prosecution, and punishment. Very few were dissatisfied with this.

The moral of the story is that **Policy alone can't decongest a circuit**. But if applied vigorously, it can educate people, secure their support for limiting abuse, help to justify increases in expenditure that would otherwise never be supported, and sustain a culture of bandwidth conservation.

The policy development process

The policy development process is as important as the policy itself. The process is what will give the policy its validity and ensure that all members of the community understand why the policy is being developed, why the regulations exist, and will hopefully ensure user buy-in. Without an appropriate development process, a policy is likely to fail at some level.

The policy development process will be linked to the organisation's structure and culture. Some or all of the following issues should be considered.

- Understand your policy environment. Who has the authority to make policy? How can this authority be invoked?
- Understand your organisation's requirements for policy formulation and follow them. Are there specific consultation procedures that must be followed? Do specific committees or individuals need to give approval?
- Review the existing policy, if any exists. Consider conditions of service for staff policies on privacy. Any new policy should be in line with existing ones.
- Understand the wider legal environment. You cannot create policy that is in conflict with your legal system or your labour relations protocols. Some aspects of national law may have to be included in your policy (e.g., controls on access to pornography).
- Document the problem you're trying to solve and why policy is necessary to solve it. It can be useful to discuss the alternatives regarding improper use of the network and the limitations associated with it. This way, people see the need for the policy. **Why is policy necessary at all?** This is the most fundamental issue, and the message needs to be transmitted with absolute clarity.
- Document usage patterns. Typically, 5% of users account for 50% of the traffic. The other 95% of users should be on your side once they realise how they will benefit from the policy
- Document what has already been done to manage bandwidth problems. People are much more likely to be sympathetic if they believe that further policy and regulation are essential to improving their Internet access.
- Benchmark. If other institutions in the same class use policy as an instrument of bandwidth management, then mention this. It provides context and can be useful in competitive environments. (If other institutions are implementing specific policy then shouldn't we?)
- Identify who will support the policy and who might object. This will help you plan your response to objections as the policy is implemented. The documented usage patterns should be useful here.
- Identify the policy development team. It should include powerful figures who carry weight in the organisation. The chairs or deans of other departments might benefit the credibility of the developed policy, by being seen as independent of the Information Technology department.
- Communicate with your users. The policy development team needs to consult as extensively as possible with those who will be using the network. The consultation process is also a process for mobilising consensus concerning usage policies. Produce drafts of regulations and consult widely.
- Take time to navigate the policy approval process. Depending on the organisation, this may take a while.

- Plan for early wins. The process often raises plenty of expectations, so some tangible benefit should be delivered as soon as possible. This will show that progress is being made while broader changes are implemented.
- Make sure that the IT department is technically capable of doing whatever the policy will require.
- Enforcement is not the sole responsibility of the IT department. It must be supported by other processes, organizational structures, and ultimately the users themselves. Whatever the situation, the policy must be enforced, not because it is policy, but because the users recognise that it exists for the good of the network.
- Review the policy at set intervals. For example, create a schedule for policy review at three months after implementation and a year after implementation. Thereafter, repeat as necessary.
- Be proud of your results. Good results, when well advertised, are likely to help win over even the strongest opponents of the policy.

Policy is needed in all environments

Policies that guide bandwidth usage are not only the domain of low bandwidth environments. They are also an essential component of high speed networks. Experiences with very high speed networks show that, without policies and technical solutions, even multi-gigabyte (Gb) circuits can become congested and encounter degradations in performance. It was recently reported that up to half of the bandwidth at Finnish universities is used for downloading movies, music, and software. The network at Carnegie Mellon approached a gigabit of consumption before measures were taken to enforce an acceptable use policy.

In addition, there are very few contexts in which policy can be dispensed with entirely. People using a network affect other people's machines, whether they are in the same organisation or outside it. If users are handling corporate data of any kind, there are risks concerning loss, unauthorised modification, or unintended disclosure of sensitive or proprietary information. Therefore, some kind of policy is needed in order to manage those risks.

In general, you need policy to manage three specific kinds of risks: (**a**) risks arising from potential abuse, such as the excessive consumption of bandwidth; (**b**) risks arising from potential liability, arising out of things that users might do on networks (such as posting inflammatory or libelous remarks about other people); and (**c**) risks that arise out of a failure to comply with governmental regulations. These risks will vary considerably from one country to another, but there are very few contexts where they are completely absent.

Policy pitfalls

Your greatest danger lies in producing a vacuous policy - that is, a policy that is devoid of meaningful content. Policy must live in the heads of people, since its purpose is to shape or channel their behaviour. If it fails to do this, then it is a dead letter. Some examples of vacuous policy include:

- **Policy that is not backed by monitoring.** Ensure that you have the technical capability to monitor your network before you finalise policy. You should really have this ability at the start of the policy development process, since having a sense of the actual traffic is essential in order to build a realistic and relevant policy.
- **Policy that is unduly complex, or couched in legalistic language.** Policy is made for people, and needs to be kept focussed and readily understandable.
- **Policy that doesn't fit your environment, because it has been cut and pasted from somewhere else.** It's always best to write a policy from scratch and mobilise consent as you do so.
- **Policy that is not enforced, because of a lack of political will.** Unenforced policy is even worse than no policy at all, because it's much harder to reinvigorate a failed policy than it is to start a completely new policy process.
- **Unofficial policy.** Policy that does not have the backing of decision making structures of the institution, or that has been implemented in isolation, will be difficult to implement and will lack "teeth." When an unofficial policy arises that is in conflict with an approved "official" version, authority is undermined and users will choose to follow the rules that suit them.

Example policies

The following links provide good examples of issues covered by policy documents. Every organisation is unique and should develop policy that meets its own needs. The documents below can be useful when you reach the drafting stage of policy development, but you should never be tempted to skip the other stages – the process of creating workshops and consulting with community, concerning policy, is what educates them and secures their buy-in. You can often learn surprisingly important things from the user community regarding their needs. If you use someone else’s documents during drafting, you should resist the temptation to cut and paste from them wholesale. Even the most generic policy needs some localisation. Editing existing policies invites inconsistency with your own network and how your community will use it. It’s always best to write a policy rather than to copy one.

- The SANS institute policy template page:
  http://www.sans.org/resources/policies/#template
- A listing of policy examples from universities in the United States:
  http://ndsl.lib.state.nd.us/AcceptableUseExp.html
- The University of Cape Town's **Policy and rules on Internet and Email use** is a short policy that exhibits many key characteristics:
  http://www.icts.uct.ac.za/modules.php?name=News&file=print&sid=633
- Here is a longer policy that also includes most of the key characteristics: the University of KwaZulu-Natal's **ELECTRONIC COMMUNICATIONS POLICY**: *http://www.nu.ac.za/itd/policies/ecommunications.pdf*

Policy checklist

The two checklists that follow are provided to help with the development and implementation of effective policies to support bandwidth management and optimisation. Before you get started on this process though, make sure that you have documented the problem you're trying to solve (and why policy is necessary to solve it). You should also document usage patterns that support your case (see chapter three, **Monitoring & Analysis**).

Once you have done that, you should have a good sense of the nature of the problem from a social and technical point of view. You are now ready to start the policy development process (although, in reality, you will already have started it!). Remember, the policy development process is just as important as the policy it produces.

The policy development process checklist

✓ Understand your policy environment

✓ Understand your organisation's requirements for policy formulation and follow them

✓ Review existing policy

✓ Understand the wider legal environment

✓ Document what has already been done to manage the bandwidth problem

✓ Benchmark

✓ Identify who supports policy, and who doesn't

✓ Identify the policy development team

✓ Communicate with your users to understand their network experiences

- ✓ Produce a draft for consultation and consult widely
- ✓ Navigate the policy approval process
- ✓ Plan for early wins
- ✓ Ensure implementation and enforcement
- ✓ Gather feedback about network performance and policy requirements
- ✓ Periodically review the policy

Of course, a process is useless unless it produces an effective policy document and environment at the end. Be sure your policy exhibits all of the key characteristics found below.

Characteristics of good policy checklist

- ✓ Good policy has an enabling purpose
- ✓ Good policy is linked to a wider objective
- ✓ Good policy has clear ownership
- ✓ Good policy is short and clear
- ✓ Good policy arises from a valid process
- ✓ Good policy works within the confines of a given authority
- ✓ Good policy is enforced
- ✓ Good policy is adaptable

Once you have checked off all of the above, you will have a policy that provides an effective framework for bandwidth management and optimisation while having carefully considered the needs of your community.

References

- Illegal software and film downloads exhaust university computer networks, *http://www.hs.fi/english/article/1101978960379*
- Carnegie Mellon University case study, page **248**.
- INASP Bandwidth management and optimisation: policy development workshop, *http://www.inasp.info/training/bandwidth/bmo-pdw/*

Sample policy collections

- Educause collation on Acceptable/Responsible Use Policies: EDUCAUSE is a nonprofit association whose mission is to advance higher education by

promoting the intelligent use of information technology, *http://www.educause.edu/content.asp?page_id=645&PARENT_ID=110&bhcp=1*

- Examples Internet Acceptable Use Policies: a large collection of example policies, mainly from US organisations. Including; Internet Acceptable Use Policies for Public Libraries; Internet Acceptable Use Policies for School Library Media Centers; Internet Acceptable Use Policies for Colleges and Universities, *http://ndsl.lib.state.nd.us/AcceptableUseExp.html*
- SANS Security Policy Resource page, a consensus research project of the SANS community. The ultimate goal of the project is to offer everything you need for rapid development and implementation of information security policies. You'll find a great set of resources posted here already including policy templates for twenty-four important security requirements, *http://www.sans.org/resources/policies/*
- Tech Republic: A framework for e-mail and Internet usage policies for your enterprise, *http://articles.techrepublic.com.com/5102-6299-1033914.html*

3 Monitoring & Analysis

There's an old saying which applies to bandwidth management: "You can't manage it until you measure it." If your Internet connection is saturated with so much traffic that it makes your daily browsing seem like a trip to the dentist, you need to take a serious look at what is going down that pipe. Once you have a complete understanding of how your Internet connection is being used, it will become clear which course of action needs to be taken in order to fix the problem.

Without the insight that good monitoring tools and techniques provide, you cannot understand the effects that changes will make. Trying to fix network problems, without first establishing a clear picture of what is happening, is a lot like trying to fix a car engine by knocking on various parts with a hammer. You might get lucky and knock something into place that gets the car going again (for the moment), but you will inevitably run into more problems later. In the process of knocking on some parts, it's likely you will cause unintended damage to other parts of the engine.

Bandwidth management is not a dark art or a mystic philosophy; it is a methodical technique of problem identification, analysis, and resolution. By monitoring the performance of your network, and analysing the resulting data over time, you will be able to make effective changes that solve performance problems, yielding measurable improvements.

Before we can answer the question of where the network bottlenecks lie, we need to understand how the network works. Once we understand what makes information flow from here to there, we will have a better idea of what to look out for when that flow is not as fast as we would like it to be.

Networking 101

If you are already comfortable with the essentials of TCP/IP networking (including addressing, routing, switches, firewalls, and routers), you may want to skip ahead to **What is Network Monitoring?** on page **62**. We will now review the basics of Internet networking.

Introduction

Venice, Italy is a fantastic city to get lost in. The roads are mere foot paths that cross water in hundreds of places, and never go in a simple straight line. Postal carriers in Venice are some of the most highly trained in the world, specialising in delivery to only one or two of the six *sestieri* (districts) of Venice. This is necessary due to the intricate layout of that ancient city. Many people find that knowing the location of the water and the sun is far more useful than trying to find a street name on a map.

Figure 3.1: Another kind of network mask.

Just after the book development team met to formalize the outline for this book, a few of us spent a couple of days in Venice. One of us happened to find a particularly beautiful papier-mâché mask, and wanted to have it shipped from the studio in S. Polo, Venezia to an office in Seattle, USA. This may sound like an ordinary (or even trivial) task, but let's look at what actually happened.

The artist packed the mask into a shipping box and addressed it to the office in Seattle, USA. They then handed this off to a postal employee, who attached some official forms and sent it to a central package processing hub for international destinations. After several days, the package cleared Italian customs and found its way onto a transatlantic flight, arriving at a central import processing location in the U.S. Once it was cleared through U.S. customs, the package was sent to the regional distribution point for the northwest U.S., then on to the Seattle postal processing centre. The package eventually made its way onto a delivery van which had a route that brought it to the proper address, on the proper street, in the proper neighborhood. A clerk at the office accepted the package and put it in the proper incoming mail box. Once it arrived, the box was retrieved and the mask itself was finally received.

The clerk at the office neither knows nor cares about how to get to the *sistiere* of S. Polo, Venezia. His job is simply to accept packages as they arrive, and deliver them to the proper person. Similarly, the postal carrier in Venice has no need to worry about how to get to the correct neighborhood in Seattle. His job is to pick up packages from his local neighborhood and forward them to the next closest hub in the delivery chain.

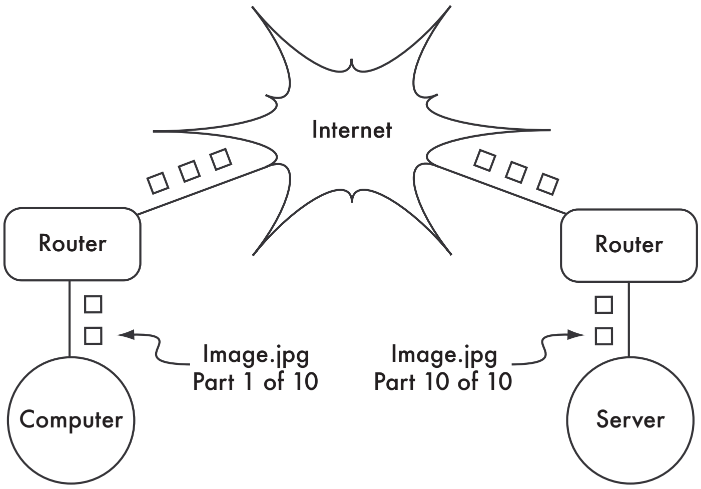


Figure 3.2: Internet networking. Packets are forwarded between routers until they reach their ultimate destination.

This is very similar to how Internet routing works. A particular message is split up into many individual ***packets***, and are labeled with their source and destination. The computer then sends these packets to a ***router***, which decides where to send them next. The router needs only to keep track of a handful of routes (for example, how to get to the local network, the best route to a few other local networks, and one route to a gateway to the rest of the Internet). This list of possible routes is called the ***routing table***. As packets arrive at the router, the destination address is examined and compared against its internal routing table. If the router has no explicit route to the destination in question, it sends the packet to the closest match it can find, which is often its own Internet gateway (via the default route). And the next router does the same, and so forth, until the packet eventually arrives at its destination.

Packages can only make their way through the international postal system because we have established a standardised addressing scheme for packages. For example, the destination address must be written legibly on the front of the package, and include all critical information (such as the recipient's name, street address, city, country, and postal code). Without this information, packages are either returned to the sender or are lost in the system.

Packets can only flow through the global Internet because we have agreed on a common addressing scheme and protocol for forwarding packets. These standard communication protocols make it possible to exchange information on a global scale.

Cooperative communications

Communication is only possible when the participants speak a common language. But once the communication becomes more complex than a simple conversation between two people, protocol becomes just as important as language. All of the people in an auditorium may speak English, but without a set of rules in place to establish who has the right to use the microphone, the communication of an individual’s ideas to the entire room is nearly impossible. Now imagine an auditorium as big as the world, full of all of the computers that exist. Without a common set of communication protocols to regulate when and how each computer can speak, the Internet would be a chaotic mess where every machine tries to speak at once.

Of course, people have developed a number of communications frameworks to address this problem. The most well-known of these is the ***OSI model***.

The OSI model

The international standard for Open Systems Interconnection (OSI) is defined by the document ISO/IEC 7498-1, as outlined by the International Standards Organisation and the International Electrotechnical Commission. The full standard is available as publication "ISO/IEC 7498-1:1994," available from *http://standards.iso.org/ittf/PubliclyAvailableStandards/*.

The OSI model divides network traffic into a number of ***layers***. Each layer is independent of the layers around it, and each builds on the services provided by the layer below while providing new services to the layer above. The abstraction between layers makes it easy to design elaborate and highly reliable ***protocol stacks***, such as the ubiquitous ***TCP/IP*** stack. A protocol stack is an actual implementation of a layered communications framework. The OSI model doesn't define the protocols to be used in a particular network, but simply delegates each communications "job" to a single layer within a well-defined hierarchy.

While the ISO/IEC 7498-1 specification details how layers should interact with each other, it leaves the actual implementation details up to the manufacturer. Each layer can be implemented in hardware (more common for lower layers) or software. As long as the interface between layers adheres to the standard, implementers are free to use whatever means are available to build their protocol stack. This means that any given layer from manufacturer A can operate with the same layer from manufacturer B (assuming the relevant specifications are implemented and interpreted correctly).

Here is a brief outline of the seven-layer OSI networking model:

Layer	Name	Description
7	Application	The ***Application Layer*** is the layer that most network users are exposed to, and is the level at which human communication happens. HTTP, FTP, and SMTP are all application layer protocols. The human sits above this layer, interacting with the application.
6	Presentation	The ***Presentation Layer*** deals with data representation, before it reaches the application. This would include MIME encoding, data compression, formatting checks, byte ordering, etc.
5	Session	The ***Session Layer*** manages the logical communications session between applications. NetBIOS and RPC are two examples of a layer five protocol.
4	Transport	The ***Transport Layer*** provides a method of reaching a particular service on a given network node. Examples of protocols that operate at this layer are TCP and UDP. Some protocols at the transport layer (such as TCP) ensure that all of the data has arrived at the destination, and is reassembled and delivered to the next layer in the proper order. UDP is a "connectionless" protocol commonly used for video and audio streaming.

Layer	Name	Description
3	Network	IP (the Internet Protocol) is the most common ***Network Layer*** protocol. This is the layer where routing occurs. Packets can leave the link local network and be retransmitted on other networks. Routers perform this function on a network by having at least two network interfaces, one on each of the networks to be interconnected. Nodes on the Internet are reached by their globally unique IP address. Another critical Network Layer protocol is ICMP, which is a special protocol which provides various management messages needed for correct operation of IP. This layer is also sometimes referred to as the ***Internet Layer***.
2	Data Link	Whenever two or more nodes share the same physical medium (for example, several computers plugged into a hub, or a room full of wireless devices all using the same radio channel) they use the ***Data Link Layer*** to communicate. Common examples of data link protocols are Ethernet, Token Ring, ATM, and the wireless networking protocols (802.11a/b/g). Communication on this layer is said to be link-local, since all nodes connected at this layer communicate with each other directly. This layer is sometimes known as the ***Media Access Control*** (***MAC***) layer. On networks modeled after Ethernet, nodes are referred to by their ***MAC address.*** This is a unique 48 bit number assigned to every networking device when it is manufactured.
1	Physical	The ***Physical Layer*** is the lowest layer in the OSI model, and refers to the actual physical medium over which communications take place. This can be a copper CAT5 cable, a fibre optic bundle, radio waves, or just about any other medium capable of transmitting signals. Cut wires, broken fibre, and RF interference are all physical layer problems.

The layers in this model are numbered one through seven, with seven at the top. This is meant to reinforce the idea that each layer builds upon, and depends upon, the layers below. Imagine the OSI model as a building, with the foundation at layer one, the next layers as successive floors, and the roof at layer seven. If you remove any single layer, the building will not stand. Similarly, if the fourth floor is on fire, then nobody can pass through it in either direction.

The first three layers (Physical, Data Link, and Network) all happen "on the network." That is, activity at these layers is determined by the configuration of cables, switches, routers, and similar devices. A network switch can only distribute packets by using MAC addresses, so it need only implement layers one and two. A simple router can route packets using only their IP addresses, so it need implement only layers one through three. A web server or a laptop computer runs applications, so it must implement all seven layers. Some advanced routers may implement layer four and above, to allow them to make decisions based on the higher-level information content in a packet, such as the name of a website, or the attachments of an email.

The OSI model is internationally recognised, and is widely regarded as the complete and definitive network model. It provides a framework for manufacturers and network protocol implementers that can be used to build networking devices that interoperate in just about any part of the world.

From the perspective of a network engineer or troubleshooter, the OSI model can seem needlessly complex. In particular, people who build and troubleshoot TCP/IP networks rarely need to deal with problems at the Session or Presentation layers. For the majority of Internet network implementations, the OSI model can be simplified into a smaller collection of five layers.

The TCP/IP model

Unlike the OSI model, the TCP/IP model is not an international standard and its definitions vary. Nevertheless, it is often used as a pragmatic model for understanding and troubleshooting Internet networks. The vast majority of the Internet uses TCP/IP, and so we can make some assumptions about networks that make them easier to understand. The TCP/IP model of networking describes the following five layers:

Layer	Name
5	Application
4	Transport
3	Internet
2	Data Link
1	Physical

In terms of the OSI model, layers five through seven are rolled into the topmost layer (the Application layer). The first four layers in both models are identical.

Many network engineers think of everything above layer four as "just data" that varies from application to application. Since the first three layers are interoperable between virtually all manufacturers' equipment, and layer four works between all hosts using TCP/IP, and everything above layer four tends to apply to specific applications, this simplified model works well when building and troubleshooting TCP/IP networks. We will use the TCP/IP model when discussing networks in this book.

The TCP/IP model can be compared to a person delivering a letter to a downtown office building. The person first needs to interact with the road itself (the Physical layer), pay attention to other traffic on the road (the Data Link layer), turn at the proper place to connect to other roads and arrive at the correct address (the Internet layer), go to the proper floor and room number (the Transport layer), and finally give it to a receptionist who can take the letter from there (the Application layer). Once they have delivered the message to the receptionist, the delivery person is free to go on their way.

The five layers can be easily remembered by using the mnemonic "**P**lease **D**on't **L**ook **I**n **T**he **A**ttic," which of course stands for "**P**hysical / **D**ata **L**ink / **I**nternet / **T**ransport / **A**pplication."

The Internet protocols

TCP/IP is the protocol stack most commonly used on the global Internet. The acronym stands for ***Transmission Control Protocol*** (***TCP***) and ***Internet Protocol*** (***IP***), but actually refers to a whole family of related communications protocols. TCP/IP is also called the ***Internet protocol suite***, and it operates at layers three and four of the TCP/IP model.

In this discussion, we will focus on version four of the IP protocol (IPv4) as this is now the most widely deployed protocol on the Internet. What follows is a brief overview of the critical aspects of TCP/IP that are needed in order to understand network utilisation. For a more thorough treatment of this complex subject, see the resources at the end of this chapter.

IP Addressing

In an IPv4 network, the address is a 32-bit number, normally written as four 8-bit numbers expressed in decimal form and separated by periods. Examples of IP addresses are 10.0.17.1, 192.168.1.1, or 172.16.5.23.

If you enumerated every possible IP address, they would range from 0.0.0.0 to 255.255.255.255. This yields a total of more than four billion possible IP addresses (255 * 255 * 255 * 255 = 4 228 250 625); although many of these are reserved for special purposes and cannot be assigned to hosts. Each of the usable IP addresses is a unique identifier that distinguishes one network node from another.

Interconnected networks must agree on an IP addressing plan. IP addresses must be unique and cannot be used in different places on the Internet at the same time; otherwise, routers would not know how best to route packets to them.

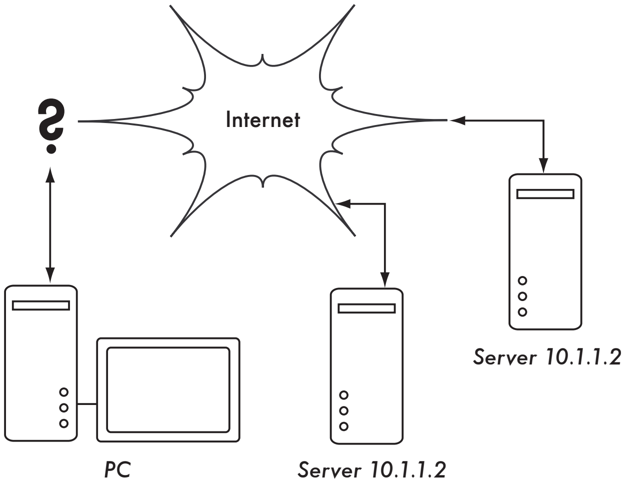


Figure 3.3: Without unique IP addresses, unambiguous global routing is impossible. If the PC requests a web page from 10.1.1.2, which server will it reach?

IP addresses are allocated by a central numbering authority, which provides a consistent and coherent numbering method. This ensures that duplicate addresses are not used by different networks. The authority assigns large blocks of consecutive addresses to smaller authorities, who in turn assign smaller consecutive blocks within these ranges to other authorities, or to their customers. These groups of addresses are called sub-networks, or ***subnets*** for short. Large subnets can be further subdivided into smaller subnets. A group of related addresses is referred to as an ***address space***.

Subnets

By applying a ***subnet mask*** (also called a ***network mask***, or simply ***netmask***) to an IP address, you can logically define both a host and the network to which it belongs. Traditionally, subnet masks have been expressed using dotted decimal form, much like an IP address. For example, 255.255.255.0 is one common netmask. You will find this notation used when configuring network interfaces, creating routes, etc. However, subnet masks are more succinctly expressed using ***CIDR notation***, which simply enumerates the number of bits in the mask after a forward slash (/). Thus, 255.255.255.0 can be simplified as

/24. CIDR is short for ***Classless Inter-Domain Routing***, and is defined in RFC1518*.

A subnet mask determines the size of a given network. Using a /24 netmask, 8 bits are reserved for hosts (32 bits total - 24 bits of netmask = 8 bits for hosts). This yields up to 256 possible host addresses ($2^8 = 256$). By convention, the first value is taken as the ***network address*** (.0 or 00000000), and the last value is taken as the ***broadcast address*** (.255 or 11111111). This leaves 254 addresses available for hosts on this network.

Subnet masks work by applying AND logic to the 32 bit IP number. In binary notation, the "1" bits in the mask indicate the network address portion, and "0" bits indicate the host address portion. A logical AND is performed by comparing two bits. The result is "1" if both of the bits being compared are also "1". Otherwise the result is "0". Here are all of the possible outcomes of a binary AND comparison between two bits.

Bit 1	Bit 2	Result
0	0	0
0	1	0
1	0	0
1	1	1

To understand how a netmask is applied to an IP address, first convert everything to binary. The netmask 255.255.255.0 in binary contains twenty-four "1" bits:

```
255      255      255      0
11111111.11111111.11111111.00000000
```

When this netmask is combined with the IP address 10.10.10.10, we can apply a logical AND to each of the bits to determine the network address.

```
  10.10.10.10: 00001010.00001010.00001010.00001010
255.255.255.0: 11111111.11111111.11111111.00000000
               -----------------------------------
   10.10.10.0: 00001010.00001010.00001010.00000000
```

* RFC is short for Request For Comments. RFCs are a numbered series of documents published by the Internet Society that document ideas and concepts related to Internet technologies. Not all RFCs are actual standards. RFCs can be viewed online at http://rfc.net/

This results in the network 10.10.10.0/24. This network consists of the hosts 10.10.10.1 through 10.10.10.254, with 10.10.10.0 as the network address and 10.10.10.255 as the broadcast address.

Subnet masks are not limited to entire octets. One can also specify subnet masks like 255.254.0.0 (or /15 CIDR). This is a large block, containing 131,072 addresses, from 10.0.0.0 to 10.1.255.255. It could be further subdivided, for example into 512 subnets of 256 addresses each. The first one would be 10.0.0.0-10.0.0.255, then 10.0.1.0-10.0.1.255, and so on up to 10.1.255.0-10.1.255.255. Alternatively, it could be subdivided into 2 blocks of 65,536 addresses, or 8192 blocks of 16 addresses, or in many different ways. It could even be subdivided into a mixture of different block sizes, as long as none of them overlap, and each is a valid subnet whose size is a power of two.

While many netmasks are possible, common netmasks include:

CIDR	Decimal	# of Hosts
/30	255.255.255.252	4
/29	255.255.255.248	8
/28	255.255.255.240	16
/27	255.255.255.224	32
/26	255.255.255.192	64
/25	255.255.255.128	128
/24	255.255.255.0	256
/16	255.255.0.0	65 536
/8	255.0.0.0	16 777 216

With each reduction in the CIDR value the IP space is doubled. Remember that two IP addresses within each network are always reserved for the network address and broadcast address.

There are three common netmasks that have special names. A /8 network (with a netmask of 255.0.0.0) defines a ***Class A*** network. A /16 (255.255.0.0) is a ***Class B***, and a /24 (255.255.255.0) is called a ***Class C***. These names were around long before CIDR notation, but are still often used for historical reasons.

Global IP Addresses

Have you ever wondered who controls the allocation of IP space? ***Globally routable IP addresses*** are assigned and distributed by ***Regional Internet Registrars*** (***RIR***s) to ISPs. The ISP then allocates smaller IP blocks to their clients as required. Virtually all Internet users obtain their IP addresses from an ISP.

The 4 billion available IP addresses are administered by the ***Internet Assigned Numbers Authority*** (***IANA***, *http://www.iana.org/*). IANA has divided this space into large subnets, usually /8 subnets with 16 million addresses each. These subnets are delegated to one of the five regional Internet registries (RIRs), which are given authority over large geographic areas.

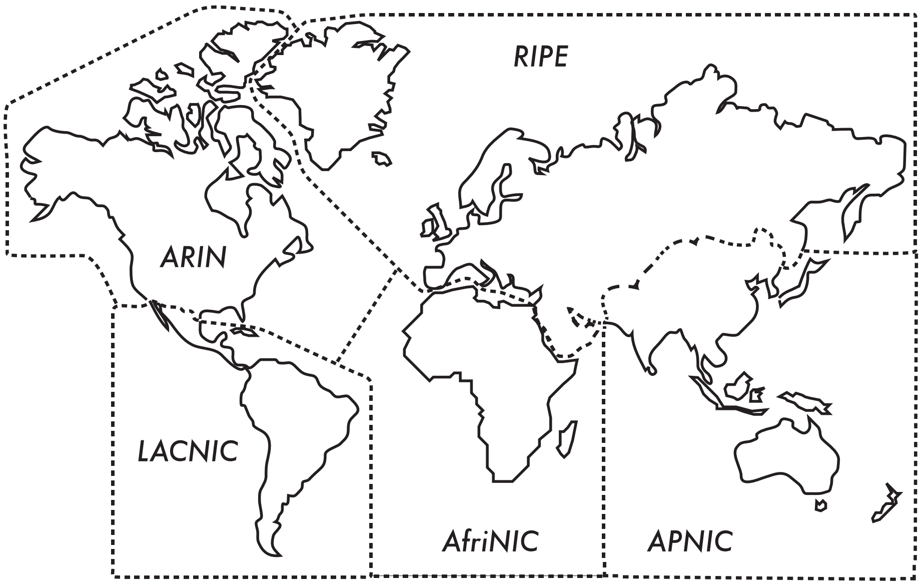


Figure 3.4: Authority for Internet IP address assignments is delegated to the five Regional Internet Registrars.

The five RIRs are:

- African Network Information Centre (AfriNIC, *http://www.afrinic.net/*)
- Asia Pacific Network Information Centre (APNIC, *http://www.apnic.net/*)
- American Registry for Internet Numbers (ARIN, *http://www.arin.net/*)
- Regional Latin-American and Caribbean IP Address Registry (LACNIC, *http://lacnic.net/*)
- Réseaux IP Européens (RIPE NCC, *http://www.ripe.net/*)

Your ISP will assign globally routable IP address space to you from the pool allocated to it by your RIR. The registry system assures that IP addresses are not reused in any part of the network anywhere in the world.

Once IP address assignments have been agreed upon, it is possible to pass packets between networks and participate in the global Internet. The process of moving packets between networks is called ***routing***.

Static IP Addresses

A static IP address is an address assignment that never changes. Static IP addresses are important because servers using these addresses may have DNS mappings pointed towards them, and typically serve information to other machines (such as email services, web servers, etc.).

Blocks of static IP addresses may be assigned by your ISP, either by request or automatically depending on your means of connection to the Internet.

Dynamic IP Addresses

Dynamic IP addresses are assigned by an ISP for non-permanent nodes connecting to the Internet, such as a home computer which is on a dial-up connection.

Dynamic IP addresses can be assigned automatically using the ***Dynamic Host Configuration Protocol*** (***DHCP***), or the ***Point-to-Point Protocol*** (***PPP***), depending on the type of Internet connection. A node using DHCP first requests an IP address assignment from the network, and automatically configures its network interface. IP addresses can be assigned randomly from a pool by your ISP, or might be assigned according to a policy. IP addresses assigned by DHCP are valid for a specified time (called the ***lease time***). The node must renew the DHCP lease before the lease time expires. Upon renewal, the node may receive the same IP address or a different one from the pool of available addresses.

Dynamic addresses are popular with Internet service providers, because it enables them to have fewer IP addresses than their total number of customers. They only need an address for each customer who is **active at any one time**. Globally routable IP addresses cost money, and some authorities that specialise in the assignment of addresses (such as RIPE, the European RIR) are very strict on IP address usage for ISP's. Assigning addresses dynamically enables ISPs to save money, and normally they will charge extra for a static IP address.

Private IP addresses

Most private networks do not require the allocation of a globally routable, public IP addresses for every computer in the organisation. In particular, computers which are not public servers do not need to be addressable from the public Internet. Organisations typically use IP addresses from the ***private address space*** for machines on the internal network.

There are currently three blocks of private address space reserved by IANA: 10.0.0.0/8, 172.16.0.0/12, and 192.168.0.0/16. These are defined in RFC1918. These addresses are not intended to be routed on the Internet, and are typically unique only within an organisation or group of organisations which choose to follow the same numbering scheme.

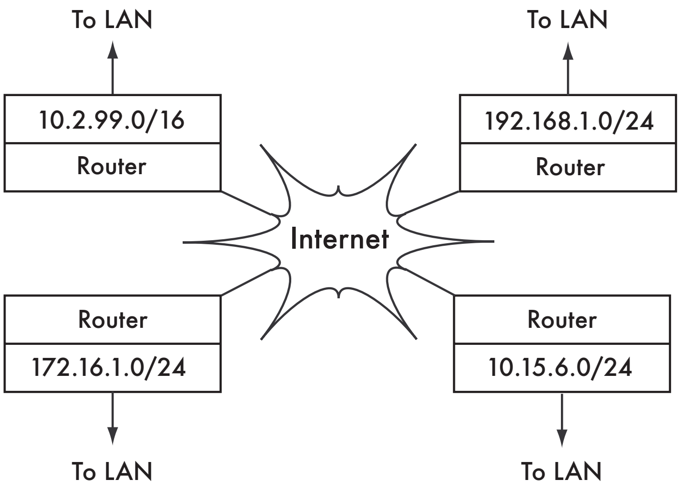


Figure 3.5: RFC1918 private addresses may be used within an organisation, and are not routed on the global Internet.

If you ever intend to link together private networks that use RFC1918 address space, be sure to use unique addresses throughout all of the networks. For example, you might break the 10.0.0.0/8 address space into multiple Class B networks (10.1.0.0/16, 10.2.0.0/16, etc.). One block could be assigned to each network according to its physical location (the campus main branch, field office one, field office two, dormitories, and so forth). The network administrators at each location can then break the network down further into multiple Class C networks (10.1.1.0/24, 10.1.2.0/24, etc.). or into blocks of any other logical size. In the future, should the networks ever be linked (either by a physical connection, wireless link, or VPN), then all of the machines will be reachable from any point in the network without having to renumber network devices.

Some Internet providers may allocate private addresses like these instead of public addresses to their customers, although this has serious disadvantages.

Since these addresses cannot be routed over the Internet, computers which use them are not really "part" of the Internet, and are not directly reachable from it. In order to allow them to communicate with the Internet, their private addresses must be translated to public addresses. This translation process is known as ***Network Address Translation*** (***NAT***), and is normally performed at the gateway between the private network and the Internet. We will look at NAT in more detail on page **42**.

Routing

Imagine a network with three hosts: A, B, and C. They use the corresponding IP addresses 192.168.1.1, 192.168.1.2 and 192.168.1.3. These hosts are part of a /24 network (their network mask is 255.255.255.0).

For two hosts to communicate on a local network, they must determine each others' MAC addresses. It is possible to manually configure each host with a mapping table from IP address to MAC address, but normally the ***Address Resolution Protocol*** (***ARP***) is used to determine this automatically.

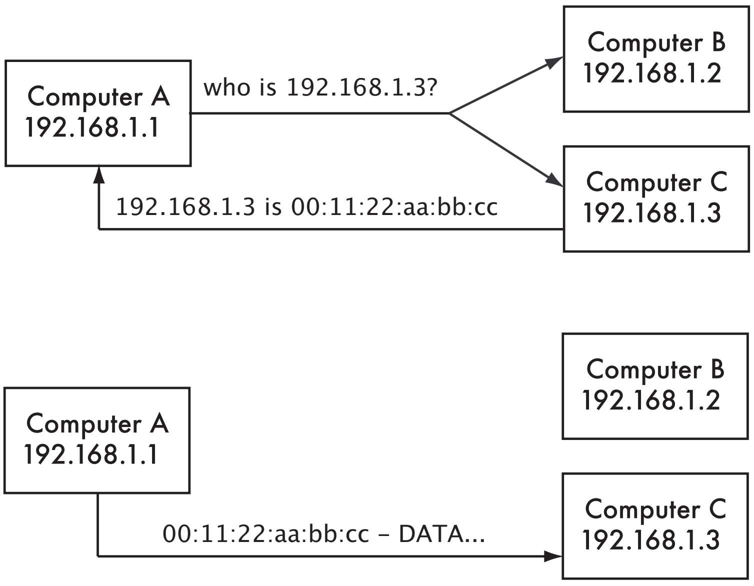


Figure 3.6: Computer A needs to send data to 192.168.1.3. But it must first ask the whole network for the MAC address that responds to 192.168.1.3.

When using ARP, host A broadcasts to all hosts the question, "Who has the MAC address for the IP 192.168.1.3?" When host C sees an ARP request for its own IP address, it replies with its MAC address.

Consider now another network with 3 hosts, D, E, and F, with the corresponding IP addresses 192.168.2.1, 192.168.2.2, and 192.168.2.3. This is another

/24 network, but it is not in the same range as the network above. All three hosts can reach each other directly (first using ARP to resolve the IP address into a MAC address, and then sending packets to that MAC address).

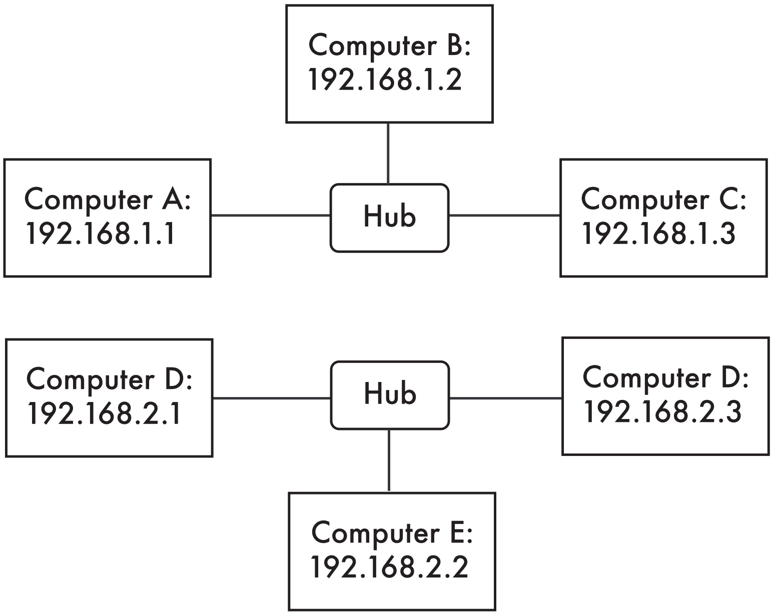


Figure 3.7: Two separate IP networks.

Now we will add host G. This host has two network cards, with one plugged into each network. The first network card uses the IP address 192.168.1.4, and the other uses 192.168.2.4. Host G is now link-local to both networks, and can route packets between them.

But what if hosts A, B, and C want to reach hosts D, E, and F? They will need to add a route to the other network via host G. For example, hosts A-C would add the following route:

```
# ip route add 192.168.2.0/24 via 192.168.1.4
```

...and hosts D-F would add the following:

```
# ip route add 192.168.1.0/24 via 192.168.2.4
```

(These examples use the Linux syntax for manipulating routes, which varies according to your operating system). The result is shown in Figure 3.8. Notice that the route is added via the IP address on host G that is link-local to the respective network. Host A could not add a route via 192.168.2.4, even though it is the same physical machine as 192.168.1.4 (host G), since that IP is not link-local.

A route tells the OS that the desired network doesn't lie on the immediate link-local network, and it must ***forward*** the traffic through the specified router. If host A wants to send a packet to host F, it would first send it to host G. Host G would then look up host F in its routing table, and see that it has a direct connection to host F's network. Finally, host G would resolve the hardware (MAC) address of host F and forward the packet to it.

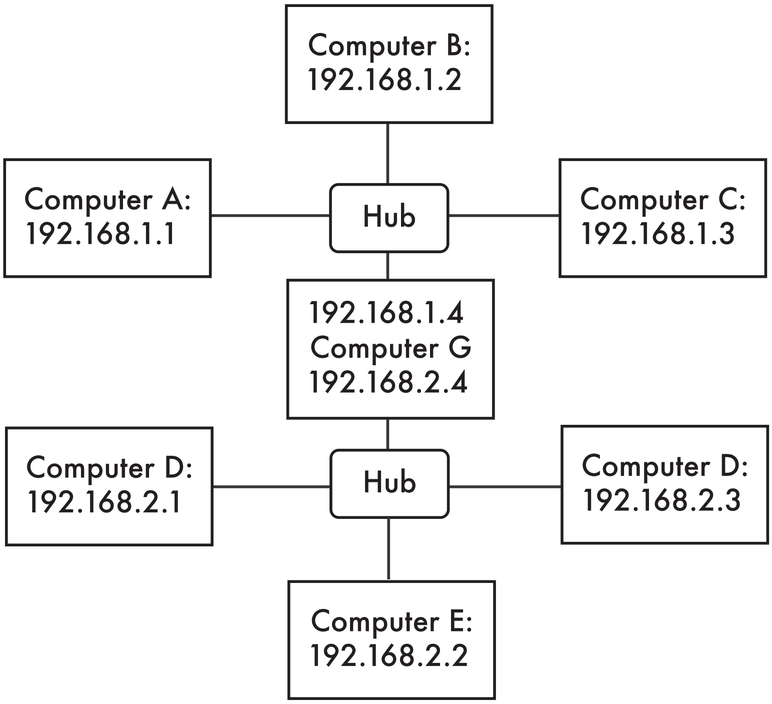
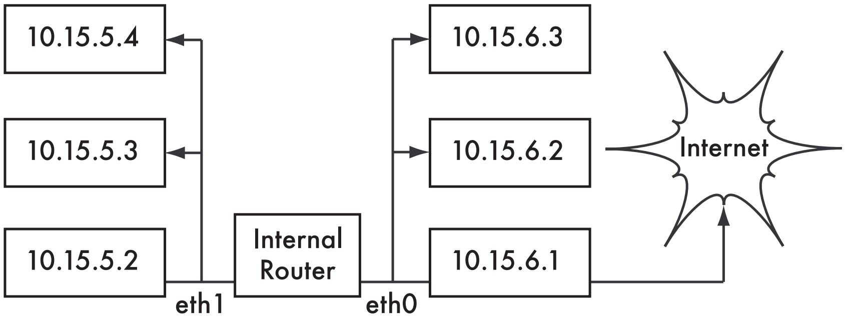


Figure 3.8: Host G acts as a router between the two networks.

This is a very simple routing example, where the destination is only a single ***hop*** away from the source. As networks get more complex, many hops may need to be made to reach the ultimate destination. Since it isn't practical for every machine on the Internet to know the route to every other, we make use of a routing entry known as the ***default route*** (also known as the ***default gateway***). When a router receives a packet destined for a network for which it has no explicit route, the packet is forwarded to its default gateway.

The default gateway is typically the best route out of your network, usually in the direction of your ISP. An example of a router that uses a default gateway is shown in Figure 3.9.



```
Routing table for internal router:

Destination      Gateway     Genmask          Flags  Metric  Iface
10.15.5.0        *           255.255.255.0    U      0       eth1
10.15.6.0        *           255.255.255.0    U      0       eth0
default          10.15.6.1   0.0.0.0          UG     0       eth0
```

Figure 3.9: When no explicit route exists to a particular destination, a host uses the default gateway entry in its routing table.

Routes can be updated manually, or can dynamically react to network outages and other events. Some examples of popular dynamic routing protocols are RIP, OSPF, BGP, and OLSR. Configuring dynamic routing is beyond the scope of this book, but for further reading on the subject, see the resources at the end of this chapter.

Network Address Translation (NAT)

In order to reach hosts on the Internet, RFC1918 addresses must be converted to global, publicly routable IP addresses. This is achieved using a technique known as ***Network Address Translation***, or ***NAT***. A NAT device is a router that manipulates the addresses of packets instead of simply forwarding them. On a NAT router, the Internet connection uses one (or more) globally routed IP addresses, while the private network uses an IP address from the RFC1918 private address range. The NAT router allows the global address(es) to be shared with all of the inside users, who all use private addresses from the same range. It converts the packets from one form of addressing to the other as the packets pass through it. As far as the network users can tell, they are directly connected to the Internet and require no special software or drivers. They simply use the NAT router as their default gateway, and address packets as they normally would. The NAT router translates outbound packets to use the global IP address as they leave the network, and translates them back again as they are received from the Internet.

The major consequence of using NAT is that machines from the Internet cannot easily reach servers within the organisation without setting up explicit forwarding rules on the router. Connections initiated from within the private address space generally have no trouble, although some applications (such as Voice over IP and some VPN software) can have difficulty dealing with NAT.

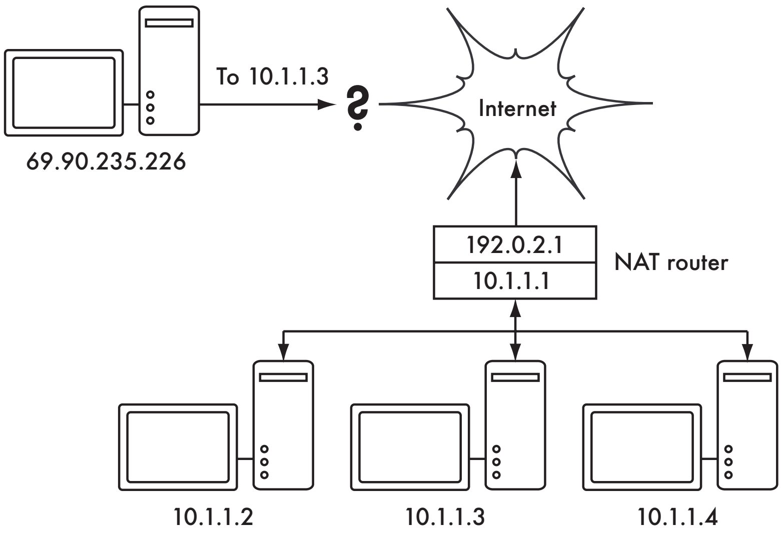


Figure 3.10: Network Address Translation allows you to share a single IP address with many internal hosts, but can make it difficult for some services to work properly.

Depending on your point of view, this can be considered a bug (since it makes it harder to set up two-way communication) or a feature (since it effectively provides a "free" firewall for your entire organisation). RFC1918 addresses should be filtered on the edge of your network to avoid accidental or intentional RFC1918 traffic entering or leaving your network. While NAT performs some firewall-like functions, it is not a replacement for a real firewall.

Internet Protocol Suite

Machines on the Internet use the Internet Protocol (IP) to reach each other, even when separated by many intermediary machines. There are a number of protocols that are run in conjunction with IP that provide features as critical to normal operations as IP itself. Every packet specifies a protocol number which identifies the packet as one of these protocols. The most commonly used protocols are the ***Transmission Control Protocol*** (***TCP***, number 6), ***User Datagram Protocol*** (***UDP***, number 17), and the ***Internet Control Message Protocol*** (***ICMP***, number 1). Taken as a group, these protocols (and others) are known as the ***Internet Protocol Suite***, or simply ***TCP/IP*** for short.

The TCP and UDP protocols introduce the concept of port numbers. These allow multiple services to be run on the same IP address, and still be distinguished from each other. Every packet has a source and destination port number. Some port numbers are well defined standards, used to reach well known services such as email and web servers. For example, web servers normally ***listen*** on TCP port 80, and SMTP email servers listen on TCP port 25. When we say that a service "listens" on a port (such as port 80), we mean that it will accept packets that use its IP as the destination IP address, and 80 as the destination port. Servers usually do not care about the source IP or source port, although sometimes they will use them to establish the identity of the other side. When sending a response to such packets, the server will use its own IP as the source IP, and 80 as the source port.

When a client connects to a service, it may use any source port number on its side which is not already in use, but it must connect to the proper port on the server (e.g. 80 for web, 25 for email). TCP is a ***session oriented*** protocol with guaranteed delivery and transmission control features (such as detection and mitigation of network congestion, retries, packet reordering and reassembly, etc.). UDP is designed for ***connectionless*** streams of information, and does not guarantee delivery at all, or in any particular order.

The ICMP protocol is designed for debugging and maintenance on the Internet. Rather than port numbers, it has ***message types***, which are also numbers. Different message types are used to request a simple response from another computer (echo request), notify the sender of another packet of a possible routing loop (time exceeded), or inform the sender that a packet that could not be delivered due to firewall rules or other problems (destination unreachable).

By now you should have a solid understanding of how computers on the network are addressed, and how information flows on the network between them. Now let's take a brief look at the physical hardware that implements these network protocols.

Networking hardware

Before you can monitor the performance of your network, you first need to understand the capabilities of the network hardware. Can your router keep up with the bandwidth provided by your ISP? Is there a bottleneck at your internal network interconnections? You should know how your hardware will respond to demands for network resources so you understand the hard limits of what the network can provide. No amount of protocol optimisation or caching can help your Internet performance if your networking hardware simply cannot keep up with the demand.

Ethernet

Ethernet is the name of the most popular standard for connecting together computers on a ***Local Area Network*** (***LAN***). It is sometimes used to connect individual computers to the Internet, via a router, ADSL modem, or wireless device. However, if you connect a single computer to the Internet, you may not use Ethernet at all. The name comes from the physical concept of the ether, the medium which was once supposed to carry light waves through free space. The official standard is called IEEE 802.3.

The most common Ethernet standard is called 100baseT. This defines a data rate of 100 megabits per second, running over twisted pair wires, with modular RJ-45 connectors on the end. The network topology is a star, with switches or hubs at the centre of each star, and end nodes (devices and additional switches) at the edges.

MAC addresses

Every device connected to an Ethernet network has a unique MAC address, assigned by the manufacturer of the network card. Its function is like that of an IP address, since it serves as a unique identifier that enables devices to talk to each other. However, the scope of a MAC address is limited to a broadcast domain, which is defined as all the computers connected together by wires, hubs, switches, and bridges, but not crossing routers or Internet gateways. MAC addresses are never used directly on the Internet, and are not transmitted across routers.

Hubs

Ethernet ***hubs*** connect multiple twisted-pair Ethernet devices together. They work at the physical layer (the lowest or first layer). They repeat the signals received by each port out to all of the other ports. Hubs can therefore be considered to be simple repeaters. Due to this design, only one port can successfully transmit at a time. If two devices transmit at the same time, they corrupt each other's transmissions, and both must back off and retransmit their packets later. This is known as a ***collision***, and each host remains responsible for detecting collisions during transmission, and retransmitting its own packets when needed.

When problems such as excessive collisions are detected on a port, some hubs can disconnect (***partition***) that port for a while to limit its impact on the rest of the network. While a port is partitioned, devices attached to it cannot communicate with the rest of the network. Hub-based networks are generally more robust than coaxial Ethernet (also known as 10base2 or ThinNet), where misbehaving devices can disable the entire segment. But hubs are limited in their usefulness, since they can easily become points of congestion on busy networks.

Switches

A ***switch*** is a device which operates much like a hub, but provides a dedicated (or ***switched***) connection between ports. Rather than repeating all traffic on every port, the switch determines which ports are communicating directly and connects them together. Switches generally provide much better performance than hubs, especially on busy networks with many computers. They are not much more expensive than hubs, and are replacing them in many situations.

Switches work at the data link layer (the second layer), since they interpret and act upon the MAC address in the packets they receive. When a packet arrives at a port on a switch, it makes a note of the source MAC address, which it associates with that port. It stores this information in an internal ***MAC table***. The switch then looks up the destination MAC address in its MAC table, and transmits the packet on the matching port. If the destination MAC address is not found in the MAC table, the packet is then sent to all of the connected interfaces. If the destination port matches the incoming port, the packet is filtered and is not forwarded.

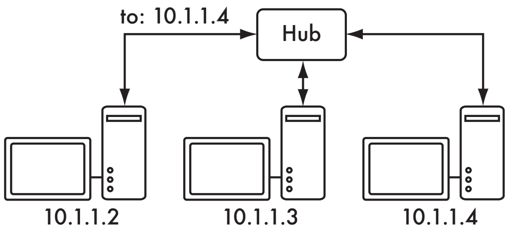
Hubs vs. Switches

Hubs are considered to be fairly unsophisticated devices, since they inefficiently rebroadcast all traffic on every port. This simplicity introduces both a performance penalty and a security issue. Overall performance is slower, since the available bandwidth must be shared between all ports. Since all traffic is seen by all ports, any host on the network can easily monitor all of the network traffic.

Switches create virtual connections between receiving and transmitting ports. This yields better performance because many virtual connections can be made simultaneously. More expensive switches can switch traffic by inspecting packets at higher levels (at the transport or application layer), allow the creation of VLANs, and implement other advanced features.

A hub should be used when repetition of traffic on all ports is desirable; for example, when you want to explicitly allow a monitoring machine to see all of the traffic on the network. Most switches provide ***monitor port*** functionality that enables repeating on an assigned port specifically for this purpose.

Hubs were once cheaper than switches. However, the price of switches have reduced dramatically over the years. Therefore, old network hubs should be replaced whenever possible with new switches.



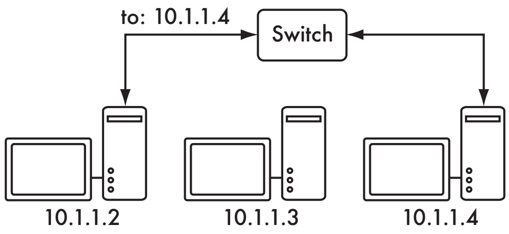


Figure 3.11: A hub simply repeats all traffic on every port, while a switch makes a temporary, dedicated connection between the ports that need to communicate.

Both hubs and switches may offer ***managed*** services. Some of these services include the ability to set the link speed (10baseT, 100baseT, 1000baseT, full or half duplex) per port, enable triggers to watch for network events (such as changes in MAC address or malformed packets), and usually include ***port counters*** for easy bandwidth accounting. A managed switch that provides upload and download byte counts for every physical port can greatly simplify network monitoring. These services are typically available via SNMP, or they may be accessed via telnet, ssh, a web interface, or a custom configuration tool.

Routers, firewalls, and NAT

While hubs and switches provide connectivity on a local network segment, a router's job is to forward packets between different network segments. A router typically has two or more physical network interfaces. It may include support for different types of network media, such as Ethernet, ATM, DSL, or dial-up. Routers can be dedicated hardware devices (such as Cisco or Juniper routers) or they can be made from a standard PC with multiple network cards and appropriate software.

Routers sit at the ***edge*** of two or more networks. By definition, they have one connection to each network, and as border machines they may take on other responsibilities as well as routing. Many routers have ***firewall*** capabilities that provide a mechanism to filter or redirect packets that do not fit security or access policy requirements. They may also provide Network Address Translation (NAT) services.

Routers vary widely in cost and capabilities. The lowest cost and least flexible are simple, dedicated hardware devices, often with NAT functionality, used to cheaply share an Internet connection between multiple computers. The next step up is a software router, which consists of an operating system running on a standard PC with multiple network interfaces. Standard operating systems such as Microsoft Windows, Linux, and BSD are all capable of routing, and are much more flexible than the low-cost hardware devices. However, they suffer from the same problems as conventional PCs, with high power consumption, a large number of complex and potentially unreliable parts, and more involved configuration.

The most expensive devices are high-end dedicated hardware routers, made by companies like Cisco and Juniper. They tend to have much better performance, more features, and higher reliability than software routers on PCs. It is also possible to purchase technical support and maintenance contracts for them.

Most modern routers offer mechanisms to monitor and record performance remotely, usually via the Simple Network Management Protocol (SNMP), although the least expensive devices often omit this feature.

Other equipment

Each physical network has an associated piece of terminal equipment. For example, VSAT connections consist of a satellite dish connected to a terminal that either plugs into a card inside a PC, or ends at a standard Ethernet connection. DSL lines use a ***DSL modem*** that bridges the telephone line to a local device, either an Ethernet network or a single computer via USB. ***Cable modems*** bridge the television cable to Ethernet, or to an internal PC card bus. Some kinds of telecom circuit (such as a T1 or T3) use a CSU/DSU to bridge the circuit to a serial port or Ethernet. Standard dialup lines use modems to connect a computer to the telephone, usually via a plug-in card or serial connection. And there are many different kinds of wireless networking equipment that connect to a variety of radios and antennas, but nearly always end at an Ethernet jack.

The functionality of these devices can vary significantly between manufacturers. Some provide mechanisms for monitoring performance, while others may not. Since your Internet connection ultimately comes from your ISP, you should follow their recommendations when choosing equipment that bridges their network to your Ethernet network.

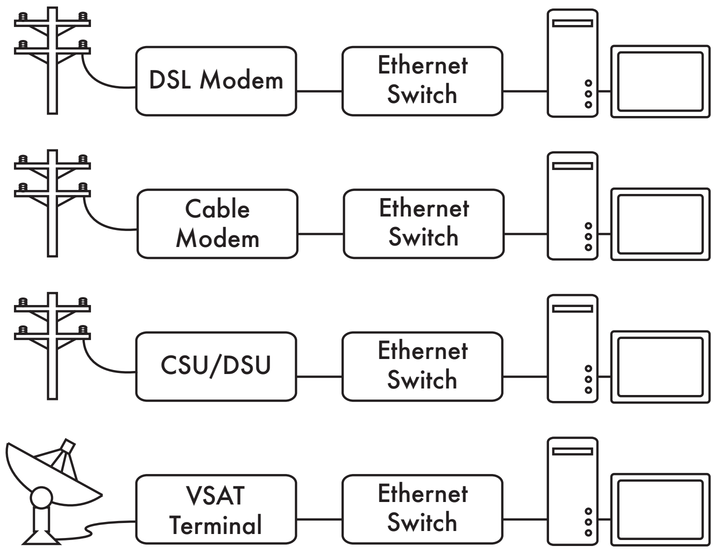


Figure 3.12: Many DSL modems, cable modems, CSU/DSUs, wireless access points, and VSAT terminals terminate at an Ethernet jack.

All components and devices should be kept in mind when analysing the network for potential bottlenecks.

Physical connectivity

Connections between computers are usually serial, that is, they transmit one bit at a time. The speed of the connection is measured in ***bits per second***, also known as ***bps***. There are eight bits in a byte, so the speed in bytes per second is eight times lower. For example, a 56,000 bps connection can transmit up to 7,000 bytes per second (Bps), which is a little less than seven kilobytes per second (7 kBps or kB/s).

The speed of the connection directly affects how fast it feels to the user. For example, the average web page is around 100 kilobytes (kB), and at 7 kBps, it takes just over 14 seconds to load. An email can be between 4 kilobytes and 2 megabytes, depending on the size of attachments, and take between half a second and five minutes to download. The information on a full CD-ROM, 680 megabytes, would take nearly 27 hours.

These examples ignore protocol overhead for the sake of simplicity. Each kind of connection necessarily adds a small amount of protocol overhead in order to maintain itself. This overhead reduces the available throughput and introduces some mandatory latency. Running TCP/IP on the connection further reduces the overall available throughput.

The Internet is a collection of many different networks, connected to each other in almost as many different ways. There is no single standard way to connect. In the early days, most networks were connected to each other by modems, but now a variety of digital and analogue technologies for interconnection exist. Each has different properties including price, range, speed, and quality, so choosing a connection method can be difficult. We will start with the oldest and slowest.

Telephone Modem

Technically, a ***modem*** is any device which converts digital signals to analogue and back, which includes almost every type of communications equipment. However, most people think of a modem as a box that connects a computer to a telephone line. This technology is old and slow, but cheap and reliable, and there are probably still more users connected to the Internet by telephone modems than any other method.

Modems come in two types: ***internal*** and ***external***. Internal modems are cards that fit inside a computer, and add a telephone port to the back panel of the computer. External modems are boxes which sit between the computer and the telephone line. Older external modems have RS-232 serial ports, usually with 25-pin D connectors, but modern external modems have USB connectors instead.

Modems send both commands and data over the same serial lines. Every modem has a slightly different set of commands, although they are all based on a standard, the ***Hayes AT command set***. Using a modem on a Windows or Macintosh computer requires driver software on the computer that knows which commands to send to it, while on Unix-based systems, you must specify the commands yourself. The supported command set is usually found in the modem's manual. Although standard AT commands will usually work, you will usually get better performance by using the right commands for the particular modem. This allows you to enable its high-speed modes and error-correction features. Some examples of advanced AT strings are provided in the **Performance Tuning** chapter, on page **177**.

There are many standards for modems that have been developed over their long life, usually to improve performance over previous models. The fastest modems today support the V.92 standard, which gives a maximum theoretical speed of 56,000 bits per second. Unfortunately, achieving this speed requires a digital telephone exchange and perfect laboratory conditions, so most connections are made at between 40,000 and 51,000 bps. On lower quality telephone lines, the maximum achievable speed may be 28,800 bps or less.

A modem connects to the public telephone network, and your computer connects to the Internet by using a modem to make a telephone call to another modem at the Internet provider. Therefore, you pay for a telephone call for the entire time that you are online. Since telephone calls are usually expensive, modems are usually used to ***dial on demand***, in other words to connect only when users actually need access to the Internet. Computers can be configured to dial regularly to collect mail even when unattended, for example in the middle of the night when calls are cheaper or more reliable.

ISDN

Integrated Services Digital Network (***ISDN***) is the direct digital equivalent of the analogue telephone line. The line uses pure digital signaling over two wires on the public side of the network, which terminates at a network terminator (NT1) box. This is then connected to an ISDN modem or digital telephone using four wires. The devices that connect computers to ISDN lines are not really modems, since the signaling is digital on both sides, but they look like modems physically and to the computer.

ISDN requires a digital telephone exchange, and is usually more expensive to install than an analogue telephone line, but it provides higher speed. The line has two "B" channels for data, and a "D" channel for control. The "B" channels can each carry a voice conversation, or up to 64,000 bps data, much faster and more reliably than analogue lines. It is usually possible to "bond" both B channels together to achieve 128,000 bps data, but this requires two telephone calls and is therefore twice as expensive, and not all Internet providers support it.

Leased Lines

A ***leased line*** is the equivalent of a physical wire between two points. Normally, the line is owned by a telephone company and is leased to a customer. It can be thought of as a telephone line that permanently connects two fixed points, usually no more than a few kilometres apart. It is usually priced similarly to a permanent telephone call, although prices have been forced down somewhat by competing technologies.

Leased lines vary widely in speed, from 64 Kbps to 10 Mbps. Standard speeds are T1 (1.5 Mbps) and E1 (2 Mbps). They are very simple technologies, are considered very reliable, and can be very fast, so many companies use them to connect to the Internet and between their offices. Leased lines are often used to host servers because of their speed and reliability, so Internet providers usually offer reasonably large numbers of static IP addresses with a leased line connection. On the other hand, they are very expensive, and many smaller companies have switched to competing technologies such as ADSL or wireless Internet.

A leased line normally comes onto your premises as a pair of wires, much like a telephone connection. If you own both ends, you can attach a modem, tin cups, or almost anything you like to the two ends. However, the effective range of leased lines is limited. If you connect to an Internet provider who is outside the range of what a leased line can reach, then then you may instead connect to the line provider's packet switching network, usually a ***frame relay*** network, which forwards the packets to your ISP. To do this, you will need to connect a device that understands the frame relay protocol to your end of the leased line. Frame relay is a Layer 2 (data link) network protocol, similar in purpose to Ethernet. Frame relay equipment is normally expensive, costing thousands of dollars per site. It may eventually be replaced by Ethernet.

Fibre Optic

Optical fibres are cables made from glass rather than copper. They carry information using visible light or infra-red rather than electrical potentials. They have very low signal loss and interference, even over very long distances, and are usually used for communications over distances greater than 5 km.

There are two standards for optical communication: ***Synchronous Digital Hierarchy*** (***SDH***) and Ethernet. SDH (and its precursor, ***SONET***, which is still in use in some parts of the world) were developed by telecommunications companies, while Ethernet was designed by computer companies. The two standards are completely incompatible. SDH is older, and much more widely used for long distance communications. Ethernet, particularly 10 Gigabit Ethernet over fibre (1000BASE-LX) offers lower cost and direct interconnection to Local Area Networks.

Optical fibre Internet access is also known as ***Fibre To The Home*** (***FTTH***) or ***Fibre To The Premises*** (***FTTP***). It is currently available in Japan, USA, Canada, some of Europe, Pakistan, New Zealand, and Australia. In other countries it may be available soon.

ADSL

ADSL stands for ***Asymmetric Digital Subscriber Line***, and is a popular implementation of ***DSL***, ***Digital Subscriber Line***. Digital in this case is misleading, as the line is not actually digital, but instead it is rated for much faster signaling than a conventional telephone line. This means it is usually quite close to the exchange, made from high quality copper, and any line filters have been removed. At the exchange is another compatible DSL device, which connects to the Internet using pure digital links. The maximum range of DSL is only a few kilometres, from the customer to the exchange.

DSL is a new technology that is designed to take advantage of recent developments in digital signal processing to offer reliable communications over relatively noisy lines. It takes data packets and breaks them into a number of frequency blocks or bands, including error correction data, and sends them down the line. Interference normally only affects one or two of these bands at a time, and therefore the DSL modem at the other end can reconstruct the signal if noise damages it, using the error correction data. This makes it much more robust over potentially noisy telephone lines, and gives it a greater range than leased lines.

Asymmetric DSL is so called because the download and upload speeds are different, unlike ***SDSL*** (***Symmetric DSL***) where they are the same. It is often the case that Internet users download more than they upload. For example, receiving email and browsing web pages are both asymmetric activities. On the other hand, running a server almost always requires more upload bandwidth, so ADSL lines are usually not suitable for servers. Telephone companies sell SDSL lines for similar prices as leased lines, but ADSL lines are usually much cheaper.

The reason that ADSL lines are cheap has very little to do with the technology. The main factors forcing the price down are competition from other Internet access technologies, such as cable modems, wireless and satellite. ADSL is often regarded as a consumer product, although small businesses are increasingly replacing leased lines with ADSL for Internet access, while SDSL and leased lines are still considered premium products.

ADSL connections vary widely in speed. The original standard, ADSL1, specified a maximum download speed of just over 8,000 kilobits per second (8 Mbps), while the latest standard, ADSL2+, allows up to 24 Mbps download. These speeds are usually not achieved in practice for three reasons:

- The maximum speed of ADSL depends on the quality of the line, and particularly on the distance between the user and the exchange. The maximum speed at a distance of 5 km is 1 Mbps, and at 6 km it is usually impossible to connect.
- Internet service providers usually place a limit on the speed of each line depending on the amount that the user pays. They will charge more for a 1 Mbps connection than a 128 kbps connection.
- ADSL Internet services are usually oversubscribed by a certain amount, for example 20:1. This is called the ***contention ratio***. It is a common practices based on the idea that most users do not use their connection all the time. In this case, the provider oversells their capacity by 20 or more times.

The issue of contention is almost a universal component of Internet access, and causes significant difficulties with effective bandwidth management since part of the bandwidth between you and the Internet is entirely outside of your control. This can of course be mitigated by signing a ***Service Level Agreement*** (***SLA***) with your ISP, but this comes at a high price, when it is available at all.

ADSL lines are wildly popular in most places where they are available, which includes most capital cities. In Europe, almost all broadband connections are ADSL, far more than cable modem or wireless.

Cable modems

A ***cable modem*** is similar to an ADSL modem, but slightly less complex. Cable modems connect to the cable television network, which is made from coaxial cable rather than pairs of telephone wires. Coaxial cable has much higher quality and lower interference than telephone wires, but is much more expensive.

Cable networks also differ in architecture compared to telephone networks. Cable networks were designed for one-way transmission (broadcasting) from a central point (known as the head end) to a large number of houses. Therefore, they are arranged in a tree structure, branching out from the head end. They are shared, rather than dedicated, so cable modems have to be careful not to talk over one another.

Cable modems transmit and receive on frequency bands, like ADSL, but usually only a single wide frequency band in each direction. This band corresponding to a TV channel on the cable TV network. Coaxial cables have low noise, so the signaling protocol is much simpler than ADSL. Cable modems usually conform to a standard known as ***DOCSIS*** (***Data Over Cable Service Interface Specification***), or the European equivalent ***EuroDOCSIS***.

Cable modems can achieve very high speeds. The fastest speed offered by the latest version, DOCSIS 3.0, is 160 Mbps download and 120 Mbps upload. However, speeds are usually limited by the cable company, and charged similarly to ADSL. Like ADSL, cable modems are usually oversubscribed by a significant margin, and excessive use by some users can have a significant negative impact on the bandwidth available to others. To make matters worse, users in the same neighborhood often share the same physical network segment. This means that many users share the same collision domain, impeding performance and making it possible for users to spy on each others' traffic.

This has recently been improved in some places through the use of hybrid fibre-coax networks. In these networks, the majority of the network backbone is fibre, and local neighborhoods connect via coax to the fibre backbone in numbers of 50-200. This significantly reduces the problems with oversubscription.

Wi-Fi

Wireless Fidelity, or ***Wi-Fi***, is a network technology that uses radio waves to link multiple computers. It grew out of proprietary technologies developed by Lucent, and was standardised by the IEEE under the inspiring and memorable name of 802.11.

The most common Wi-Fi standard is ***802.11b***, which specifies a maximum data rate of 11 Mbps on each of 13 channels around 2.4 GHz. Due to protocol overhead, the maximum available throughput on 802.11b is approximately 5.5 Mbps. The 2.4 GHz frequency band is reserved for Industrial, Scientific, and Medical applications by the ITU, and is known as the ***ISM band***. The ITU permits its use at low power without any license, and therefore Wi-Fi equipment does not require a license to own or operate in most countries. However, the power restrictions limit its range to 300 metres under normal conditions, and around 20 kilometres under ideal conditions with specially adapted equipment. Also, there is no protection from interference in this band, so common household appliances (such as cordless phones and microwave ovens) can generate significant amounts of interference.

Wi-Fi equipment has become very popular in Europe and America now that it is extremely cheap, and almost every modern laptop has a Wi-Fi interface built in. Most companies run their own Wi-Fi networks to allow laptop users to access the corporate network and the Internet away from their desks without trailing cables. Many businesses, for example coffee shops, sell or give away Internet access to their customers via Wi-Fi. Many people install a Wi-Fi network at home to avoid the need to purchase and install cables around the house.

Newer 802.11 standards offer higher speeds. The most popular of these new standards is 802.11g, which provides a radio rate of up to 54 Mbps (22 Mbps throughput) at 2.4 GHz, and is backwards compatible with 802.11b. There is also 802.11a, which operates around 5 GHz. However, 802.11a suffers from range problems and limitations on outdoor use, and is not nearly as ubiquitous and well supported as 802.11b/g.

Some Internet providers offer access using Wi-Fi, which is a very cheap option since the hardware costs about a hundred times less than that required for fixed wireless or microwave networks, and no licenses are required. However, the hardware can be less reliable, is much more susceptible to interference, and can be much less secure than other types of connections.

For a more in-depth look at how Wi-Fi can be adapted to provide Internet connectivity across very long distances, see Wireless Networking in the Developing World, *http://wndw.net/.*

Satellite

There are several standards for satellite Internet access, including ***Digital Video Broadcast*** (***DVB-S***), the ***Broadband Global Access Network*** (***BGAN***) and ***Very Small Aperture Terminal*** (***VSAT***).

Digital Video Broadcasting, or DVB, is a standard for digital communications via satellite. Originally designed for broadcasting television, it has been extended to allow two-way Internet communications. Several satellite providers offer Internet access via DVB, at relatively low cost. Access requires a satellite dish and a DVB-Data tuner card or external modem. Installation requires very precise positioning of the local satellite dish, and therefore must be done by a professional with the necessary tools.

DVB Internet access is available anywhere in the world, even on ships at sea. It is often used in rural areas of Europe and America, where terrestrial broadband is not yet available. It offers high bandwidth, with download speeds up to 16 Mbps, and upload up to 1 Mbps.

BGAN is designed for portable use, and the terminals are very small. It offers speeds up to 492 kbps in both directions. The terminal is not too expensive, but bandwidth prices are usually very high.

VSAT requires a fixed installation, and the dishes are larger than those used for DVB and satellite TV. It offers speeds up to 2 Mbps in each direction.

All satellites used for Internet access are geostationary, and due to their distance from the Earth, Internet access suffers from high round trip times, usually between 500 ms and one second. Depending on the location of the satellite that you use, you may find that your data is routed via the USA, Europe or Asia. The satellite may be very close to the horizon from your location, which results in lower signal strength and interference from objects on the ground, and signal quality may be affected by weather such as rain.

We will review different protocols and variables that affect satellite performance in greater detail in chapter six, **Performance Tuning**.

Fixed Microwave

Fixed microwave networks use radio frequencies between 2 and 31 GHz to connect base stations up to 20 kilometres apart. They are usually point-to-point links, which extend an existing network or connect two networks together. They use high power to achieve such high range and bandwidth, and require licenses to operate. Microwave networks are much cheaper to install than wires, especially because they do not require land ownership or access rights for the territory they cross. They are also more physically secure, since copper and fibre optic cables can easily be damaged or stolen from the ground.

In many countries, one or more telecommunications companies compete with the state telephone operator by offering wireless "leased line" equivalents using fixed microwave networks. They offer almost the same speed and reliability as wired leased lines, at much lower cost, and are much more easily available in remote areas.

Mobile telephone operators often use fixed microwave networks to connect their rural base stations, and land telephone operators often use them to connect remote towns, villages, and isolated businesses in areas where no fixed telephone lines are available. The antennas often look like wide, thick white discs, flat on the front side and slightly curved or angled on the back side, and can be seen on almost all mobile telephone masts.

Fixed microwave equipment is usually proprietary and incompatible with equipment made by different manufacturers. It is also usually quite expensive, costing several thousand dollars for the equipment at each end.

WiMax

The WiMax standard, formally known as IEEE 802.16, is an attempt to standardise fixed microwave networks and achieve compatibility between equipment made by different manufacturers. The standard is very new, and little equipment that conforms to it is currently available. It is expected to bring down the cost of fixed microwave links, and wireless broadband access, by increasing competition.

It is often claimed that WiMAX will provide access to the Internet at up to 70 Mbps, range up to 70 miles and to a user travelling up to 70 mph, but these are not all true simultaneously. We expect that WiMAX will offer Internet access to remote and rural areas at significantly longer ranges and higher speeds than current options.

WiMAX controls access to the network by allocating time slots to individual users. This can allow more efficient use of radio spectrum than Wi-Fi, and guarantee minimum bandwidth availability, which Wi-Fi cannot do. The standard is specified in the frequency range from 2 to 66 GHz, but it seems likely that most deployments will use frequencies between 2.3 and 5 GHz.

Comparison Table

The table on the following page lists the general features of each type of connection for easy comparison. Data rates given are the rate stated by the manufacturer and reflects the theoretical maximum before accounting for protocol overhead.

	Round Trip Time (ms)	Cost (USD)	Quality	Range	Max Up	Max Down	Availability
Modem	100 to 160	cents / minute	High	Global	33.6 kbps	56 kbps	All but the most rural areas
ISDN	20 to 40	cents / minute	High	Global	128 kbps	128 kbps	All but the most rural areas
Leased Line	5 to 20	1000 / month	High	2 km	10 Mbps	10 Mbps	All but the most rural areas
Fibre Optic	Less than 1	50 to 1000 / month	Very High	20 km	1 Gbps	1 Gbps	Some cities
ADSL	10 to 20	20 to 200 / month	High	5 km	1 Mbps	24 Mbps	Most cities
Cable	5 to 80	20 to 200 / month	Medium to High	60+ km	120 Mbps	160 Mbps	Most cities
Wi-Fi	1 to 40	20 to 200 / month	Medium	up to 20 km	54 Mbps	54 Mbps	Some cities
Satellite	At least 500	50 to 450 / month	Medium	Global	1 Mbps	16 Mbps	Global
Fixed Microwave & WiMax	1 to 30	150 to 3000 / month	High	20 km or more	70 Mbps	70 Mbps	Some cities

Virtual connectivity

So far, we have talked about different types of physical network connections. But those physical connections can be broken into even more logical networks, providing multiple isolated virtual networks on a single physical network. The benefits of doing this include being able to restrict which hosts can communicate with each other, separate legacy networks which require control of the broadcast domain from each other, and protect machines from attack over the public Internet.

Virtual LAN (VLAN)

A ***Virtual LAN*** (***VLAN***) is a logical network that can coexist with, and be isolated from, other VLANs on the same physical medium. VLANs are normally implemented by network switching hardware, and so it makes no difference to a computer whether it is connected to a LAN or a VLAN. VLANs can be used to partition a single network switch into a number of isolated domains, effectively simulating a number of independent unconnected switches. This may reduce cost, space, and administration overheads in wiring cabinets. Switches can also be reconfigured remotely, allowing you to logically regroup computers or network ports within an organisation, without the need to physically rewire them.

In terms of bandwidth management, VLANs allow you to group computers by function, and make access policy and bandwidth shaping rules by group. Each VLAN becomes an isolated broadcast domain for the computers it contains. For example, you may create separate VLANs for public terminals, research machines, and administrative computers. Each may be allocated a different amount of bandwidth, or may have different rules in place to permit (or deny) access to particular sites and services. With VLANs in place, these computers do not need to be physically connected to the same switch. Also, computers connected to the same switch can be differentiated by the gateway according to which VLAN they are placed in.

VLAN membership can be decided in three different ways. Support for these may depend on the network switch.

- **Port-based**. A switch port is manually configured to be a member of a VLAN, so any device plugged into that physical port is automatically part of it. Some VLAN switches support ***trunking***, which sends and receives packets tagged with VLAN numbers, and also allows VLANs to extend between switches.
- **Protocol-based**. Layer three protocol information within the frame is used to determine VLAN membership. The major disadvantage of this method is that it violates the independence of the layers. For example, the switch may fail to enforce the VLAN when upgrading from IPv4 to IPv6.
- **MAC-based**. VLAN membership is based on the MAC address of the workstation. The switch has a table listing the MAC address of each machine along with the VLAN to which it belongs. This can allow VLAN policies to work, even if the same machine is plugged into different switch ports (e.g., a laptop user moving around in a building).

The format for tagging packets with the VLAN number, which allows switches and firewalls to share VLAN information, has been standardised by the IEEE as 802.1q.

Virtual Private Network (VPN)

An organisation's network normally offers some private services to its members, such as file servers and web servers. Those services are normally not accessible over the Internet for security reasons. When members of the organisation are operating outside of this network, they will not be able to access these resources. ***Virtual Private Networks*** (***VPNs***) provide a mechanism for securely connecting to such resources, and even linking entire networks together, over an untrusted public network such as the Internet. They are often used to connect remote users to an organisation's network when travelling or working from home.

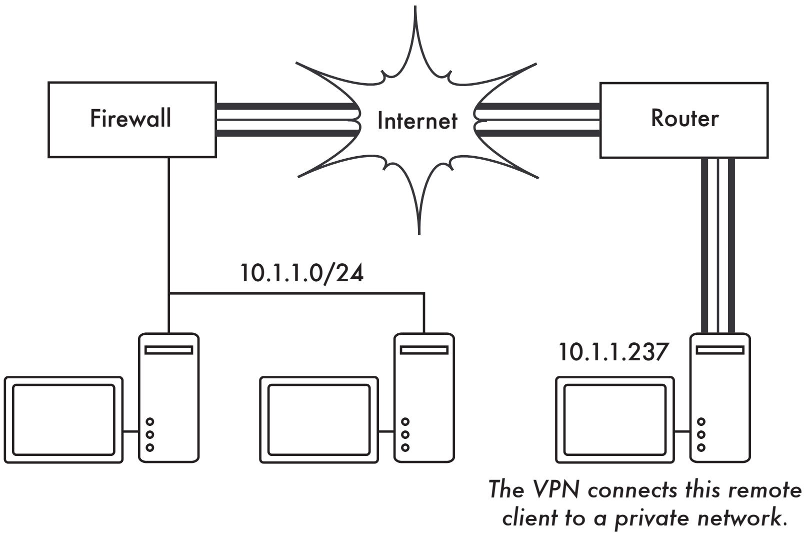


Figure 3.13: VPNs protect sensitive data when crossing untrusted networks. Eavesdroppers cannot penetrate the encrypted VPN encapsulation, represented by the heavy lines.

VPNs may create permanent, on-demand, or temporary connections between two or more networks. They can provide bridging between networks (at the Data Link Layer) or a routed connection (at the Network Layer). Data on the internal network is encrypted and transported within a ***tunnel*** to the remote side, where it is then decrypted and passed on. The encrypted tunnel protects communications from potential eavesdroppers. The establishment of a VPN is normally protected by strong authentication, such as private keys, certificates, or hardware tokens. This provides a strong guarantee to the organisation that its private resources are not being accessed by an unauthorised user.

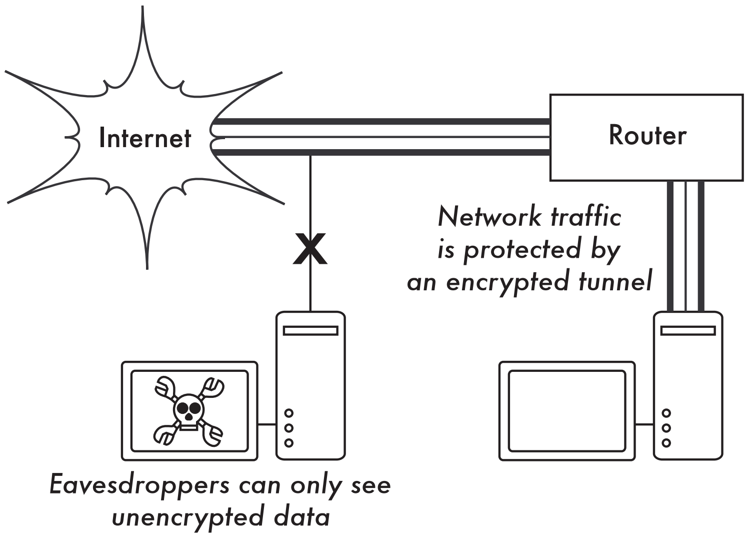

Internet
Router
X
Network traffic is protected by an encrypted tunnel
Eavesdroppers can only see unencrypted data

Figure 3.14: Encrypted tunnels can protect data that is not otherwise encrypted by the application itself.

While VPNs create a very flexible and secure connection between remote networks, the tunnel adds overhead to every packet sent over the public network. This makes using a VPN less efficient than accessing a service directly. Attempting to run VPNs over an already crowded Internet connection can make things even worse. One particular problem is that firewalls and bandwidth management software can only see encrypted VPN packets, and not the traffic within, which can make securing, policing, and controlling VPNs much more difficult. Unfortunately, most VPNs are nearly useless over high latency VSAT connections without extensive tweaking.

While VPNs will not improve bandwidth utilisation, they are useful when you need to provide secure access to private resources across an untrusted network, and your bandwidth is not too heavily used.

Summary

Every computer on the Internet has at least one unique IP address. This address is used by other computers to contact it, and send information to it. IP addresses are grouped into subnets of different sizes. Subnets are normally physically connected networks such as Local Area Networks (LANs), and they are connected together by routers and firewalls. There are many different technologies for connecting your network to the Internet, each with its own advantages and disadvantages. Bandwidth management is the process of controlling the information that flows between your network and the Internet.

For more information about the basics of networking, you can find some useful guides online, including Rusty Russell's Linux Networking Concepts: *http://www.netfilter.org/documentation/HOWTO/networking-concepts-HOWTO.html*

By now you should have a clear idea of how information flows between your network and the Internet. Understanding how your network and Internet connection are being used is the first step towards optimising them, improving performance and reducing your operating costs.

What is network monitoring?

Network monitoring is the use of logging and analysis tools to accurately determine traffic flows, utilisation, and other performance indicators on a network. Good monitoring tools give you both hard numbers and graphical aggregate representations of the state of the network. This helps you to visualise precisely what is happening, so you know where adjustments may be needed. These tools can help you answer critical questions, such as:

- What are the most popular services used on the network?
- Who are the heaviest network users?
- At what time of the day is the network most utilised?
- What sites do your users frequent?
- Is the amount of inbound or outbound traffic close to our available network capacity?
- Are there indications of an unusual network situation that is consuming bandwidth or causing other problems?
- Is our Internet Service Provider (ISP) providing the level of service that we are paying for? This should be answered in terms of available bandwidth, packet loss, latency, and overall availability.

And perhaps the most important question of all:

- Do the observed traffic patterns fit our expectations?

If you cannot answer these questions, then you are likely wasting bandwidth. With good network monitoring tools in place, it is easier to troubleshoot problems, identify your biggest bandwidth users, and plan for future capacity needs.

Let's look at how a typical system administrator can make good use of network monitoring tools.

An effective network monitoring example

For the purposes of example, let's assume that we are in charge of a network that has been running for three months. It consists of 50 computers and three servers - email, web, and proxy servers. While initially things are going well, users begin to complain of slow network speeds and an increase in spam emails. As time goes on, computer performance slows to a crawl (even when not using the network), causing considerable frustration in your users.

With frequent complaints and very low computer usage, the Board is questioning the need for so much network hardware. The Board also wants evidence that the bandwidth they are paying for is actually being used. As the network administrator, you are on the receiving end of these complaints. How can you diagnose the sudden drop in network and computer performance and also justify the network hardware and bandwidth costs?

Monitoring the LAN (local traffic)

To get an idea of exactly what is causing the slow down, you should begin by looking at traffic on the local LAN. There are several advantages to monitoring local traffic:

- Troubleshooting is greatly simplified.
- Viruses can be detected and eliminated.
- Malicious users can be detected and dealt with.
- Network hardware and resources can be justified with real statistics.

Assume that all of the switches support the ***Simple Network Management Protocol*** (***SNMP***). SNMP is an application-layer protocol designed to facilitate the exchange of management information between network devices. By assigning an IP address to each switch, you are able to monitor all the interfaces on that switch, and can therefore monitor the entire network from a single point. This is much easier than enabling SNMP on all computers in a network.

By using a free tool such as MRTG (page **83**), you can monitor each port on the switch and present data graphically, as an aggregate average over time. The graphs are accessible from the web, so you are able to view the graphs from any machine at anytime.

With MRTG monitoring in place, it becomes obvious that the internal LAN is swamped with far more traffic than the Internet connection can support, even when the lab is unoccupied. This is a pretty clear indication that some of the computers are infested with a network virus. After installing good anti-virus and anti-spyware software on all of the machines, the internal LAN traffic settles down to expected levels. The machines run much more quickly, spam emails are reduced, and the users' morale quickly improves.

Monitoring the WAN (external traffic)

In addition to watching the traffic on the internal LAN, you need to demonstrate that the bandwidth the organisation is paying for is actually what they are getting from their ISP. You can achieve this by monitoring ***external traffic***.

External traffic is generally classified as anything sent over a ***Wide Area Network*** (***WAN***). Anything received from (or sent to) a network other than your internal LAN also qualifies as external traffic. The advantages of monitoring external traffic include:

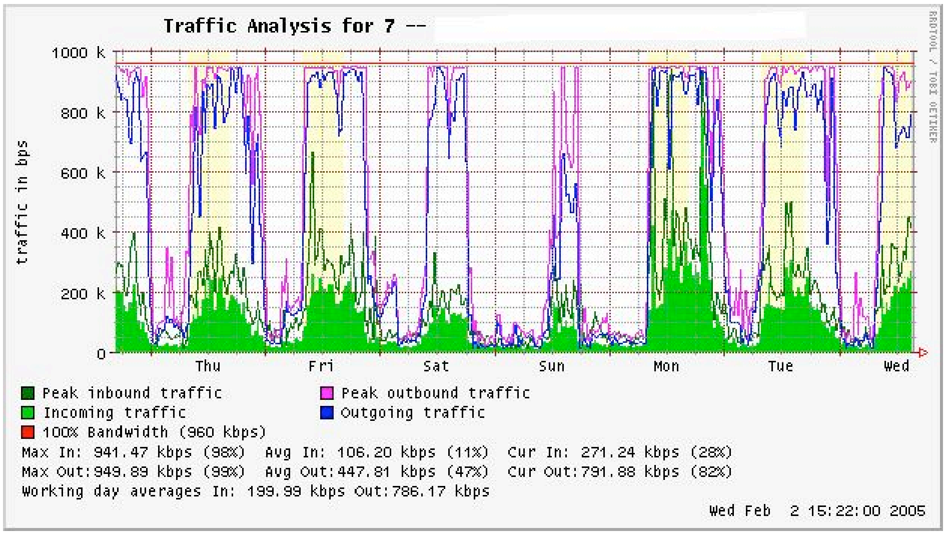
- Internet bandwidth costs are justified by showing actual usage, and whether that usage agrees with your ISP's bandwidth charges.
- Future capacity needs are estimated by watching usage trends and predicting likely growth patterns.
- Intruders from the Internet are detected and filtered before they can cause problems.

Monitoring this traffic is easily done with the use of MRTG on an SNMP enabled device, such as a router. If your router does not support SNMP, then you can add a switch between your router and your ISP connection, and monitor the port traffic just as you would with an internal LAN.

Detecting Network Outages

With monitoring tools in place, you now have an accurate measurement of how much bandwidth the organisation is using. This measurement should agree with your ISP's bandwidth charges. It can also indicate the actual throughput of your connection if you are using close to your available capacity at peak times. A "flat top" graph is a fairly clear indication that you are operating at full capacity. Figure 3.15 shows flat tops in peak outbound traffic in the middle of every day except Sunday.

It is clear that your current Internet connection is overutilised at peak times, causing network lag. After presenting this information to the Board, you can make a plan for further optimising your existing connection (by upgrading your proxy server and using other techniques in this book) and estimate how soon you will need to upgrade your connection to keep up with the demand. This is also an excellent time to review your operational policy with the Board, and discuss ways to bring actual usage in line with that policy.



```
Max In: 941.47 kbps (98%)  Avg In: 106.20 kbps (11%)  Cur In: 271.24 kbps (28%)
Max Out:949.89 kbps (99%)  Avg Out:447.81 kbps (47%)  Cur Out:791.88 kbps (82%)
Working day averages In: 199.99 kbps Out:786.17 kbps
                                                    Wed Feb  2 15:22:00 2005
```

Figure 3.15: A graph with a "flat top" is one indication of overutilisation.

Later in the week, you receive an emergency phone call in the evening. Apparently, no one in the lab one can browse the web or send email. You rush to the lab and hastily reboot the proxy server, with no results. Browsing and email are still broken. You then reboot the router, but there is still no success. You continue eliminating the possible fault areas one by one until you realise that the network switch is off - a loose power cable is to blame. After applying power, the network comes to life again.

How can you troubleshoot such an outage without such time consuming trial and error? Is it possible to be notified of outages as they occur, rather than waiting for a user to complain? One way to do this is to use a program such as ***Nagios*** that continually polls network devices and notifies you of outages. Nagios will report on the availability of various machines and services, and will alert you to machines that have gone down. In addition to displaying the network status graphically on a web page, it will send notifications via SMS or email, alerting you immediately when problems arise.

With good monitoring tools in place, you are able to justify the cost of equipment and bandwidth by demonstrating how it is being used by the organisation. You are notified automatically when problems arise, and you have historical statistics of how the network devices are performing. You can check the current performance against this history to find unusual behaviour, and head off problems before they become critical. When problems do come up, it is simple to determine the source and nature of the problem. Your job is easier, the Board is satisfied, and your users are much happier.

Monitoring your network

Managing a network without monitoring is similar to driving a vehicle without a speedometer or a fuel gauge. How do you know how fast you are going? Is the car consuming fuel as efficiently as promised by the dealers? If you do an engine overhaul several months later, is the car any faster or more efficient than it was before?

Similarly, how can you pay for an electricity or water bill without seeing your monthly usage from a meter? You must have an account of your network bandwidth utilisation in order to justify the cost of services and hardware purchases, and to account for usage trends.

There are several benefits to implementing a good monitoring system for your network:

1. **Network budget and resources are justified.** Good monitoring tools can demonstrate without a doubt that the network infrastructure (bandwidth, hardware, and software) is suitable and able to handle the requirements of network users.
2. **Network intruders are detected and filtered.** By watching your network traffic, you can detect attackers and prevent access to critical internal servers and services.
3. **Network viruses are easily detected.** You can be alerted to the presence of network viruses, and take appropriate action before they consume Internet bandwidth and destabilise your network
4. **Troubleshooting of network problems is greatly simplified.** Rather than attempting "trial and error" to debug network problems, you can be instantly notified of specific problems. Some kinds of problems can even be repaired automatically.
5. **Network performance can be highly optimised.** Without effective monitoring, it is impossible to fine tune your devices and protocols to achieve the best possible performance.
6. **Capacity planning is much easier.** With solid historical performance records, you do not have to "guess" how much bandwidth you will need as your network grows.
7. **Proper network usage can be enforced.** When bandwidth is a scarce resource, the only way to be fair to all users is to ensure that the network is being used for its intended purpose.

Fortunately, network monitoring does not need to be an expensive undertaking. There are many freely available open source tools that will show you exactly what is happening on your network in considerable detail. This chapter will help you identify many invaluable tools and how best to use them.

The dedicated monitoring server

While monitoring services can be added to an existing network server, it is often desirable to dedicate one machine (or more, if necessary) to network monitoring. Some applications (such as ***ntop***) require considerable resources to run, particularly on a busy network. But most logging and monitoring programs have modest RAM and storage requirements, typically with little CPU power required. Since open source operating systems (such as Linux or BSD) make very efficient use of hardware resources, this makes it possible to build a very capable monitoring server from recycled PC parts. There is usually no need to purchase a brand new server to relegate to monitoring duties.

The exception to this rule is in very large installations. If your network includes more than a few hundred nodes, or if you consume more than 50 Mbps of Internet bandwidth, you will likely need to split up monitoring duties between a few dedicated machines. This depends largely on exactly what you want to monitor. If you are attempting to account for all services accessed per MAC address, this will consume considerably more resources than simply measuring network flows on a switch port. But for the majority of installations, a single dedicated monitoring machine is usually enough.

While consolidating monitoring services to a single machine will streamline administration and upgrades, it can also ensure better ongoing monitoring. For example, if you install monitoring services on a web server, and that web server develops problems, then your network may not be monitored until the problem is resolved.

To a network administrator, the data collected about network performance is nearly as important as the network itself. Your monitoring should be robust and protected from service outages as well as possible. Without network statistics, you are effectively blind to problems with the network.

Where does the server fit in my network?

If you are only interested in collecting network flow statistics from a router, you can do this from just about anywhere on the LAN. This provides simple feedback about utilisation, but cannot give you comprehensive details about usage patterns. Figure 3.16 shows a typical MRTG graph generated from the Internet router. While the inbound and outbound utilisation are clear, there is no detail about which computers, users, or protocols are using bandwidth.

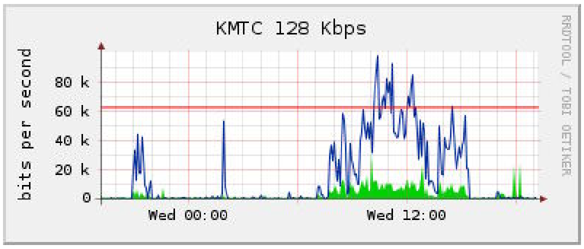


Figure 3.16: Polling the edge router can show you the overall network utilisation, but you cannot break the data down further into machines, services, and users.

For more detail, the dedicated monitoring server must have access to everything that needs to be watched. Typically, this means it must have access to the entire network. To monitor a WAN connection, such as the Internet link to your ISP, the monitoring server must be able to see the traffic passing through the edge router. To monitor a LAN, the monitoring server is typically connected to a ***monitor port*** on the switch. If multiple switches are used in an installation, the monitoring server may need a connection to all of them. That connection can either be a physical cable, or if your network switches support it, a VLAN specifically configured for monitoring traffic.

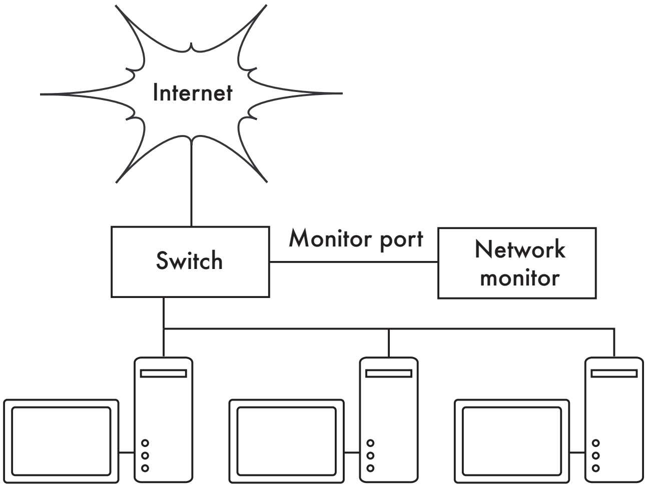


Figure 3.17: Use the monitor port on your switch to observe traffic crossing all of the network ports.

If monitor port functionality is not available on your switch, the monitoring server may be installed between your internal LAN and the Internet. While this will work, it introduces a single point of failure for the network, as the network will fail if the monitoring server develops a problem. It is also a potential performance bottleneck, if the server cannot keep up with the demands of the network.

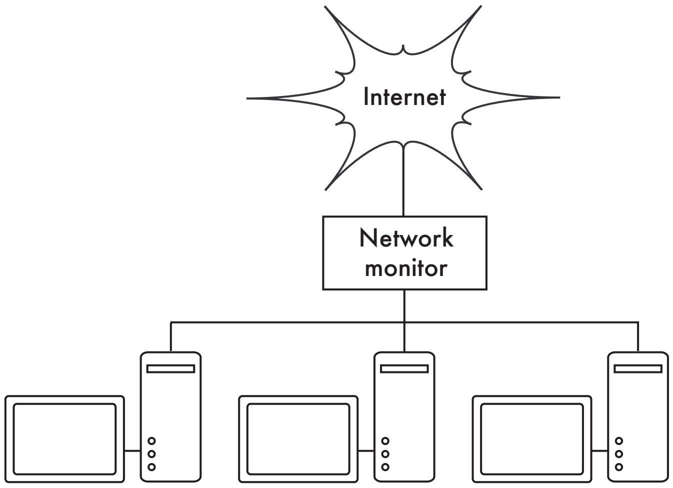


Figure 3.18: By inserting a network monitor between the LAN and your Internet connection, you can observe all network traffic.

A better solution is to use a simple network hub (not a switch) which connects the monitoring machine to the internal LAN, external router, and the monitoring machine. While this does still introduce an additional point of failure to the network (since the entire network will be unreachable if the hub dies), hubs are generally considered to be much more reliable than routers. They are also very easily replaced should they fail.

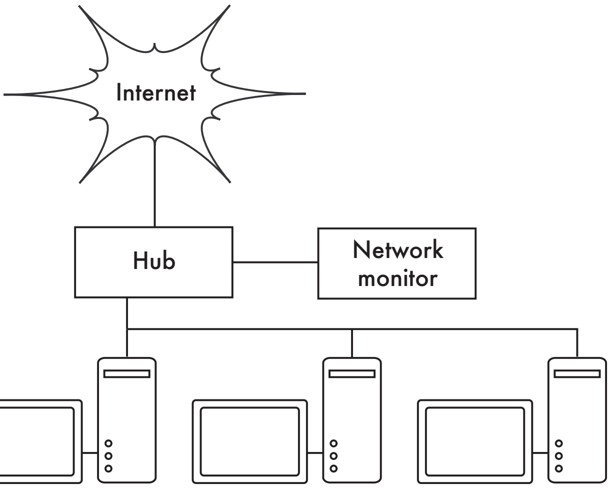


Figure 3.19: If your switch does not provide monitor port functionality, you can insert a network hub between your Internet router and the LAN, and connect the monitoring server to the hub.

Once your monitoring server is in place, you are ready to start collecting data.

What to monitor

It is possible to plot just about any network event and watch its value on a graph over time. Since every network is slightly different, you will have to decide what information is important in order to gauge the performance of your network.

Here are some important indicators that many network administrators will typically track.

Switch statistics

- Bandwidth usage per switch port
- Bandwidth usage broken down by protocol
- Bandwidth usage broken down by MAC address
- Broadcasts as a percentage of total packets
- Packet loss and error rate
- Wireless signal, noise, and data rate

Internet statistics

- Internet bandwidth use by host and protocol
- Proxy server cache hits
- Top 100 sites accessed
- DNS requests
- Number of inbound emails / spam emails / email bounces
- Outbound email queue size
- Availability of critical services (web servers, email servers, etc.).
- Ping times and packet loss rates to your ISP
- Status of backups

System health statistics

- Memory usage
- Swap file usage
- Process count / zombie processes
- System load

- Uninterruptible Power Supply (UPS) voltage and load
- Temperature, fan speed, and system voltages
- Disk SMART status
- RAID array status

You should use this list as a suggestion of where to begin. As your network matures, you will likely find new key indicators of network performance, and you should of course track those as well. There are many freely available tools that will show you as much detail as you like about what is happening on your network. You should consider monitoring the availability of any resource where unavailability would adversely affect your network users.

For example, your users may dial into modems on your site to gain remote access to your network. If all the modems are used, or if any are faulty, then users will be denied access and will probably complain. You can predict and avoid such problems by monitoring the number of available modems, and provisioning extra capacity before you run out.

Don't forget to monitor the monitoring machine itself, for example its CPU usage and disk space, in order to receive advance warning if it becomes overloaded or faulty. A monitoring machine that is low on resources can affect your ability to monitor the network effectively.

How to select tools to monitor the network

In order to properly and effectively monitor a network, you must first select a monitoring tool. When evaluating tools, you should consider which of the following properties are most important to you. Good monitoring tools should be:

- **Appropriate.** Just as you do not measure the contents of a gas tank with a thermometer, you should use the proper monitoring tool for the job. Some tools (such as ***tcpdump***) can give you enormous amounts of detail, but very little idea of trends over time. Others (such as ***MRTG***) give you great historical detail of network flows, but no details about a particular protocol or user. Your monitoring solution will include several tools, each performing a particular monitoring task.
- **Affordable.** There are many monitoring programs that are totally free for download on the Internet and require no payments or license fees. If your project has a limited budget, these can be ideal. Even if you have considerable budget to spend on monitoring tools, be sure to compare the functionality of proprietary solutions with the free or open source equivalent.
- **Lightweight.** This means the tool must not be processor or memory exhaustive, especially if you are unable to dedicate a machine to monitoring. It must

also be able to perform for long periods without taking up much hard disk space. Some programs may require huge amounts of space to function, while others may need little or no storage at all. Be sure that the tool you choose fits the hardware on which it is used.

- **Flexible**. Your tools should adapt to your networking environment, not the other way around. Since every network is different, your monitoring tools should be able to be configured to accurately display the data you need to track.
- **Graphical.** The best programs plot graphs or have colorful displays to allow for easy and quick understanding of network activities. For example, red will indicate an alert or error while green will indicate no problems. This turns data into information as opposed to a meaningless string of numbers and percentages. Most modern programs will do both.
- **Supported.** Ensure that the program is well supported by its authors (or the community at large) with updates and patches. Such programs are more secure and more reliable in the long run, since developers are constantly reviewing functionality and addressing problems.
- **Data retentive.** The program should be able to log and retain data over long periods (e.g. two or three years). This will help you discover trends in a network environment by comparing values in year A to values in year B.
- **User friendly and feature rich.** The best programs are easy to use and present data in an easy to understand way. The program should also have features such as web accessibility and even web administration, so that people with modest computer skills can help monitor the network.

While not all tools will meet every one of these criteria, it is important to keep these points in mind when deciding which tools you will use to monitor your network.

Types of monitoring tools

We will now look at several different classes of monitoring tools. ***Spot check*** tools are designed for troubleshooting and normally run interactively for short periods of time. A program such as **`ping`** may be considered an active spot check tool, since it generates traffic by polling a particular machine. Passive spot check tools include ***protocol analysers***, which inspect every packet on the network and provide complete detail about any network conversation (including source and destination addresses, protocol information, and even application data). ***Trending*** tools perform unattended monitoring over long periods, and typically plot the results on a graph. ***Realtime monitoring*** tools perform similar monitoring, but notify administrators immediately if they detect a problem. ***Log analysis*** tools summarise the logs of various other programs, and may notify an administrator when problems arise. ***Intrusion detection*** tools watch for undesirable or unexpected network traffic, and take appropriate action (typically denying access and/or notifying a network administrator). Finally, ***benchmarking*** tools estimate the maximum performance of a service or network connection.

The class of tool to use depends on the question you need to answer.

- **How do I troubleshoot an immediate problem?** If your logs or graphs don't indicate the source of the problem (page **83**), use a spot check tool (page **74**).
- **How do I estimate the performance of my connection?** Use a benchmarking tool (page **89**).
- **How do I recognise a virus or denial of service attack?** Look at bandwidth consumption trends (page **83**), watch your email and web server logs (page **80**), or use an intrusion detection tool or spot check tool (page **74**). Also see the discussion on pages **170** and **174**.
- **How can I be automatically notified when outages occur?** Use a realtime monitoring tool (page **87**).
- **How do I ensure that bandwidth is used for suitable services/purposes?** Use an intrusion detection tool (page **87**) or protocol analyser (page **78**.) It is also useful to install a trending tool to monitor protocols used broken down by bandwidth (or even IP or MAC address) over time (page **83**.)
- **When should I upgrade my bandwidth?** Use a trending tool (page **83**) in conjunction with a protocol analyser (page **78**). When you are sure that your bandwidth is being used appropriately, a trending tool will show you how much you have consumed in the past, and you can predict how much you will need in the future.

Walking around the lab

Do not discount the effectiveness of simply walking into a lab environment. If you wish to have an understanding of the social habits of users in an educational facility, visit the lab during the busiest times of the day. While walking around different areas of the lab, I have been surprised at what I have found. In areas where there is little physical policing, students often browse personal sites, play games, listen to music, and perform other recreational activities. In the areas where more rigid physical policing has been performed through notices and by lab consultants, the atmosphere tends to much more work oriented.

Being available to (and noticed by) your users can often improve network performance significantly. This is nearly always easier to implement than any technical solution.

Spot check tools

There may be times when you suspect someone is misappropriating network resources. Perhaps someone has found a way around the proxy, and is downloading music or movies. ***Spot check tools*** can help you locate the source of the problem. Spot check tools give you a quick view of what is happening on your connection in real time. Some tools (such as ***ntop***) can run continually, while others (such as ***wireshark***) are usually only run when needed.

ping

Probably the most commonly used spot check tool is the ubiquitous ***ping*** command. Just about every operating system (including Windows, Mac OS X, and of course Linux and BSD) includes a version of this utility. It uses ICMP packets to attempt to contact a specified host, and tells you how long it takes to get a response.

Knowing what to ping is just as important as knowing how to ping. If you find that you cannot connect to a particular service in your web browser (say, *http://yahoo.com/*), you could try to ping it:

```
$ ping yahoo.com
PING yahoo.com (66.94.234.13): 56 data bytes
64 bytes from 66.94.234.13: icmp_seq=0 ttl=57 time=29.375 ms
64 bytes from 66.94.234.13: icmp_seq=1 ttl=56 time=35.467 ms
64 bytes from 66.94.234.13: icmp_seq=2 ttl=56 time=34.158 ms
^C
--- yahoo.com ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max/stddev = 29.375/33.000/35.467/2.618 ms
```

Hit control-C when you are finished collecting data. If packets take a long time to come back, there may be network congestion. If return ping packets have an unusually low ***Time To Live*** (***TTL***), you may have routing problems between your machine and the remote end. But what if the ping doesn't return any data at all? Some Internet hosts do not respond to pings for security reasons, so it may help to try another address on the Internet which is known to respond to pings. However, if you are pinging a host name instead of an IP address, you may be running into DNS problems.

To check this, try pinging an IP address on the Internet, instead of a domain name. If that works, then you have a problem with your DNS server. If you don't get a response from the IP address, try to ping your default router:

```
$ ping 216.231.38.1
PING 216.231.38.1 (216.231.38.1): 56 data bytes
64 bytes from 216.231.38.1: icmp_seq=0 ttl=126 time=12.991 ms
64 bytes from 216.231.38.1: icmp_seq=1 ttl=126 time=14.869 ms
```

```
64 bytes from 216.231.38.1: icmp_seq=2 ttl=126 time=13.897 ms
^C
--- 216.231.38.1 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max/stddev = 12.991/13.919/14.869/0.767 ms
```

If you can't ping your default router, then chances are you won't be able to get to the Internet either. If you can't even ping other IP addresses on your local LAN, then it's time to check your connection. If you're using Ethernet, is it plugged in? If you're using wireless, are you connected to the proper wireless network, and is it in range?

Network debugging with ping is a bit of an art, but it is useful to learn. Since you will likely find ping on just about any machine you will work on, it's a good idea to learn how to use it well.

traceroute and mtr

As with ping, ***traceroute*** is found on most operating systems (it's called ***tracert*** in some versions of Microsoft Windows). By running traceroute, you can find the location of problems between your computer and any point on the Internet:

```
$ traceroute -n google.com
traceroute to google.com (72.14.207.99), 64 hops max, 40 byte packets
 1  10.15.6.1  4.322 ms  1.763 ms  1.731 ms
 2  216.231.38.1  36.187 ms  14.648 ms  13.561 ms
 3  69.17.83.233  14.197 ms  13.256 ms  13.267 ms
 4  69.17.83.150  32.478 ms  29.545 ms  27.494 ms
 5  198.32.176.31  40.788 ms  28.160 ms  28.115 ms
 6  66.249.94.14  28.601 ms  29.913 ms  28.811 ms
 7  172.16.236.8  2328.809 ms  2528.944 ms  2428.719 ms
 8  * * *
```

The **-n** switch tells traceroute not to bother resolving names in DNS, and makes the trace run more quickly. You can see that at hop seven, the round trip time shoots up to more than two seconds, while packets seem to be discarded at hop eight. This might indicate a problem at that point in the network. If this part of the network is in your control, it might be worth starting your troubleshooting effort there. Note that traceroute tools can only indicate the path forward from the local machine to the remote end, but cannot display the actual return path.

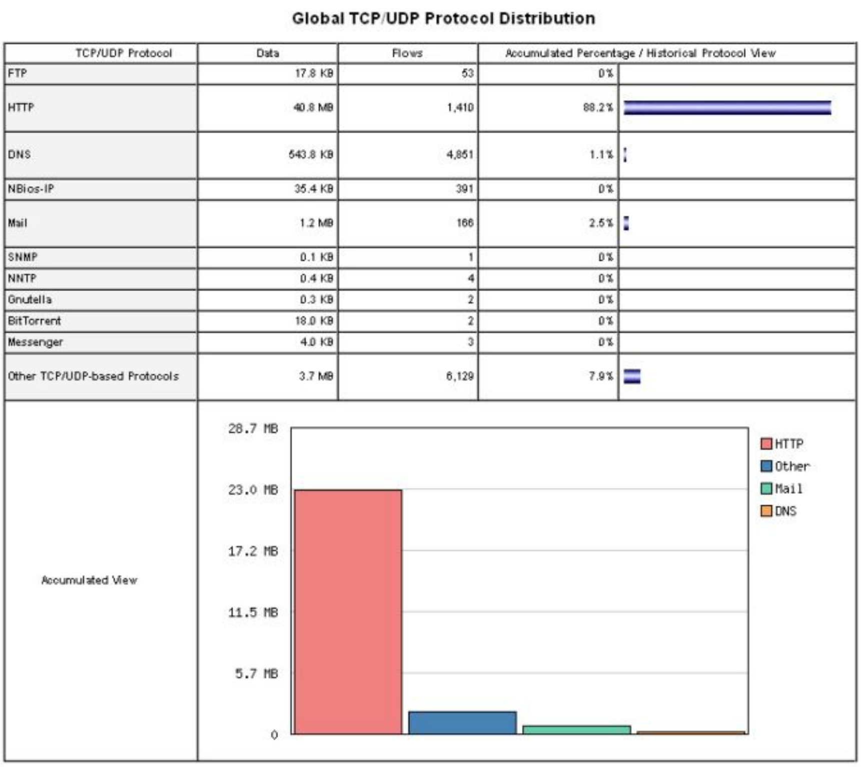
My TraceRoute (***mtr***) is a handy program that combines ping and traceroute into a single tool. It is available at *http://www.bitwizard.nl/mtr/.* By running mtr, you can see an ongoing average of round trip times and packet loss to a single host, instead of the momentary snapshot that ping and traceroute provide.

```
                             My traceroute  [v0.69]
tesla.rob.swn (0.0.0.0)        (tos=0x0 psize=64 bitpatSun Jan  8 20:01:26 2006
Keys:  Help   Display mode   Restart statistics   Order of fields   quit
                                       Packets               Pings
 Host                                Loss%   Snt   Last   Avg  Best  Wrst StDev
 1. gremlin.rob.swn                   0.0%     4    1.9   2.0   1.7   2.6   0.4
 2. er1.sea1.speakeasy.net            0.0%     4   15.5  14.0  12.7  15.5   1.3
 3. 220.ge-0-1-0.cr2.sea1.speakeasy.  0.0%     4   11.0  11.7  10.7  14.0   1.6
 4. fe-0-3-0.cr2.sfo1.speakeasy.net   0.0%     4   36.0  34.7  28.7  38.1   4.1
 5. bas1-m.pao.yahoo.com              0.0%     4   27.9  29.6  27.9  33.0   2.4
 6. so-1-1-0.pat1.dce.yahoo.com       0.0%     4   89.7  91.0  89.7  93.0   1.4
 7. ae1.p400.msr1.dcn.yahoo.com       0.0%     4   91.2  93.1  90.8  99.2   4.1
 8. ge5-2.bas1-m.dcn.yahoo.com        0.0%     4   89.3  91.0  89.3  93.4   1.9
 9. w2.rc.vip.dcn.yahoo.com           0.0%     3   91.2  93.1  90.8  99.2   4.1
```

The data will be continuously updated and averaged over time. As with ping, you should hit control-C when you are finished looking at the data. Note that you must have root privileges to run mtr.

ntop

Ntop is probably one of the most useful traffic monitoring programs around. It is essentially a network protocol analyser with an intuitive built-in web interface that delivers a wealth of information.



Global TCP/UDP Protocol Distribution

TCP/UDP Protocol	Data	Flows	Accumulated Percentage / Historical Protocol View	
FTP	17.8 KB	53	0%	
HTTP	40.8 MB	1,410	88.2%	
DNS	543.8 KB	4,851	1.1%	
NBios-IP	35.4 KB	391	0%	
Mail	1.2 MB	166	2.5%	
SNMP	0.1 KB	1	0%	
NNTP	0.4 KB	4	0%	
Gnutella	0.3 KB	2	0%	
BitTorrent	18.0 KB	2	0%	
Messenger	4.0 KB	3	0%	
Other TCP/UDP-based Protocols	3.7 MB	6,129	7.9%	
Accumulated View				

Figure 3.20: ntop displays a wealth of information about how your network is utilised by various clients and servers.

Some of its more useful features include:

- Traffic display can be sorted by various criteria (source, destination, protocol, MAC address, etc.).
- Traffic statistics grouped by protocol and port number
- An IP traffic matrix which shows connections between machines
- Network flows for routers or switches that support the NetFlow protocol
- Host operating system identification
- P2P traffic identification
- Numerous graphical charts
- Perl, PHP, and Python API

Ntop is available from *http://www.ntop.org/* and is available for most operating systems. It is often included in many of the popular Linux distributions, including RedHat, Debian, and Ubuntu. While it can be left running to collect historical data, ntop can be fairly CPU intensive, depending on the amount of traffic observed. If you are going to run it for long periods you should monitor the CPU utilisation of the monitoring machine.

The main disadvantage of ntop is that it does not provide instantaneous information, only long-term totals and averages. This can make it difficult to use to diagnose a problem that starts suddenly.

iptraf

IPTraf is a lightweight but powerful LAN monitor. It has an ncurses interface and runs in a command shell. IPTraf takes a moment to measure observed traffic, and then displays various network statistics including TCP and UDP connections, ICMP and OSPF information, traffic flows, IP checksum errors, and more. It is a simple to use program that uses minimal system resources.

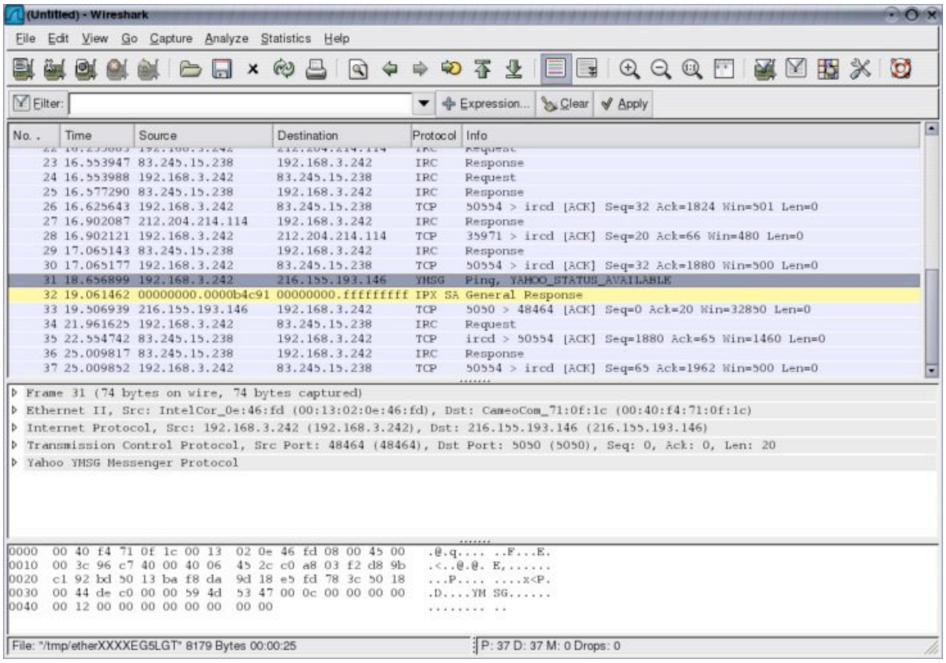
While it does not keep historical data, it is very useful for displaying an instantaneous usage report. IPTraf is available from *http://iptraf.seul.org/.*

tcpdump

Tcpdump (*http://www.tcpdump.org/*) is a command-line tool for monitoring network traffic. It does not have all the bells and whistles of wireshark (page **78**) but it does use fewer resources. Tcpdump can capture and display all network protocol information down to the link layer. It can show all of the packet headers and data received, or just the packets that match particular criteria. Packets captured with tcpdump can be loaded into wireshark for visual analysis and further diagnostics. This is very useful if you wish to monitor an interface on a remote system and bring the file back to your local machine for analysis. The tcpdump tool is available as a standard tool in Unix derivatives (Linux, BSD, and Mac OS X). There is also a Windows port called ***WinDump*** available at *http://www.winpcap.org/windump/.*

Wireshark (Ethereal)

Wireshark (formerly known as Ethereal) is a free network protocol analyser for Unix and Windows, and is billed as "The World's Most Popular Network Protocol Analyser." It allows you to examine data from a live network or from a capture file on disk, and interactively browse and sort the captured data. Both summary and detailed information is available for each packet, including the full header and data portions. Wireshark has several powerful features, including a rich display filter language and the ability to view the reconstructed stream of a TCP session.



```
No. .  Time      Source              Destination         Protocol Info
    23 16.553947 83.245.15.238       192.168.3.242       IRC      Response
    24 16.553988 192.168.3.242       83.245.15.238       IRC      Request
    25 16.577290 83.245.15.238       192.168.3.242       IRC      Response
    26 16.625643 192.168.3.242       83.245.15.238       TCP      50554 > ircd [ACK] Seq=32 Ack=1824 Win=501 Len=0
    27 16.902087 212.204.214.114     192.168.3.242       IRC      Response
    28 16.902121 192.168.3.242       212.204.214.114     TCP      35971 > ircd [ACK] Seq=20 Ack=66 Win=480 Len=0
    29 17.065143 83.245.15.238       192.168.3.242       IRC      Response
    30 17.065177 192.168.3.242       83.245.15.238       TCP      50554 > ircd [ACK] Seq=32 Ack=1880 Win=500 Len=0
    31 18.656899 192.168.3.242       216.155.193.146     YMSG     Ping, YAHOO_STATUS_AVAILABLE
    32 19.061462 00000000.0000b4c91 00000000.ffffffff IPX SA General Response
    33 19.506939 216.155.193.146     192.168.3.242       TCP      5050 > 48464 [ACK] Seq=0 Ack=20 Win=32850 Len=0
    34 21.961625 192.168.3.242       83.245.15.238       IRC      Request
    35 22.554742 83.245.15.238       192.168.3.242       TCP      ircd > 50554 [ACK] Seq=1880 Ack=65 Win=1460 Len=0
    36 25.009817 83.245.15.238       192.168.3.242       IRC      Response
    37 25.009852 192.168.3.242       83.245.15.238       TCP      50554 > ircd [ACK] Seq=65 Ack=1962 Win=500 Len=0

▷ Frame 31 (74 bytes on wire, 74 bytes captured)
▷ Ethernet II, Src: IntelCor_0e:46:fd (00:13:02:0e:46:fd), Dst: CameoCom_71:0f:1c (00:40:f4:71:0f:1c)
▷ Internet Protocol, Src: 192.168.3.242 (192.168.3.242), Dst: 216.155.193.146 (216.155.193.146)
▷ Transmission Control Protocol, Src Port: 48464 (48464), Dst Port: 5050 (5050), Seq: 0, Ack: 0, Len: 20
▷ Yahoo YMSG Messenger Protocol

0000  00 40 f4 71 0f 1c 00 13  02 0e 46 fd 08 00 45 00   .@.q.... ..F...E.
0010  00 3c 96 c7 40 00 40 06  45 2c c0 a8 03 f2 d8 9b   .<..@.@. E,......
0020  c1 92 bd 50 13 ba f8 da  9d 18 e5 fd 78 3c 50 18   ...P.... ....x<P.
0030  00 44 de c0 00 00 59 4d  53 47 00 0c 00 00 00 00   .D....YM SG......
0040  00 12 00 00 00 00 00 00  00 00                     ........ ..

File: "/tmp/etherXXXXEG5LGT" 8179 Bytes 00:00:25   P: 37 D: 37 M: 0 Drops: 0
```

Figure 3.21: Wireshark (formerly Ethereal) is a powerful network protocol analyser that can show you as much detail as you like about any packet that crosses the wire.

It can be daunting to use for first time users or those that are not familiar with the OSI layers. It is typically used to isolate and analyse specific traffic to or from an IP address, but it can be also used as a general purpose fault finding tool. For example, a machine infected with a network worm or virus can be identified by looking for the machine that is send out the same sort of TCPIP packets to large groups of IP addresses.

Wireshark is available from *http://www.wireshark.org/.* For examples of how to use wireshark to debug specific network problems, see chapter five, **Troubleshooting**.

ngrep

Ngrep provides most of GNU grep's pattern matching features, but applies them to network traffic. It currently recognises IPv4 and IPv6, TCP, UDP, ICMP, IGMP, PPP, SLIP, FDDI, Token Ring, and much more. As it makes extensive use of regular expression matches, it is a tool suited to advanced users or those that have a good knowledge of regular expressions.

But you don't necessarily need to be a regex expert to be able to make basic use of ngrep. For example, to view all packets that contain the string GET (presumably HTTP requests), try this:

```
# ngrep -q GET
```

Pattern matches can be constrained further to match particular protocols, ports, or other criteria using BPF filters. This is the filter language used by common packet sniffing tools, such as tcpdump and snoop. To view GET or POST strings sent to destination port 80, use this command line:

```
# ngrep -q 'GET|POST' port 80
```

By using ngrep creatively, you can detect anything from virus activity to spam email. You can download ngrep at *http://ngrep.sourceforge.net/.*

EtherApe

EtherApe (*http://etherape.sourceforge.net/*) displays a graphical representation of network traffic. Hosts and links change size depending on the amount of traffic sent and received. The colors change to represent the protocol most used. As with wireshark and tcpdump, data can be captured "off the wire" from a live network connection or read from a tcpdump capture file.

EtherApe doesn't show quite as much detail as ntop, but its resource requirements are much lighter.

Argus

Argus (*http://qosient.com/argus/*) stands for ***Audit Record Generation and Utilization System***. Argus is also the name of the mythological Greek god who had hundreds of eyes.

From the Argus website:

> *Argus generates flow statistics such as connectivity, capacity, demand, loss, delay, and jitter on a per transaction basis. Argus can be used to analyse and report on the contents of packet capture files or it can run as a continuous monitor, examining data from a live interface; generating an audit log of all the network activity seen in the packet stream. Argus can be deployed to monitor individual end-systems, or an entire enterprises network activity. As a continuous monitor, Argus provides both push and pull data handling models, to allow flexible strategies for collecting network audit data. Argus data clients support a range of operations, such as sorting, aggregation, archival and reporting.*

Argus consists of two parts: a master collector that reads packets from a network device, and a client that connects to the master and displays the usage statistics. Argus runs on BSD, Linux, and most other UNIX systems.

NeTraMet

NeTraMet (*http://www.auckland.ac.nz/net/NeTraMet/*) is another popular flow analysis tool. Like Argus, NeTraMet consists of two parts: a collector that gathers statistics via SNMP, and a manager that specifies which flows should be watched. Flows are specified using a simple programming language that define the addresses used on either end, and can include Ethernet, IP, protocol information, or other identifiers. NeTraMet runs on DOS and most UNIX systems, including Linux and BSD.

Log analysers

Information that has been derived from logging and trending tools can be very useful in determining the expected future usage patterns of your network. While some spot check tools can be used over long periods to capture information, many of them will generate a lot of data. Managing the storage and archiving of this data can be tricky.

Often, simple data derived from log files can be just as useful in determining where potential problem areas lie. Log analysers and trending tools tend to require much less disk space and CPU power than spot check tools. Here are several popular and freely available logging and trending tools.

Firewall logs

Your firewall can generate a potentially huge amount of statistical information. This can be used to compile a list of top bandwidth users (or abusers), and identify users that find holes in your firewall rules to run bandwidth intense applications such as peer-to-peer programs. If you are running a commercial firewall, it usually includes a reporting feature to notify you of abnormal traffic patterns. If you are running an open source firewall under Linux or BSD, there are any number of log file reporters available. Some of these include:

- IPTables log analyser, *http://www.gege.org/iptables/*
- ADMLogger, *http://aaron.marasco.com/linux.html*
- adcfw-log, *http://adcfw-log.sourceforge.net/*
- logwatch, *http://www.logwatch.org/*

To use a log watcher with your firewall, you will need to configure your firewall to log the packets you want to watch. For example, you could configure your firewall to log packets bound for port 445, a port used by the Sasser worm. Analysis of activity on this port may show an attacker's IP address, from which you would want to block traffic.

Proxy (cache) logs

Proxy logs contain extensive details about the browsing habits of your users. By analysing this information, you can find users who make inappropriate use of network resources, and identify sites that may benefit from mirroring (page **144**). These logs can also help pinpoint potential problem areas on your network, such as viruses. While we will discuss Squid cache logs in particular, these techniques will work with many different kinds of cache servers.

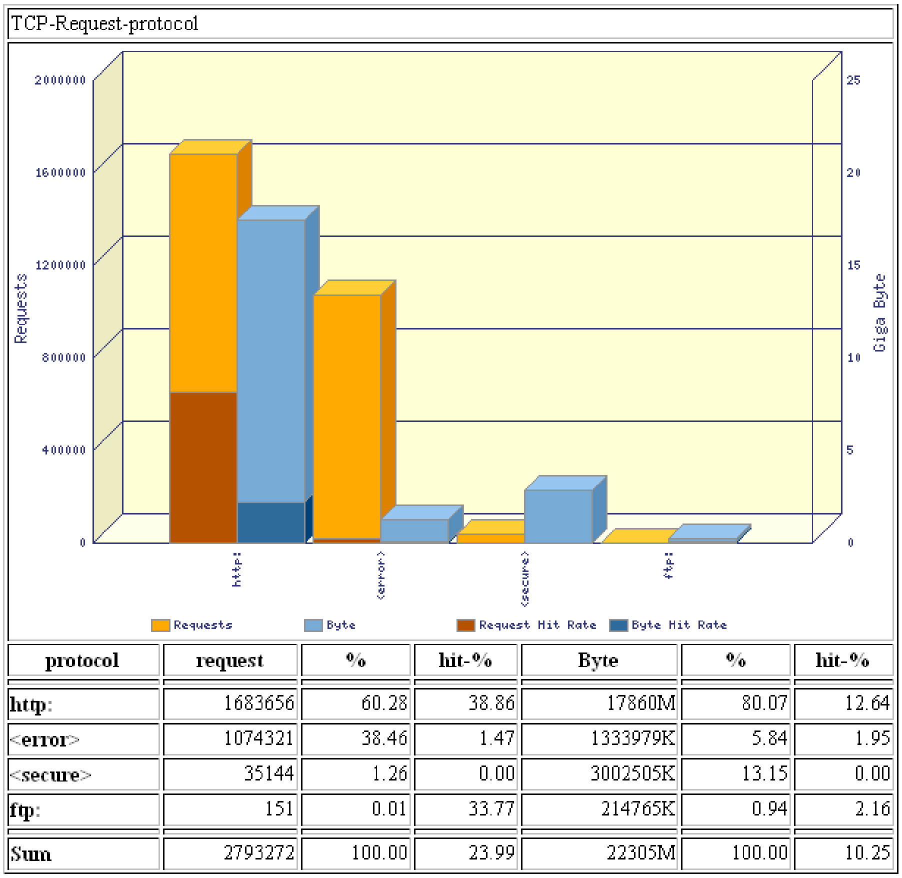
Unless otherwise specified, Squid log files use a standard web server log format. Popular log file analysers such as ***Analog*** (*http://www.analog.cx/*), ***Webalizer*** (*http://www.mrunix.net/webalizer/*), and ***AWStats*** (*http://awstats.sourceforge.net/*) will read Squid log files and produce detailed reports on the data collected. The Squid project site has a page full of links to compatible log file analysers at *http://www.squid-cache.org/Scripts/*.

One popular analyser is ***Calamaris***, *http://cord.de/tools/squid/calamaris/.* Calamaris produces impressive html reports and graphs of proxy utilisation. It is written in Perl, and requires gdlib to be installed to display graphs. Calamaris, perl, and gdlib are available on most Linux distributions as standard packages. Calamaris is shown in Figure 3.22.

Some people are frustrated by the use of the Unix epoch time format used in the Squid logs. This perl snippet:

```
tail -f /var/log/squid/access.log | perl -pe 's/^\d+/localtime $&/e;'
```

...will convert the timestamps into a human readable format.



protocol	request	%	hit-%	Byte	%	hit-%
http:	1683656	60.28	38.86	17860M	80.07	12.64
<error>	1074321	38.46	1.47	1333979K	5.84	1.95
<secure>	35144	1.26	0.00	3002505K	13.15	0.00
ftp:	151	0.01	33.77	214765K	0.94	2.16
Sum	2793272	100.00	23.99	22305M	100.00	10.25

Figure 3.22: Calamaris will analyse your Squid logs to give you informative reports about your users' web traffic patterns.

Another popular log analyser is ***Sawmill*** (*http://www.sawmill.net/*). It is commercial software, which among other things means that it comes with technical support. Sawmill can produce more detailed and professional-looking reports than many open source alternatives. It also has some interesting advanced features, such as geographic location reports. Sawmill runs on Unix and Windows. It requires a commercial license based on the type and amount of data to be analysed.

In the same way that there are programs for analysing web and cache server logs, there are programs for analysing mail server logs. Using these, it is possible to better understand the demands that email places on your network. The open source program ***Isoqlog*** (*http://www.enderunix.org/isoqlog/*) understands the log files generated by four commonly-used UNIX mail servers (qmail, postfix, sendmail and exim), and generates statistics from these.

Trending tools

Trending tools are used to see how your network is used over a long period of time. They work by periodically monitoring your network activity, and displaying a summary in a human-readable form (such as a graph). Unlike log analysers, trending tools collect data as well as analyse and reporting on it.

Below are some examples of trending tools. Some of them need to be used in conjunction with each other, as they are not stand-alone programs.

MRTG

The ***Multi Router Traffic Grapher*** (***MRTG***) monitors the traffic load on network links using SNMP. MRTG generates graphs that provide a visual representation of inbound and outbound traffic. These are typically displayed on a web page. MRTG is available from *http://oss.oetiker.ch/mrtg/.*

MRTG can be a little confusing to set up, especially if you are not familiar with SNMP. But once it is installed, MRTG requires virtually no maintenance, unless you change something on the system that is being monitored (such as its IP address).

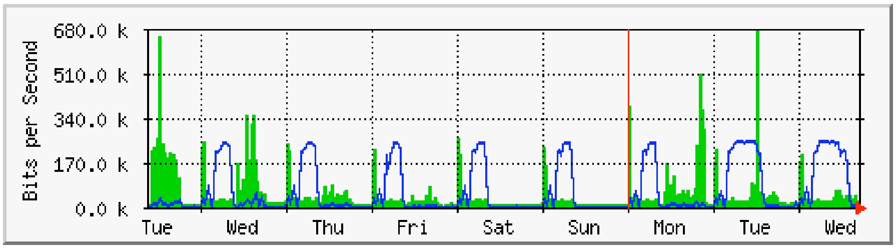


Figure 3.23: MRTG is probably the most widely installed network flow grapher.

RRDtool

RRD is short for ***Round Robin Database***. RRD is a database that stores information in a very compact way that does not expand over time. ***RRDtool*** refers to a suite of tools that allow you to create and modify RRD databases, as well as generate useful graphs to present the data. It is used to keep track of time-series data (such as network bandwidth, machine room temperature, or server load average) and can display that data as an average over time.

Note that RRDtool itself does not contact network devices to retrieve data. It is merely a database manipulation tool. You can use a simple wrapper script (typically in shell or Perl) to do that work for you. RRDtool is also used by many full featured front-ends that present you with a friendly web interface for configuration and display. RRD graphs give you more control over display options and the number of items available on a graph as compared to MRTG.

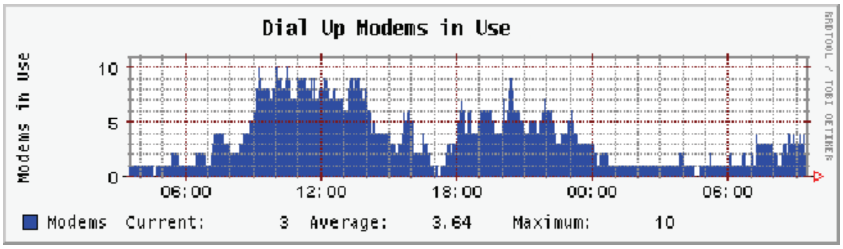
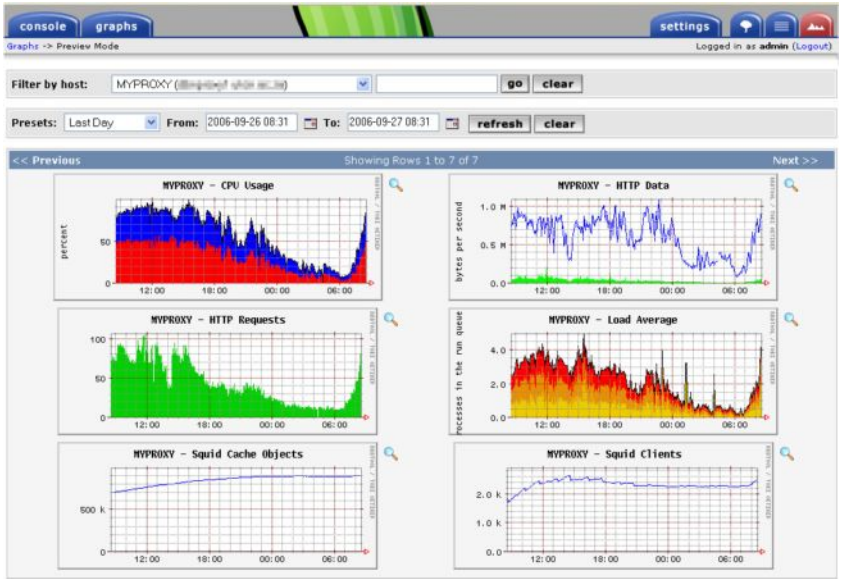


Figure 3.24: RRDtool gives you a lot of flexibility in how your collected network data may be displayed.

RRDtool is included in virtually all modern Linux distributions, and can be downloaded from *http://oss.oetiker.ch/rrdtool/.*

Cacti

Cacti (*http://www.cacti.net/*) is one such front-end for RRDtool. It stores all of the necessary information to create graphs in a MySQL database. The front-end is written in PHP. Cacti does the work of maintaining graphs, data sources, and handles the actual data gathering. There is support for SNMP devices, and custom scripts can easily be written to poll virtually any conceivable network event.



Filter by host: MYPROXY ([redacted])

Presets: Last Day From: 2006-09-26 08:31 To: 2006-09-27 08:31

Figure 3.25: Cacti can manage the polling of your network devices, and can build very complex and informative visualisations of network behaviour.

Cacti can be somewhat confusing to configure, but once you work through the documentation and examples, it can yield very impressive graphs. There are hundreds of templates for various systems available on the cacti website, and the code is under rapid development.

NetFlow

NetFlow is a protocol for collecting IP traffic information invented by Cisco. From the Cisco website:

> *Cisco IOS NetFlow efficiently provides a key set of services for IP applications, including network traffic accounting, usage-based network billing, network planning, security, Denial of Service monitoring capabilities, and network monitoring. NetFlow provides valuable information about network users and applications, peak usage times, and traffic routing.*

Cisco routers can generate NetFlow information which is available from the router in the form of UDP packets. NetFlow is also less CPU-intensive on Cisco routers than using SNMP. It also provides more granular information than SNMP, letting you get a more detailed picture of port and protocol usage.

This information is collected by a NetFlow collector that stores and presents the data as an aggregate over time. By analysing flow data, one can build a picture of traffic flow and traffic volume in a network or on a connection. There are several commercial and free NetFlow collectors available. Ntop (page **76**) is one free tool that can act as a NetFlow collector and probe. Another is Flowc, which is described in detail on page **86**.

It can also be desirable to use Netflow as a spot check tool, by just looking at a quick snapshot of data during a network crisis. Think of NetFlow as an alternative to SNMP for Cisco devices. For more information about NetFlow, see *http://en.wikipedia.org/wiki/Netflow* .

SmokePing

SmokePing (*http://oss.oetiker.ch/smokeping/*) is a deluxe latency measurement tool written in Perl. It can measure, store and display latency, latency distribution and packet loss all on a single graph. SmokePing uses RRDtool for data storage, and can draw very informative graphs that present up to the minute information on the state of your network connection.

It is very useful to run SmokePing on a host with good connectivity to your entire network. Over time, trends are revealed that can point to all sorts of network problems. Combined with MRTG (page **83**) or Cacti (page **84**), you can observe the effect that network congestion has on packet loss and latency. SmokePing can optionally send alerts when certain conditions are met, such as when excessive packet loss is seen on a link for an extended period of time. An example of SmokePing in action is shown in Figure 3.26.

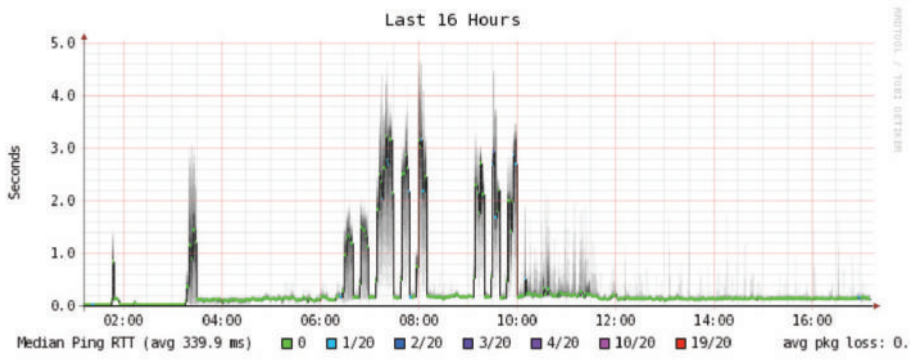


Figure 3.26: SmokePing can simultaneously display packet loss and latency spreads in a single graph.

Flowc

Flowc (*http://netacad.kiev.ua/flowc/*) is an open source NetFlow collector (see NetFlow above). It is lightweight and easy to configure. Flowc uses a MySQL database to store aggregated traffic information. Therefore, it is possible to generate your own reports from the data using SQL, or use the included report generators. The built-in report generators produce reports in HTML, plain text or a graphical format.

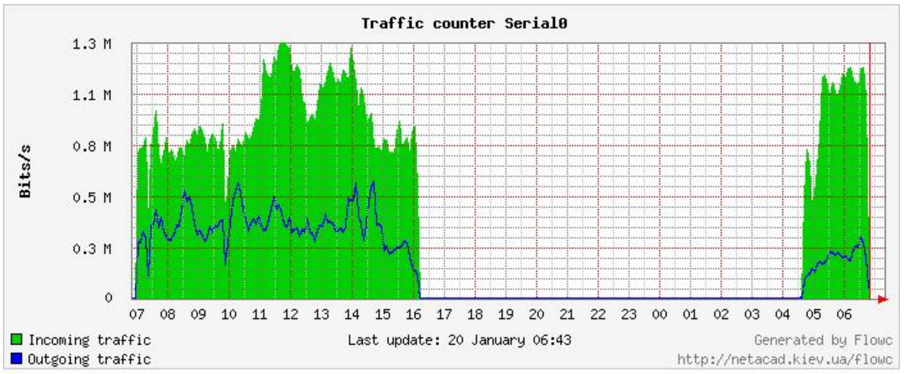


Figure 3.27: A typical flow chart generated by Flowc.

The large gap in data probably indicates a network outage. Trending tools typically will not notify you of outages, but merely log the occurrence. To be notified when network problems occur, use a realtime monitoring tool such as Nagios (page **88**).

Realtime tools

It is desirable to find out when people are trying to break into your network, or when some part of the network has failed. Because no system administrator can be monitoring a network all the time, there are programs that constantly monitor the status of the network and can send alerts when notable events occur. The following are some open source tools that can help perform this task.

Snort

Snort (*http://www.snort.org/*) is a packet sniffer and logger which can be used as a lightweight network intrusion detection system. It features rule-based logging and can perform protocol analysis, content searching, and packet matching. It can be used to detect a variety of attacks and probes, such as stealth port scans, CGI attacks, SMB probes, OS fingerprinting attempts, and many other kinds of anomalous traffic patterns. Snort has a real-time alert capability that can notify administrators about problems as they occur with a variety of methods.

Installing and running Snort is not trivial, and depending on the amount of network traffic, will likely require a dedicated monitoring machine with considerable resources. Fortunately, Snort is very well documented and has a strong user community. By implementing a comprehensive Snort rule set, you can identify unexpected behaviour that would otherwise mysteriously eat up your Internet bandwidth.

See *http://snort.org/docs/* for an extensive list of installation and configuration resources.

Apache: mod_security

ModSecurity (*http://www.modsecurity.org/*) is an open source intrusion detection and prevention engine for web applications. This kind of security tool is also known as a ***web application firewall***. ModSecurity increases web application security by protecting web applications from known and unknown attacks. It can be used on its own, or as a module in the Apache web server (*http://www.apache.org/*).

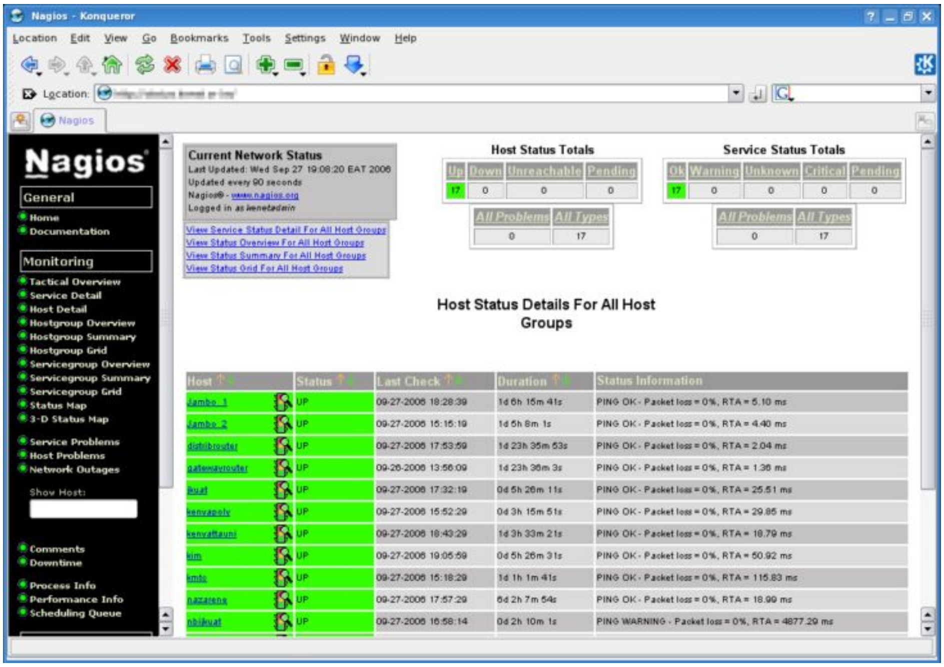
There are several sources for updated mod_security rules that help protect against the latest security exploits. One excellent resource is GotRoot, which maintains a huge and frequently updated repository of rules:

http://gotroot.com/tiki-index.php?page=mod_security+rules

Web application security is important in defending against attacks on your web server, which could result in the theft of valuable or personal data, or in the server being used to launch attacks or send spam to other Internet users. As well as being damaging to the Internet as a whole, such intrusions can seriously reduce your available bandwidth.

Nagios

Nagios (*http://nagios.org/*) is a program that monitors hosts and services on your network, notifying you immediately when problems arise. It can send notifications via email, SMS, or by running a script, and will send notifications to the relevant person or group depending on the nature of the problem. Nagios runs on Linux or BSD, and provides a web interface to show up-to-the-minute system status.



Current Network Status
Last Updated: Wed Sep 27 19:08:20 EAT 2006
Updated every 90 seconds
Nagios® - www.nagios.org
Logged in as kenetadmin

View Service Status Detail For All Host Groups
View Status Overview For All Host Groups
View Status Summary For All Host Groups
View Status Grid For All Host Groups

Host Status Totals

Up	Down	Unreachable	Pending
17	0	0	0

All Problems	All Types
0	17

Service Status Totals

Ok	Warning	Unknown	Critical	Pending
17	0	0	0	0

All Problems	All Types
0	17

Host Status Details For All Host Groups

Host	Status	Last Check	Duration	Status Information
Jambo_1	UP	09-27-2006 18:28:39	1d 6h 15m 41s	PING OK - Packet loss = 0%, RTA = 5.10 ms
Jambo_2	UP	09-27-2006 15:15:19	1d 5h 8m 1s	PING OK - Packet loss = 0%, RTA = 4.40 ms
distribrouter	UP	09-27-2006 17:53:59	1d 23h 35m 53s	PING OK - Packet loss = 0%, RTA = 2.04 ms
gatewayrouter	UP	09-26-2006 13:56:09	1d 23h 36m 3s	PING OK - Packet loss = 0%, RTA = 1.36 ms
jkuat	UP	09-27-2006 17:32:19	0d 5h 26m 11s	PING OK - Packet loss = 0%, RTA = 25.51 ms
kenyapoly	UP	09-27-2006 15:52:29	0d 3h 15m 51s	PING OK - Packet loss = 0%, RTA = 29.85 ms
kenyattauni	UP	09-27-2006 18:43:29	1d 3h 33m 21s	PING OK - Packet loss = 0%, RTA = 18.79 ms
kim	UP	09-27-2006 19:05:59	0d 5h 26m 31s	PING OK - Packet loss = 0%, RTA = 50.92 ms
kmtc	UP	09-27-2006 16:18:29	1d 1h 1m 41s	PING OK - Packet loss = 0%, RTA = 115.83 ms
nazarene	UP	09-27-2006 17:57:29	6d 2h 7m 54s	PING OK - Packet loss = 0%, RTA = 18.99 ms
nbijkuat	UP	09-27-2006 16:58:14	0d 2h 10m 1s	PING WARNING - Packet loss = 0%, RTA = 4877.29 ms

Figure 3.28: Nagios keeps you informed the moment a network fault or service outage occurs.

Nagios is extensible, and can monitor the status of virtually any network event. It performs checks by running small scripts at regular intervals, and checks the results against an expected response. This can yield much more sophisticated checks than a simple network probe. For example, ping (page **74**) may tell you that a machine is up, and nmap may report that a TCP port responds to requests, but Nagios can actually retrieve a web page or make a database request, and verify that the response is not an error.

Nagios can even notify you when bandwidth usage, packet loss, machine room temperature, or other network health indicator crosses a particular threshold. This can give you advance warning of network problems, often allowing you to respond to the problem before users have a chance to complain.

Zabbix

Zabbix (*http://www.zabbix.org/*) is an open source realtime monitoring tool that is something of a hybrid between Cacti and Nagios. It uses a SQL database for data storage, has its own graph rendering package, and performs all of the functions you would expect from a modern realtime monitor (such as SNMP polling and instant notification of error conditions). Zabbix is released under the GNU General Public License.

Benchmarking

How fast can the network go? What is the actual usable capacity of a particular network link? You can get a very good estimate of your throughput capacity by flooding the link with traffic and measuring how long it takes to transfer the data.

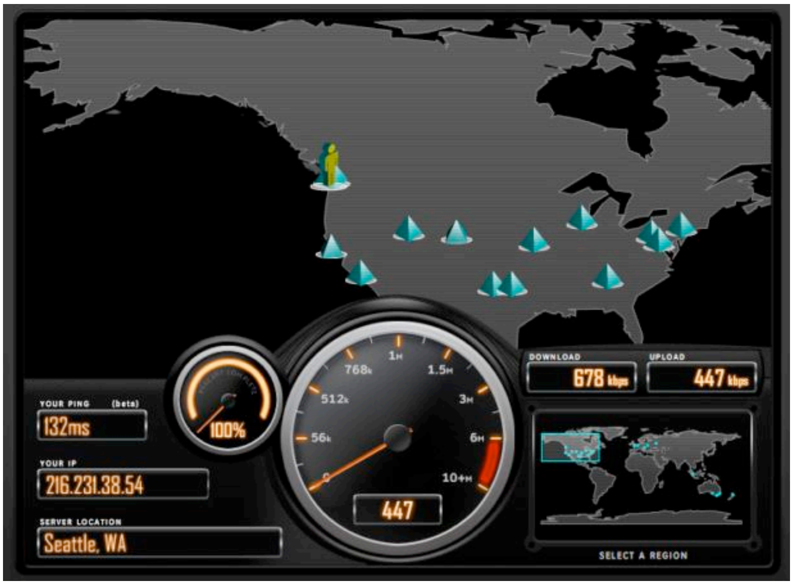


Figure 3.29: Tools such as this one from SpeedTest.net are pretty, but don't always give you an accurate picture of network performance.

While there are web pages available that will perform a "speed test" in your browser (such as *http://www.dslreports.com/stest* or *http://speedtest.net/*), these tests are increasingly inaccurate as you get further from the testing source. Even worse, they do not allow you to test the speed of a given link, but only the speed of your link to a particular site on the Internet. Here are three tools that will allow you to perform throughput testing on your own networks.

- **ttcp** (*http://ftp.arl.mil/ftp/pub/ttcp/*). Now a standard part of most Unix-like systems, ttcp is a simple network performance testing tool. One instance is run on either side of the link you want to test. The first node runs in receive mode, and the other transmits:

```
node_a$ ttcp -r -s

node_b$ ttcp -t -s node_a
ttcp-t: buflen=8192, nbuf=2048, align=16384/0, port=5001  tcp -> node_a
ttcp-t: socket
ttcp-t: connect
ttcp-t: 16777216 bytes in 249.14 real seconds = 65.76 KB/sec +++
ttcp-t: 2048 I/O calls, msec/call = 124.57, calls/sec = 8.22
ttcp-t: 0.0user 0.2sys 4:09real 0% 0i+0d 0maxrss 0+0pf 7533+0csw
```

  After collecting data in one direction, you should reverse the transmit and receive partners to test the link in the other direction. It can test UDP as well as TCP streams, and can alter various TCP parameters and buffer lengths to give the network a good workout. It can even use a user-supplied data stream instead of sending random data. Remember that the speed readout is in kilobytes, not kilobits. Multiply the result by 8 to find the speed in kilobits per second.

  The only real disadvantage to ttcp is that it hasn't been developed in years. Fortunately, the code has been released in the public domain and is freely available. Like ping and traceroute, ttcp is found as a standard tool on many systems.

- **iperf** (*http://dast.nlanr.net/Projects/Iperf/*). Much like ttcp, iperf is a commandline tool for estimating the throughput of a network connection. It supports many of the same features as ttcp, but uses a "client" and "server" model instead of a "receive" and "transmit" pair. To run iperf, launch a server on one side and a client on the other:

```
node_a$ iperf -s

node_b$ iperf -c node_a
------------------------------------------------------------
Client connecting to node_a, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  5] local 10.15.6.1 port 1212 connected with 10.15.6.23 port 5001
[ ID] Interval       Transfer     Bandwidth
[  5]  0.0-11.3 sec   768 KBytes   558 Kbits/sec
```

  The server side will continue to listen and accept client connections on port 5001 until you hit control-C to kill it. This can make it handy when running multiple test runs from a variety of locations.

The biggest difference between ttcp and iperf is that iperf is under active development, and has many new features (including IPv6 support). This makes it a good choice as a performance tool when building new networks.

- **bing** (*http://www.freenix.fr/freenix/logiciels/bing.html*). Rather than flood a connection with data and see how long the transfer takes to complete, Bing attempts to estimate the available throughput of a point-to-point connection by analysing round trip times for various sized ICMP packets. While it is not always as accurate as a flood test, it can provide a good estimate without transmitting a large number of bytes.

  Since bing works using standard ICMP echo requests, so it can estimate available bandwidth without the need to run a special client on the other end, and can even attempt to estimate the throughput of links outside your network. Since it uses relatively little bandwidth, bing can give you a rough idea of network performance without running up the charges that a flood test would certainly incur.

What is normal?

If you are looking for a definitive answer as to what your traffic patterns **should** look like, you are going to be disappointed. There is no absolute right answer to this question, but given some work you can determine what is normal for your network. While every environment is different, some of the factors that can influence the appearance of your traffic patterns are:

- The capacity of your Internet connection
- The number of users that have access to the network
- The social policy (byte charging, quotas, honor system, etc.).
- The number, types, and level of services offered
- The health of the network (presence of viruses, excessive broadcasts, routing loops, open email relays, denial of service attacks, etc.).
- The competence of your computer users
- The location and configuration of control structures (firewalls, proxy servers, caches, and so on)

This is not a definitive list, but should give you an idea of how a wide range of factors can affect your bandwidth patterns. With this in mind, let's look at the topic of baselines.

Establishing a baseline

Since every environment is different, you need to determine for yourself what your traffic patterns look like under normal situations. This is useful because it allows you to identify changes over time, either sudden or gradual. These changes may in turn indicate a problem, or a potential future problem, with your network.

For example, suppose that your network grinds to a halt, and you are not sure of the cause. Fortunately, you have decided to keep a graph of broadcasts as a percentage of the overall network traffic. If this graph shows a sudden increase in the amount of broadcast traffic, it may mean that your network has been infected with a virus. Without an idea of what is "normal" for your network (a baseline), you would not be able to see that the number of broadcasts had increased, only that it was relatively high, which may not indicate a problem.

Baseline graphs and figures are also useful when analysing the effects of changes made to the network. It is often very useful to experiment with such changes by trying different possible values. Knowing what the baseline looks like will show you whether your changes have improved matters, or made them worse.

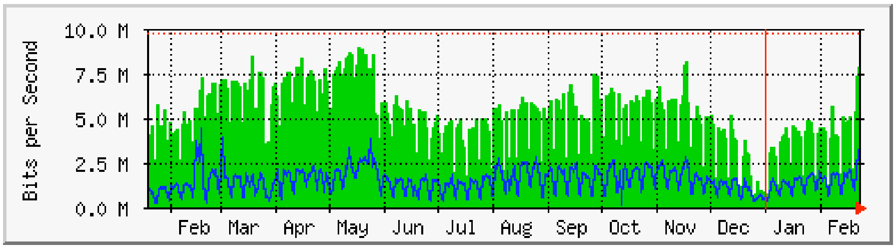


Figure 3.30: By collecting data over a long period of time, you can predict the growth of your network and make changes before problems develop.

In Figure 3.30, we can see the effect the implementation of delay pools has made on Internet utilisation around the period of May. If we did not keep a graph of the line utilisation, we would never know what the effect of the change over the long term was. When watching a total traffic graph after making changes, don't assume that just because the graph does not change radically that your efforts were wasted. You might have removed frivolous usage from your line only to have it replaced by genuine legitimate traffic. You could then combine this baseline with others, say the top 100 sites accessed or the average utilisation by your top twenty users, to determine if habits have simply changed. As we will see later, MRTG, RRDtool, and Cacti are excellent tools you can use to keep a baseline.

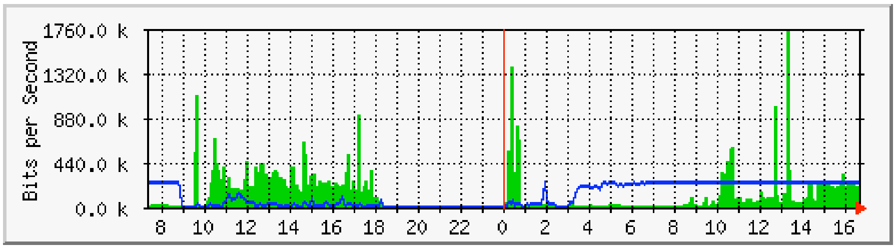


Figure 3.31: The traffic trend at Aidworld logged over a single day.

Figure 3.31 shows traffic on an Aidworld firewall over a period of 24 hours. There is nothing apparently wrong with this graph, but users were complaining about slow Internet access.

Figure 3.32 shows that the upload bandwidth use (dark area) was higher during working hours on the last day than on previous days. A period of heavy upload usage started every morning at 03:00, and was normally finished by 09:00, but on the last day it was still running at 16:30. Further investigation revealed a problem with the backup software, which ran at 03:00 every day.

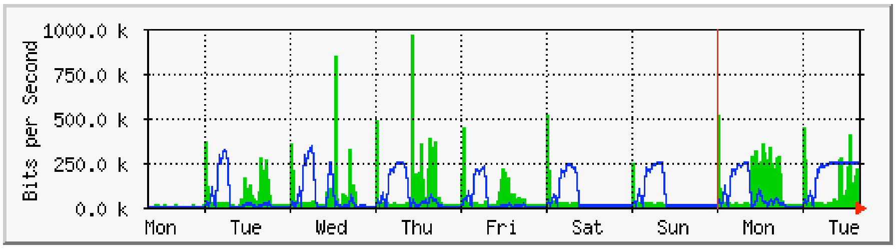


Figure 3.32: The same network logged over an entire week reveals a problem with backups, which caused unexpected congestion for network users.

Figure 3.33 shows measurements of latency on the same connection as measured by a program called SmokePing. The position of the dots shows the average latency, while the gray smoke indicates the distribution of latency (jitter). The color of the dots indicates the number of lost packets. This graph over a period of four hours does not help to identify whether there are any problems on the network.

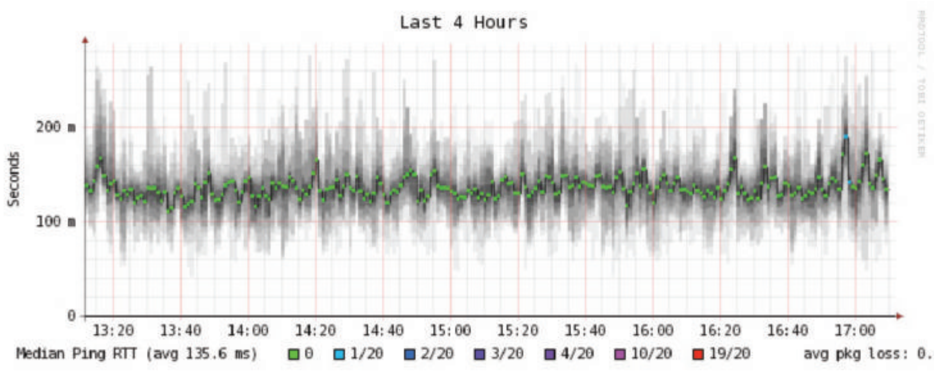


Figure 3.33: Four hours of jitter and packet loss.

The next graph (figure 3.34) shows the same data over a period of 16 hours. This indicates that the values in the graph above are close to the normal level (baseline), but that there were significant increases in latency at several times during the early morning, up to 30 times the baseline value. This indicates that additional monitoring should be performed during these early morning periods to establish the cause of the high latency, which is probably heavy traffic of some kind.

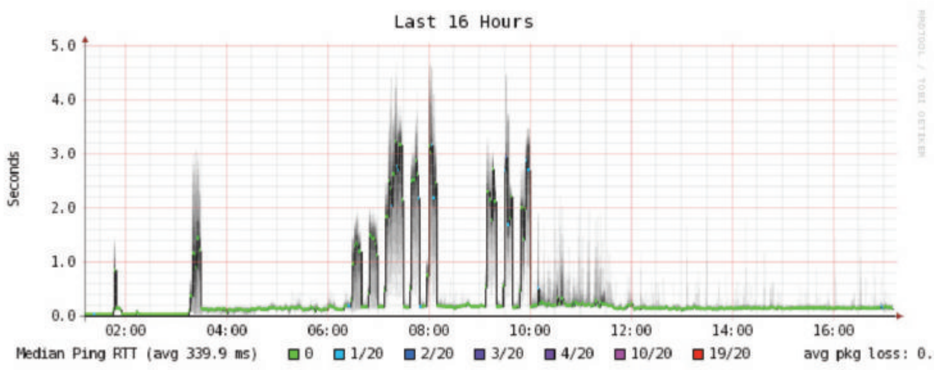


Figure 3.34: A higher spread of jitter is revealed in the 16 hour log.

Figure 3.35 shows that Tuesday was significantly worse than Sunday or Monday for latency, especially during the early morning period. This might indicate that something has changed on the network.

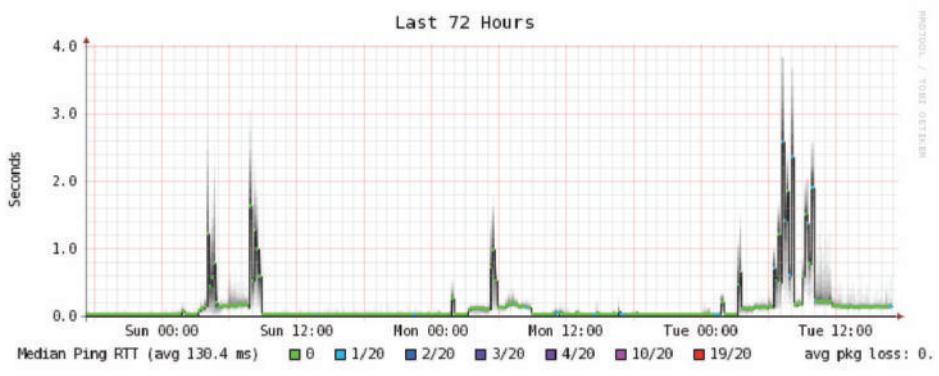


Figure 3.35: Zooming out to the week long view reveals a definite repetition of increased latency and packet loss in the early morning hours.

How do I interpret the traffic graph?

In a basic network flow graph (such as that generated by the network monitor MRTG), the green area indicates ***inbound traffic***, while the blue line indicates ***outbound traffic***. Inbound traffic is traffic that originates from another network (typically the Internet) and is addressed to a computer inside your network. Outbound traffic is traffic that originates from your network, and is addressed to a computer somewhere on the Internet. Depending on what sort of network environment you have, the graph will help you understand how your network is actually being used. For example, monitoring of servers usually reveals larger amounts of outbound traffic as the servers respond to requests (such as sending mail or serving web pages), while monitoring client machines might reveal higher amounts of inbound traffic to the machines as they receive data from the servers.

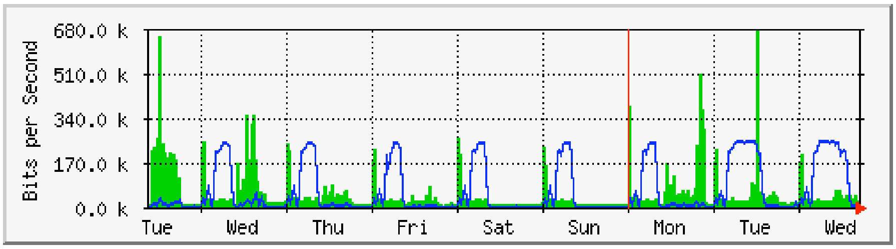
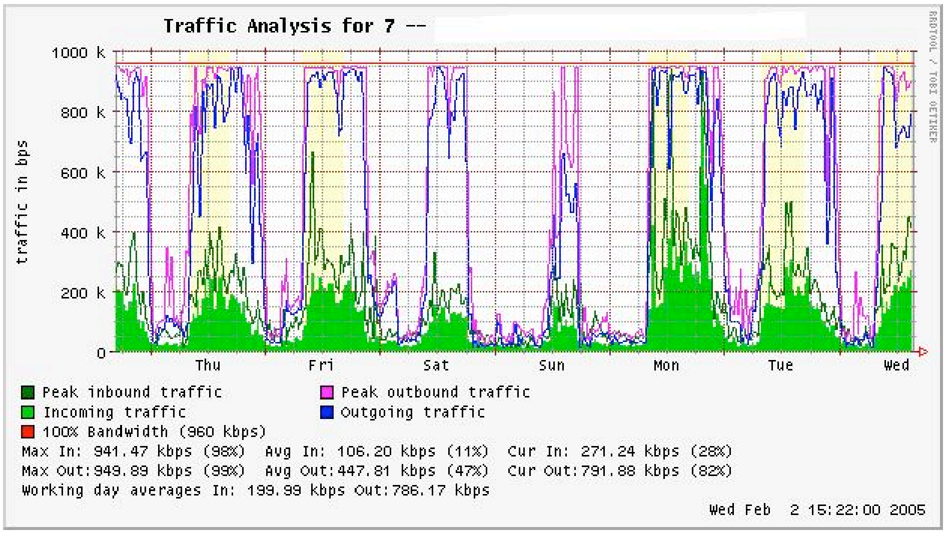


Figure 3.36: The classic network flow graph. The dark area represents inbound traffic, while the line represents outbound traffic. The repeating arcs of outbound traffic show when the nightly backups have run.

Traffic patterns will vary with what you are monitoring. A router will normally show more incoming traffic than outgoing traffic as users download data from the Internet. An excess of outbound bandwidth that is not transmitted by your network servers may indicate a peer-to-peer client, unauthorised server, or even a virus on one or more of your clients. There are no set metrics that indicate what outgoing traffic to incoming traffic should look like. It is up to you to establish a baseline to understand what normal network traffic patterns look like on your network.

Detecting network overload

Figure 3.37 shows traffic on an overloaded Internet connection.



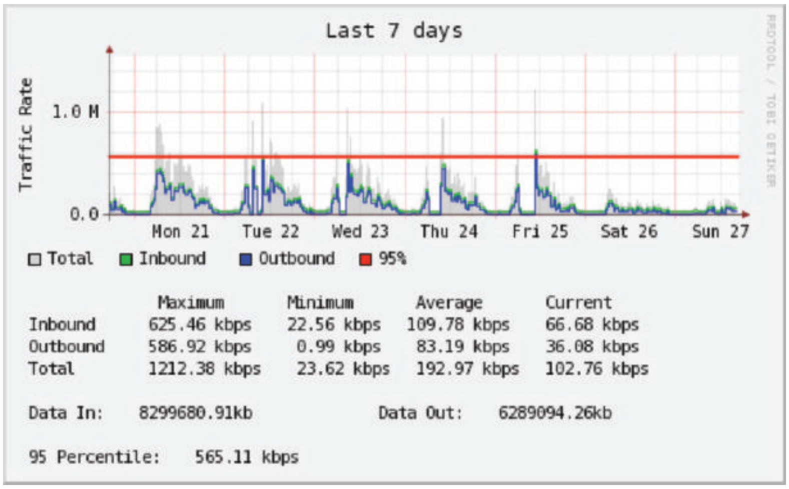
```
Max In: 941.47 kbps (98%)  Avg In: 106.20 kbps (11%)  Cur In: 271.24 kbps (28%)
Max Out:949.89 kbps (99%)  Avg Out:447.81 kbps (47%)  Cur Out:791.88 kbps (82%)
Working day averages In: 199.99 kbps Out:786.17 kbps
                                                      Wed Feb  2 15:22:00 2005
```

Figure 3.37: Flat-topped graphs indicate that a line is using the maximum available bandwidth, and is overutilised during these times.

The most apparent sign of overloading is the flat tops on outbound traffic during the middle of every day. Flat tops may indicate overloading, even if they are well below the maximum theoretical capacity of the link. In this case it may indicate that you are not getting as much bandwidth from your service provider as you expect.

Measuring 95th percentile

The 95th percentile is a widely used mathematical calculation to evaluate regular and sustained utilisation of a network pipe. Its value shows the highest consumption of traffic for a given period. Calculating the 95th percentile means that 95% of the time the usage is below a certain amount, and 5% of the time usage is above that amount. The 95th percentile is a good value to use to show the bandwidth that is actually used at least 95% of the time.



```
☐ Total   🟩 Inbound   🟦 Outbound   🟥 95%

            Maximum        Minimum       Average       Current
Inbound     625.46 kbps    22.56 kbps    109.78 kbps   66.68 kbps
Outbound    586.92 kbps     0.99 kbps     83.19 kbps   36.08 kbps
Total       1212.38 kbps   23.62 kbps    192.97 kbps   102.76 kbps

Data In:    8299680.91kb               Data Out:    6289094.26kb

95 Percentile:    565.11 kbps
```

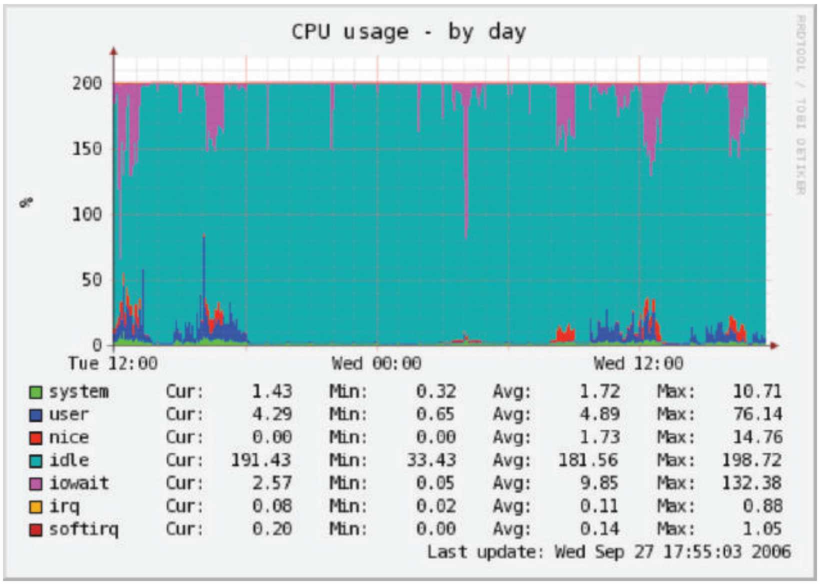
Figure 3.38: The horizontal line shows the 95th percentile amount.

MRTG and Cacti will calculate the 95th Percentile for you. This is a sample graph of a 960 Kbps connection. The 95th percentile came to 945 Kbps after discarding the highest 5% of traffic.

Monitoring RAM and CPU usage

By definition, servers provide critical services that should always be available. Servers receive and respond to client machine requests, providing access to services that are the whole point of having a network in the first place. Therefore, servers must have sufficient hardware capabilities to accommodate the work load. This means they must have adequate RAM, storage, and processing power to accommodate the number of client requests. Otherwise, the server will take longer to respond, or in the worst case, may be incapable of responding at all. Since hardware resources are finite, it is important to keep track of how system resources are being used. If a core server (such as a proxy server or email server) is overwhelmed by requests, access times become slow. This is often perceived by users as a network problem.

There are several programs that can be used to monitor resources on a server. The simplest method on a Windows machine is to access the Task Manager using the **Ctrl Alt + Del** keys, and then click on the Performance tab. On a Linux or BSD box, you can type **`top`** in a terminal window. To keep historical logs of such performance, MRTG or RRDtool (page **83**) can also be used.



```
🟩 system    Cur:    1.43  Min:   0.32  Avg:   1.72  Max:  10.71
🟦 user      Cur:    4.29  Min:   0.65  Avg:   4.89  Max:  76.14
🟥 nice      Cur:    0.00  Min:   0.00  Avg:   1.73  Max:  14.76
🟦 idle      Cur:  191.43  Min:  33.43  Avg: 181.56  Max: 198.72
🟪 iowait    Cur:    2.57  Min:   0.05  Avg:   9.85  Max: 132.38
🟧 irq       Cur:    0.08  Min:   0.02  Avg:   0.11  Max:   0.88
🟥 softirq   Cur:    0.20  Min:   0.00  Avg:   0.14  Max:   1.05
                         Last update: Wed Sep 27 17:55:03 2006
```

Figure 3.39: RRDtool can show arbitrary data, such as memory and CPU usage, expressed as an average over time.

Mail servers require adequate space, as some people may prefer to leave their email messages on the server for long periods of time. The messages can accumulate and fill the hard disk, especially if quotas are not in use. If the disk or partition used for mail storage fills up, the mail server cannot receive mail. If that disk is also used by the system, all kinds of system problems may occur as the operating system runs out of swap space and temporary storage.

File servers need to be monitored, even if they have large disks. Users will find a way to fill any size disk more quickly than you might think. Disk usage can be enforced through the use of quotas, or by simply monitoring usage and telling people when they are using too much. Nagios (page **88**) can notify you when disk usage, CPU utilisation, or other system resources cross a critical threshold.

If a machine becomes unresponsive or slow, and measurements show that a system resource is being heavily used, this may be an indication that an upgrade is required. If processor usage constantly exceeds 60% of the total, it may be time to upgrade the processor. Slow speeds could also be as a result of insufficient RAM. Be sure to check the overall usage of CPU, RAM, and disk space before deciding to upgrade a particular component.

A simple way to check whether a machine has insufficient RAM is to look at the hard disk light. When the light is on constantly, it usually means that the machine is constantly swapping large amounts of data to and from the disk. This is known as ***thrashing***, and is extremely bad for performance. It can usually be fixed by investigating which process is using the most RAM, and killing or re-configuring that process. Failing that, the system needs more RAM.

You should always determine whether it is more cost effective to upgrade an individual component or purchase a whole new machine. Some computers are difficult or impossible to upgrade, and it often costs more to replace individual components than to replace the entire system. Since the availability of parts and systems varies widely around the world, be sure to weigh the cost of parts vs. whole systems, including shipping and taxes, when determining the cost of upgrading.

Resources

- JANET Bandwidth Management Review, *http://www.ja.net/services/network-services/bmas/papers/review/BMAS_Bandwidth_Management_Review.htm*
- Linux Advanced Routing and Traffic Control HOWTO, *http://lartc.org/*
- Linux security and admin software, *http://www.linux.org/apps/all/Networking/Security_/_Admin.html*
- Network monitoring implementation guides and tutorials, *http://wiki.debian.org/Network_Monitoring*
- Optimising Internet Bandwidth in Developing Country Higher Education, *http://www.inasp.info/pubs/bandwidth/index.html*
- Planet Malaysia blog on bandwidth management, *http://planetmy.com/blog/?p=148*
- Rusty Russell's Linux Networking Concepts, *http://www.netfilter.org/documentation/HOWTO/networking-concepts-HOWTO.html*
- *TCP/IP Illustrated, Volume 1: The Protocols*, Addison Wesley, 1994
- Wireless Networking in the Developing World, *http://wndw.net/*

4
Implementation

At this point, you should have a clear operational policy that governs how your users may make use of network services. You should also have good network monitoring tools installed and collecting data, so you can tell at a glance precisely how your network is actually being used. With these major components in place, you are now ready to make changes to the network configuration that bring actual utilisation in line with policy. The ***implementation phase*** of network building closes the feedback loop that allows policy to be consulted upon and revised, and network services to be implemented and changed, based on information provided by the monitoring systems.

This chapter will show you the essential technical components that should be in place in virtually every network connected to the Internet. These techniques will allow you to limit, prioritise, and optimise the flow of information between the Internet and your users.

While these technical constraints are necessary to maintain the health of the network, there is one management technique that is often overlooked, yet nearly always makes the greatest impact on network utilisation: **communication with your users**. If your users don't understand that their actions directly impact the performance of the network, how can they be expected to know that the network is overutilised and not "just broken?" When users are frustrated with network performance, and met with indifference (or outright contempt) by network administrators, they tend to try to find ways around the rules in order to "get their work done." More often than not, these workarounds will consume even more bandwidth, causing problems for users and administrators alike. For example, an enterprising user might discover that they can bypass a slow and badly configured network proxy by using an anonymising proxy server found somewhere on the Internet. While this may show improvement in performance for the user, that traffic is not being cached locally, so bandwidth is wasted. As news of this "fix" spreads among the users, even more bandwidth is wasted, making the entire network slower for everyone. If the network administrators had been monitoring the performance of the proxy server, or listening to user complaints, then the problem could have been addressed much earlier.

Of course, the vast majority of users are not malicious. Often they simply do not understand the repercussions of their actions, and may be completely caught off-guard when they realise that they are consuming significantly more bandwidth than other users. For example, when network administrators at Carnegie Mellon approached one user about excessive traffic coming from their computer, their response was: "I don't understand--what's bandwidth? How do I reduce it? What am I doing wrong? What's going on?!" (Read the full story in the **Case Studies** chapter, page **235**).

While technical solutions are indeed important, they must go hand-in-hand with education and a responsive network management team. No technical solution can help a network if all of the users insist on consuming as much bandwidth as possible without regard for others. Likewise, it is impossible to effectively manage a network without communicating with your users. You can only provide the best possible service to your users by understanding their needs. The best way to understand those needs, and to make your users appreciate the reality of limited resources, is through clear and honest communication.

The importance of user education

The actions of your users will ultimately determine your bandwidth utilisation. Therefore, your users should be informed of the rights and obligations that govern network use. Having an operational policy does little good if your users are not informed about its scope or details. While this could mean sending out blanket information that is targeted at all users, you may find that you have better results when personally contacting individual users.

But how can you realistically expect to personally speak with every user on a 1000+ node network? Fortunately, it is rarely necessary to give **every** user a personal interview. You can do wonders by simply starting with the biggest "bandwidth hogs" and proceed from there.

The 5/50 rule

More often than not, the majority of users make light use of network resources. They occasionally check their email and browse a few web pages without making excessive use of the network. However, there are always a few users who will use all available bandwidth, both inbound and outbound, by running servers, peer-to-peer file sharing programs, video streams, and other bandwidth intensive applications. Those are the circumstances where the 5/50 rule comes into play. By focusing on the 5% of users who typically consume 50% of the bandwidth, you can make a huge impact on network performance with less effort.

Targeting the top 5% means identifying and contacting people in order to inform them of their high utilisation. Unless you have a clear idea of who is using your network, there is no way to identify the source of the problem (see chapter three: **Monitoring & Analysis** for ways to reliably identify individual users). Once you have identified problem users, you should inform them that their usage deviates from that which is stated as acceptable by the network access policy.

It may become apparent that such users have legitimate needs differing from those of the typical user, in which case you could make arrangements to specifically support them. Often, users have no idea that they are causing problems (particularly if their computer has been infected by a virus, or if it is running a peer-to-peer file sharing application). Of course, a technical solution can solve the problem by simply denying access to the user once utilisation exceeds a particular threshold. But it is more likely that informing the user of the problem, and showing them how to fix it, will lead that user to take greater responsibility for their own network activity in the future.

Providing feedback to users about network load

If users know there is network congestion, they tend to cut down on their usage, which in turn makes the network experience better for everyone. Without that feedback, users tend to think that the network is "broken" and may inadvertently increase the network load. A classic example of this can be seen when a user becomes impatient with a slowly loading web page and will repeatedly click the reload key. This action submits more network requests, so that the network moves even more slowly, causing the user to continue clicking the reload key, ensuring that the vicious cycle continues. That is, it continues until the user gives up in disgust and complains that the network is "broken."

Here are a few methods you can use to communicate the state of the network to your users.

Public utilisation graphs

One obvious method for showing network congestion to your users is to simply publish your utilisation graphs on a web server. Even when the Internet connection is experiencing heavy use, the monitoring server will be available since it resides on the local network. This will show one and all that the lag is caused by too much network access, and not by a misconfiguration of the servers. By including individual user data, as well as aggregated throughput statistics, you can encourage people to remind their peers that bandwidth is limited. Some sites even post a "top ten" or "top 100" list of excessive bandwidth users, aggregated daily or weekly. If your name appears on that list, you can be sure that your friends and colleagues will have a few words for you. This method can be even more effective than receiving a phone call from the network administrator.

You can set the utilisation graphs as the default home page (or perhaps make it a single click away) for computers installed in public labs. You might even set up a computer with a continuous display of the current network utilisation and post it in a highly visible place. This helps to serve as a constant reminder for users that their actions have a direct impact on their colleagues' ability to access Internet resources.

Using a captive portal

A ***captive portal*** allows you to "capture" a user's browsing session and redirect them to a web page where they may be asked to perform some task. Wireless hotspots often use captive portals to capture the client's session and redirect them to a site where they can enter their credit card details or other credentials.

A real world example of effective use of a captive portal is the implementation at the University of KwaZulu-Natal. All users at the institution were being authenticated to several Squid proxy servers. The Squid log files were then in turn being imported into a MySQL database. This resulted in a comprehensive database that could be interrogated to derive statistics. Queries run against the database showed that during work hours, up to 20% of the bandwidth was being used by just 20 users out of a total of roughly 12000. That is, 0.2% of the user base was consuming 20% of the bandwidth for the entire university! In addition, the majority of the sites were clearly not of academic content. The university had a policy about bandwidth usage, however its users were either ignoring the policy or simply had not read it. A decision was made that the top 20 users needed to be shown the policy, and if they continued to ignore it that action would be taken against them.

A captive portal mechanism was implemented to notify the top 20 users automatically. Squid has a feature called redirection, which enables you to redirect a user to a web page if they match a specific Squid ***Access Control List*** (***ACL***). Every night a script was run that compiled a list of the top 20 users. These users were then added to a special ACL contained in a text file on the Squid servers. When the user tried to browse the Internet the next morning, they were redirected to a PHP based web page that highlighted the relevant sections of the policy. When they clicked on the OK, button the script would remove them from the ACL and they could continue browsing. If a user was shown this message more than twice in a 14 day period, they were then disabled from browsing the Internet during work hours for a week.

The effect of this over the course of the first month was very interesting. The top 20 users reduced their work hours bandwidth usage from 16-20% to around 6%, but after hours usage increased. Clearly a lot of the "abusers" had now moved their browsing habits to after hours where there was less policing. Did the condition of the Internet line change? No it did not, but what did change is the the number of users and legitimate sites visited. These both showed an increase, which indicated that the latent demand for bandwidth had absorbed the bandwidth freed up by the abusive top users.

If you are keeping detailed logs of network usage, you can interrupt bandwidth hogs before they cause real problems. By using the redirect functionality of Squid (page **184**), it is simple to redirect "errant users" to a page where they are reminded of, and must to agree to, the network access policy. Once they have done this, Squid then allows them to continue browsing. To do this, you will need to set up the following:

- Configure Squid to authenticate and log traffic per user (page **186**)
- Each night, compile list of top users. If any individual user exceeds the "excessive" usage threshold, add this user's name to a Squid ***Access Control List*** (***ACL***) (page **269**).
- The redirect function in Squid will match the ACL the next time the user browses the web, and the redirect page will be shown instead of the requested page.
- When the user clicks the "I Agree" button, they are removed from the ACL and can then browse normally.
- Traffic continues to be logged, and the process begins again.

This is just one example of a creative technical solution that, when combined with social reinforcement, can change users' behaviour.

General good practices

There are several techniques that your users can implement on their own to help keep bandwidth utilisation to a minimum. While there is never a guarantee that users will completely comply with the techniques in this list, making them aware of these techniques will empower them to start making bandwidth management part of their daily routine. These techniques of ***network etiquette*** aren't really rules, so much as guidelines for being a good net neighbor.

Optimise your web browser

Every web browser includes options that will limit bandwidth usage. Some of these options include:

1. **Disable bandwidth-hungry plugins like Java and Flash.** Unless a particular site requires Java for access, it can simply be disabled. Since the vast majority of Java and Flash applications are simple animations, games, and videos, there is rarely any need to download them except for entertainment purposes. Use the HTML version of sites that include both HTML and Flash options. Note that many sites require the use of JavaScript, which is significantly smaller and faster than Java, and can usually be enabled without any noticeable speed penalty.

2. **Disable automatic updates.** While keeping your browser software up-to-date is vitally important from a security point of view, updating in an ad-hoc and automatic way may waste significant bandwidth at peak times. While this can be sped up considerably using a good caching proxy (page **135**) or a local software update repository (page **144**), simply disabling automatic updates immediately reduces background bandwidth usage. You must remember to manually update the software when the network is not busy in order to apply security patches and feature updates.

3. **Increase the size of the local cache.** If there are sufficient resources on the local machine, increase the browser size. More is generally better, but a cache of several hundred megabytes is usually reasonable.

4. **Disable pop-ups.** Pop-up windows are nearly always unwanted advertisements containing large images or flash movies, and will automatically consume significant amounts of unintentionally requested bandwidth. Pop-ups can be disabled in all modern browsers. Well-coded sites that require pop-up windows for functionality will still work, and the user can always allow pop-ups on a case by case basis.

5. **Use ad blocking software.** By blocking ads before you download them, you can save bandwidth and reduce user frustration. Free and commercial ad blocking packages are available for every browser. For Mozilla Firefox, try AdBlock Plus: *https://addons.mozilla.org/firefox/1865/*

6. **Install anti-spyware tools.** Malicious sites may install spyware programs that consume bandwidth and introduce security problems. These attacks nearly always come through the web browser. Using a spyware detection and removal tool (such as Lavasoft AdAware) will keep these problems to a minimum. *http://www.lavasoftusa.com/software/adaware/*

7. **Disable images.** For many kinds of online work, graphical images may not be required. Since graphic files are considerably larger than HTML code, disabling the display of images (even temporarily) can significantly improve response time and reduce bandwidth use. If possible, configure your browser to only display graphics when explicitly requested.

8. **Use Mozilla Firefox instead of Internet Explorer.** Although it is the default browser on most Windows boxes, IE is a notorious attack point for spyware and viruses, and is widely considered obsolete since the release

of Mozilla Firefox (*http://www.mozilla.com/firefox/*). Since it is an open source project, Mozilla Firefox has a very large and flexible set of extensions that allow you to configure and optimise just about every aspect of how the browser works. One popular extension is FireTune (*http://www.totalidea.com/content/firetune/firetune-index.html*), which groups many common optimisation options into a simple, easy to understand menu. Other extensions provide excellent content filtering, presentation, and download optimisation features.

Of course, the most effective bandwidth optimisation tool is simply refraining from requesting information that you don't need. Ask yourself if it's really appropriate to try to stream videos from YouTube during peak network times (even if the video is really funny...). The more you request from a busy network, the longer everyone will have to wait for their requests to be filled. Be considerate of your fellow network users, and your network will be healthier and faster.

Optimise your email

Web browsing and email are the most commonly used services on the Internet. Just as you can optimise your web browser to minimise bandwidth usage, there are many steps you can take as a user to make your email work better too.

1. **Don't use web mail.** Sites such as Hotmail, Gmail, and Yahoo! Mail use significantly more bandwidth than do traditional email services. With graphics, advertisements, and an HTML interface, an individual email may represent thousands of times the number of bytes of the equivalent text email. If your network provides email services (via Mozilla Thunderbird, MS Outlook, or another email client), then use them. If email service is not provided for you, ask your network administrator if it makes sense to set it up for your organisation.

   Some web mail services, such as Gmail and Fastmail, allow access via POP3 and IMAP from a standard mail program. This is much faster and more efficient than using their web interfaces.

2. **Send emails in plain text, not HTML.** HTML emails are bigger than their equivalents in plain text, so it is preferable to send plain text emails to reduce bandwidth requirements. Most email clients let you set plain text as the default format, and/or on a per email basis. As well as reducing the amount of bandwidth needed, you'll also benefit from your email being less likely to be treated as spam by the recipient.

3. **Limit the use of attachments.** While it is possible to send colossally huge files through email, the protocol wasn't really designed for this use. Instead of sending a large file to a group of people, post the file to a web server

and send the link instead. This will not only speed up your own email, but it will save significant bandwidth for everyone receiving the message. If you **must** include an attachment, make it as small as possible. You can use a compression utility to reduce the size of the file (WinZip is one popular commercial tool, but many free alternatives exist. One list of WinZip alternatives is *http://free-backup.info/free-winzip.html*). If the attachment is a photograph, reduce the image size to something reasonable, such as 1024x768 or 800x600. This size is usually fine for on-screen viewing, and is significantly smaller than a raw 3+ Megapixel image.

In addition, websites exist which allow files to be uploaded and included in the email as a link. This allows recipients to choose whether or not to download the file. One such service is Dropload, *http://www.dropload.com/*.

4. **Filter your email on the server, not the client.** If your network admin provides email services, they should also provide spam and virus filtering. While it is possible to do this kind of filtering within your mail client, it is far more efficient to do it on the server side (since junk messages can be discarded before they are downloaded). If your network offers filtering services, be sure these are enabled on the server.
5. **Use IMAP instead of POP.** The IMAP protocol makes much more efficient use of network bandwidth by using a caching system that only downloads messages when needed. The POP protocol downloads all email messages as they are received, whether they are opened or not. If possible, use IMAP instead of POP. Your network administrator should be able to tell you if an IMAP server is available. This is especially critical when accessing your email over a low-bandwidth link.
6. **Don't send spam.** Junk email (commonly referred to as ***spam***) comes in many guises. While we are all familiar with the "get rich quick" and "improve your love life" variety, other kinds are more subtle. One example of this is the so-called "boycott" email, where someone sends you a petition for a popular cause (such as lowering gasoline prices or changing an unjust law). You are instructed to "sign" the petition with your email address, and send it to six of your friends. These messages are completely fabricated by spammers who then use the collected email addresses as targets for more spam. These kinds of scams are easily avoided if you ask yourself, "what will really happen if I send this message with my email address to total strangers?" The answer: wasted bandwidth, with the promise of more wasted bandwidth to follow.

Use of collaborative tools vs. Word attachments

One popular phenomenon among inexperienced users is the proliferation of Microsoft Word file attachments in normal email. While programs like Word and OpenOffice are excellent choices for desktop publishing, they are nearly always a waste of bandwidth and effort when used for normal communications.

Consider the case of a group of ten people collaborating on a research project. If the group uses a large cc: list of email addresses, and includes Word attachments in the development process, they are inevitably going to run into several problems:

- Each message now takes up ten, one hundred, or even thousands of times the storage space of a simple email. This makes everyone's email take longer to upload and download, and wastes storage space on the server and everyone's personal computer.
- The work flow of each user is interrupted as they must now download the attachment and open it in a separate program. They must download every attachment sent to the group, even if they do not need it for their work.
- Every user needs to have the same version of the program installed (in this case, Word) or they will not be able to read the message. By contrast, plain email can be read by hundreds of different email clients.
- Maintaining a consistent set of revisions is difficult or impossible, since users tend to work asynchronously. Some may reply with revisions in email, while others may respond with new Word documents. Someone on the team will need to make sense of all of these revisions and maintain a single authoritative copy of the work. This problem is known in project management circles as ***versionitis***.
- If the list of team members is maintained by hand, it is easy for people to be accidentally (or intentionally) removed from the list, resulting in general confusion.
- If any user has problems with their own computer (e.g., it is lost, damaged, stolen, or infected by viruses), then any work they have not yet sent to the list is permanently lost.

There is simply no need to indulge in these bad habits! You can be more effective and save bandwidth by using collaborative tools designed to make group projects easy. Most organisations provide mailing list services free to their users. This makes management of the group discussion much simpler than trying to maintain a list by hand. Mailing list software will often provide restrictions on the type and size of file attachments, discouraging this bad habit. If the discussion group is very large, web-based forum software can be implemented that will allow any number of participants.

For project development, one very popular collaborative tool is the ***wiki***. A wiki is a web site that allows any user to edit the contents of any page. The wiki can be protected by a simple password, an access control list (based on IP address or other identifying information), or simply left open for public access. All changes to the wiki are recorded, allowing users to view all revisions made to the work. This effectively cures the revision control problem, provides excellent backups for the work, and makes efficient use of available bandwidth. Attachments (such as graphics, slideshows, and even Word documents) can be uploaded to any page and downloaded on demand. The resulting work can be easily searched and linked to, just as with any web page. Users can be notified of changes to the wiki via email or using ***Really Simple Syndication*** (***RSS***).

This book was developed using a wiki in conjunction with a mailing list. Trying to manage a team of a dozen people scattered all over the world by using attachments in email would have been nearly impossible, and would certainly have been a waste of bandwidth.

If you are a user, ask your network administrator what tools are available for collaborative work. If you are a network admin, see the list of resources at the end of this chapter for many freely available software packages you can install on your own network.

One last word about attachments

Consider what happens when you send a file attachment to a large group of people. The flow of information may look something like Figure 4.1:

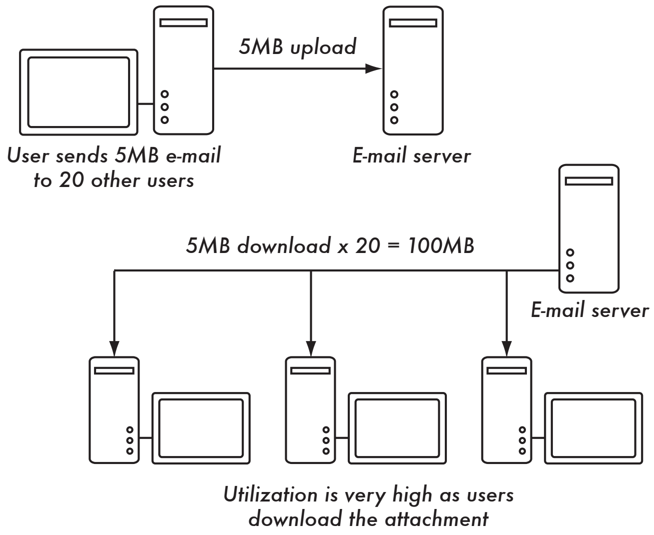

5MB upload
User sends 5MB e-mail to 20 other users
E-mail server
5MB download x 20 = 100MB
E-mail server
Utilization is very high as users download the attachment

Figure 4.1: A single unnecessary email attachment can quickly add up to hundreds of megabytes of wasted bandwidth as users forward it to each other.

A single 5 Megabyte attachment can easily blossom to several hundred megabytes of wasted information as it is forwarded between people on the network. If the original sender had sent a hyperlink instead of the original file, considerable bandwidth would have been saved, while people could still easily view the file when needed. Your users should consider this the next time they want to send an attachment to their friends or colleagues.

Download managers & peer-to-peer clients

The word "client" is not exactly accurate when applied to ***peer-to-peer*** programs (such as BitTorrent, Gnutella, KaZaA, and eDonkey2000). These programs work by turning every computer into both a client and a server, where users exchange information directly with each other. While this can be a tremendous boon to publishers who use these protocols to avoid high bandwidth costs, they can be a nightmare for network administrators. These programs often maximise both inbound and outbound bandwidth, reducing the performance of other network services to a crawl. They will often aggressively attempt to circumvent firewall restrictions, and can even disguise some of their protocol information in order to avoid filtering. These programs continue to utilise outbound bandwidth even when the user has finished downloading, serving copies of the retrieved data to the Internet at large.

So-called ***download managers*** can include peer-to-peer technology (sometimes referred to as ***swarming***) in order to "speed up" user downloads. Examples of some of these types of download managers include FlashGet, GetRight, Download Accelerator Plus, LimeWire Download Manager, DownloadThemAll (DTA), and GetBot.

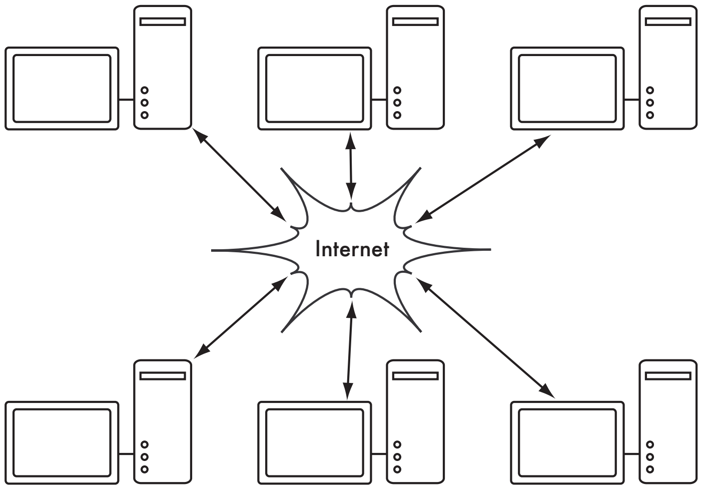


Figure 4.2: Peer-to-peer programs work by turning every node into a server that uploads data to the others. This can be a boon on very fast and cheap networks, but it is disastrous on slow and expensive connections such as VSAT.

The lure of these programs can be great for your users, since they promise fast downloads of music, movies, and other data. Your operational policy should be very clear about the use of peer-to-peer applications and download managers, with regard to both legitimate and potentially infringing uses.

Users are often unaware that these programs introduce a heavy toll on a network. They may not realise that their download manager is consuming resources even when they are not using it. While many of these services can be blocked or limited at the firewall, the best policy is simply not to use them on low-bandwidth links. If they must be used, then use them to schedule large downloads when the network is less busy, which can improve network performance for everyone. Make sure your users understand the impact that these programs will have on the network.

Essential services

Now that your users have a clear understanding of your usage policy, as well as an idea of general good practices, you can implement hard restrictions on the flow of network traffic. The primary method for making your Internet connection more responsive is to ensure that network usage never quite reaches the available capacity. However large or complex your network installation, you will likely make use of a few essential kinds of technology to ensure that your Internet connection is used in an appropriate and efficient manner.

The most basic class of tool in the bandwidth management toolkit is the ***firewall***. A firewall can be thought of as a filter for network traffic. By making good choices in your firewall configuration, you can allow legitimate traffic through, and drop traffic destined for inappropriate services before it is transmitted to your ISP. This allows the available bandwidth to be used exclusively for its intended purpose. For example, by blocking access to peer-to-peer services, considerable bandwidth can be saved, making email and the web servers more responsive.

Firewalls can also add considerable security to your network by preventing access to machines on your private network from the Internet. It is generally accepted as standard practice to have at least one firewall in place if you have a connection to the Internet.

Caches exist everywhere in the computing field. The term cache means literally, to store. Most people are familiar with the web browser cache, which keeps a local copy of content retrieved from the Internet. This local copy can be retrieved and reused faster than making the same request from the Internet. By using a cache, images and web pages can be displayed much more quickly without making use of the Internet connection. Of course, information stored in the cache may or may not be used again, so caches are only beneficial when the cost of storing the information is less than the cost of retrieving the information again. This cost is measured both in system resources (disk space and RAM) and in time.

In addition to the cache available in your web browser, caching services can be set up to be used by an entire network. Using a ***site-wide web cache*** can save considerable bandwidth, since one user's visit to a popular site (such as *www.yahoo.com*) will cause a local copy to be saved and automatically served to other users who visit the same site. Other network services, such as DNS lookups, may also be cached, considerably improving the "feel" of a connection while saving Internet bandwidth for other uses.

Caches are typically implemented through use of a ***proxy server***. This is a kind of buffer or middleman between a computer and the Internet resources it requests. A client makes a request of the proxy server, which then contacts the web server (or other server) on its behalf. The response is sent back to the client as if it had come from the original server. A ***socks proxy*** is a typical example of this kind of server. Proxies can provide a controlled and secure way to access resources that lie outside your firewall by requiring authentication from the client. While proxies typically also include caching services, this is not required. You may wish to make use of a proxy server to provide access control and an audit trail to monitor your users' usage patterns.

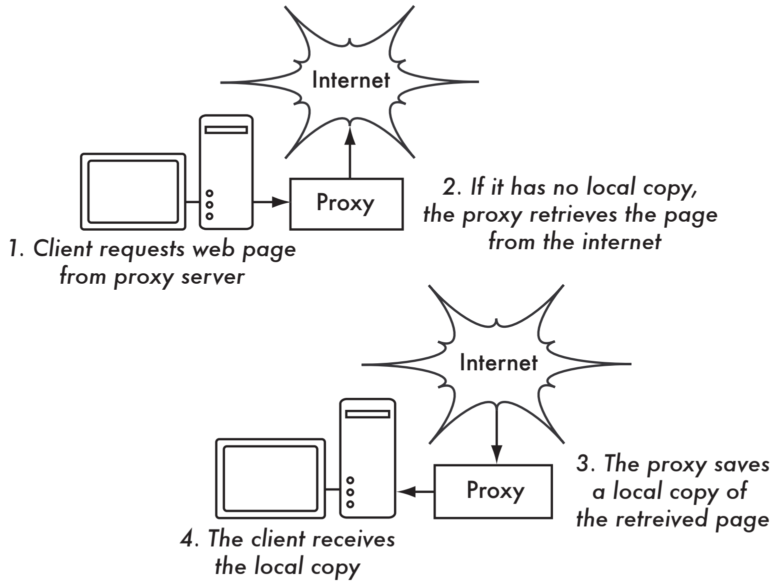

Internet
Proxy
2. If it has no local copy, the proxy retrieves the page from the internet
1. Client requests web page from proxy server
Internet
Proxy
3. The proxy saves a local copy of the retreived page
4. The client receives the local copy

Figure 4.3: Proxy servers make requests on behalf of a client program. They may also implement authentication or caching.

Mirrors can be thought of as a kind of manually updated caching server. Whole copies of popular websites and data stores can be stored locally, and updated when network utilisation is low (after working hours, or in the middle of the night). Users can then use the local mirror rather than requesting information directly from the Internet. This can save considerable bandwidth, and is much more responsive from a user's point of view, particularly at peak utilisation times. However, users must be aware of the existence of the mirror, and know when to use it, otherwise it will waste bandwidth by downloading information that is never used.

Email can become one of the most heavily abused services on a network, even though legitimate email service itself may use relatively little bandwidth. Unless your email servers are properly configured, they may allow unauthenticated users to send messages to any destination. This is referred to as an ***open relay***, and such servers are often abused by spammers to send huge numbers of messages to the Internet at large. In addition, ***spam*** and ***viruses*** can clog your legitimate email services unless you employ an effective form of ***content filtering***. If your mail service is slow, or unreasonably difficult to use, then your users might turn to ***webmail*** services (such as Hotmail, Yahoo mail, or Gmail) and waste even more bandwidth. If you offer email services to your users, then these services must be properly configured to make the best possible use of your Internet connection. Spam and viruses should be filtered before they cross your Internet line (page **174**), open relays should be closed (page **166**), and web mail services should be avoided.

By implementing these essential services (firewall, caching, mirrors, and proper email configuration) you will make a significant impact on your bandwidth utilisation. These basic services should be considered mandatory for any network connected to the Internet. For more advanced topics (such as bandwidth shaping, fairness queueing, and protocol tweaks) see chapter six, **Performance Tuning**.

Firewall

The word ***firewall*** refers to a physical barrier in a building or vehicle designed to limit damage in the event of a fire. It prevents fire on one side of the wall from spreading to the other. In a car, the firewall is generally a solid plate that seals off the fuel tank or engine compartment from the passenger compartment. In buildings, the firewall may be made of concrete or metal, and it seals off different sections of the building. This provides protection from fire and possibly the total collapse of the structure.

The logical network firewall functions in a similar manner. Although, instead of protecting your network against fire, it protects against undesirable traffic. For example, it may deny access to peer-to-peer file sharing services on the Internet, or to prevent unauthorised connections to servers inside your organisation. The firewall can filter both inbound and outbound traffic. In order to do this most effectively, the firewall needs to be located at the border where your network meets the Internet.

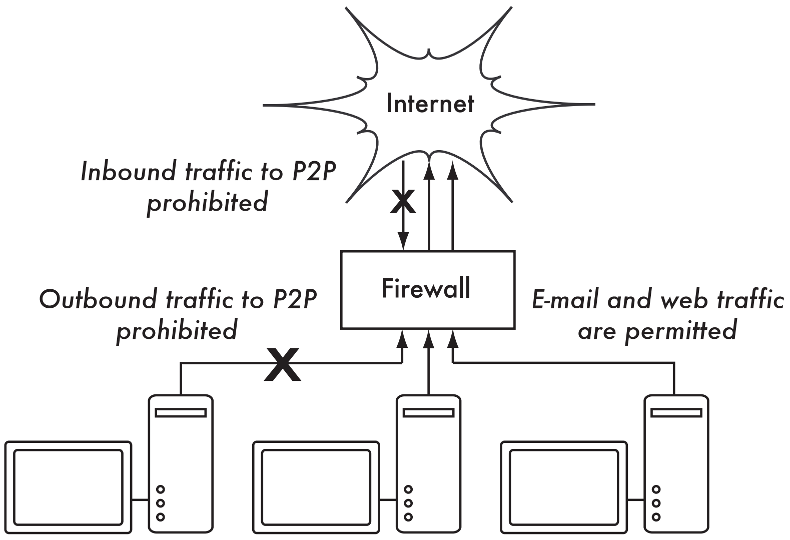

Internet
Inbound traffic to P2P prohibited
Firewall
Outbound traffic to P2P prohibited
E-mail and web traffic are permitted

Figure 4.4: Firewalls are most effective at the border between your network and the external world (e.g. the Internet).

Users may also choose to implement their own ***personal firewall***. This provides a "last line of defense" for an individual computer by blocking access to all (or most) services. Linux, BSD, Windows XP, and Mac OS X all have built-in firewall support. There are also a number of third party firewall packages available for Windows; ***ZoneAlarm*** and ***Norton Firewall*** are two popular commercial packages.

A personal firewall is a good idea if properly implemented, but for most users it may seem like an inconvenience because some services (such as Voice-over-IP) will not work properly unless the software is configured to allow such access. Since users often disable their firewall when they are having trouble, it is not a good idea to rely exclusively on personal firewalls to protect your network.

Firewalls can make filtering decisions based on any of the network layers from two and above (see chapter three: **Monitoring & Analysis** for a discussion of layered network models) but traditionally are grouped into two classes. ***Packet filters*** operate at the Internet layer by inspecting source and destination IP addresses, port numbers, and protocols.

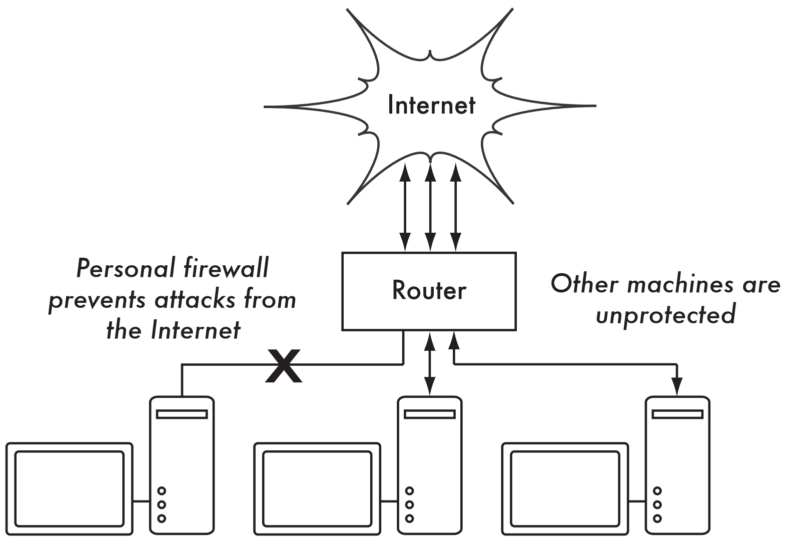

Internet
Router
Personal firewall prevents attacks from the Internet
Other machines are unprotected

Figure 4.5: Personal firewalls can provide a small measure of protection, but should not be relied upon to protect a large organisation.

Application firewalls operate at the top layer, and make filtering decisions based on the application protocol being spoken. They tend to be more flexible than simple packet filters, but they tend to require more in the way of system resources. A packet filter may block all communications on port 80, while an application firewall can block HTTP traffic on any port. One example of an application firewall is L7-filter, *http://l7-filter.sourceforge.net/*.

Access philosophy

There is a very old joke floating around the Internet that goes something like this:

In France, everything is permitted, except what is explicitly forbidden.
In Germany, everything is forbidden, except what is explicitly permitted.
In Russia, everything is forbidden, including what is explicitly permitted.
In Italy, everything is permitted, ***especially*** *what is explicitly forbidden.*

When building firewalls, you may choose to implement either of the first two models as a general policy: the "French" model (everything that is not expressly forbidden is permitted) or the "German" model (everything that is not expressly permitted is forbidden). While the first approach may seem easier from a network administrator's point of view, it is far less secure, and can be more difficult to maintain over the long term. It is much safer to err on the side of denying traffic first, and make exceptions for legitimate traffic as the need arises. If you are already monitoring your network extensively (as detailed in chapter three), then you should have a good idea of which services your users need to access.

For a solid network firewall, you will need to implement the following four rules as a standard policy:

1. Allow already established and related connection traffic.
2. Allow TCP/IP SYN packets to the services you wish to permit.
3. Allow UDP packets to the services you wish to permit.
4. Deny ALL other traffic, and optionally log denied traffic to disk.

This configuration works well for the vast majority of networks connected to the Internet. If your organisation requires support for other protocols as well (such as GRE, which is required for VPN services such as PPTP), you can add those exceptions just before step four. You may consider logging denied traffic to disk in order to debug problems and detect attempts to circumvent your firewall. But this can quickly fill up your disk on a very busy network or in the event of a denial of service attack. Depending on your circumstances, you may wish to only enable logging when you need to debug firewall problems.

Here are some examples of how to set up a good default firewall in Linux and BSD.

Building a firewall in Linux

The firewall implemented in modern Linux kernels is called ***netfilter***. Netfilter is extremely powerful, flexible and complex, and a whole book could easily be written on netfilter alone. We will only cover the very basics here. Full documentation is available at *http://www.netfilter.org/*.

Netfilter consists of kernel code that filters packets, and userspace programs to control the kernel code. The interface that most people are familiar with is the **`iptables`** command, which allows you to list and change the firewall rules from the command line.

Netfilter rules are divided into sections called ***tables***. The default table is the ***filter*** table, but there are additional tables used for Network Address Translation and other purposes. Each table contains a number of processing phases, called ***chains***. Each chain in turn contains ***rules*** that determine the fate of any packet that enters the chain. Each table and chain is used during a different phase of the filtering process, allowing for very flexible packet matching.

There are three chains defined by the **filter** table:

- The **INPUT** chain is used for every packet destined for the firewall itself. For example, packets bound for a web server or SSH server running on the firewall itself must first traverse the INPUT chain.
- The **OUTPUT** chain is used for each packet generated by the firewall itself. For example, web requests or SSH connections made from the firewall itself first pass through the OUTPUT chain.
- The **FORWARD** chain is read for each packet passing through the firewall that was not generated the firewall itself, or destined for it. This is where the majority of filtering rules are inserted on firewall machines.

The **nat** table (for Network Address Translation) defines these chains:

- The **PREROUTING** chain is read for each packet passing through the firewall, not generated by or destined for it.
- The **OUTPUT** chain is used for packets generated by the firewall itself, but is executed before the OUTPUT stage of the filter table.
- The **POSTROUTING** chain is read for each packet passing through the firewall, not generated by or destined for it.

Figure 4.6 shows the path that a packet takes as it passes through the netfilter system when using NAT.

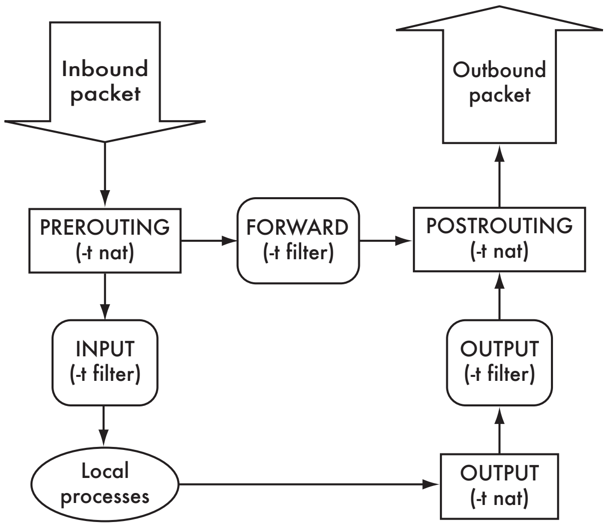


Figure 4.6: The netfilter process is applied to packets according to this schematic.

Each chain contains a number of rules. The kernel starts reading from the top of the chain, beginning with the first rule. It checks the conditions on each rule in turn, and if they match the packet, it executes the associated ***target*** on that packet.

Some targets cause the packet processing to finish immediately. These targets imply that a final decision has been made on what to do with the packet, such as whether to accept (allow) or drop (discard) it. The ***ACCEPT***, ***DROP***, and ***REJECT*** targets are examples of such terminating targets. Other targets cause a side effect, but allow processing to continue. For example, the ***LOG*** target writes some information about the packet to the system logs, but the packet will continue to pass down the chain until it reaches a rule with a terminating target.

On most systems, support for netfilter is not built into the kernel, and must be loaded as a kernel module. You should at least load the **`ip_tables`**, **`iptable_filter`**, and **`ipt_state`** modules, as well as any other advanced features you may want to use (such as **`iptable_nat`** and **`ipt_MASQUERADE`** for NAT). Because these commands affect the system's security, only the root user can run them. The following commands will load the basic modules:

```
# modprobe ip_tables
# modprobe iptable_filter
# modprobe ipt_state
```

To load these at boot time, add the appropriate lines to **`/etc/modules`**.

Before configuring a firewall, you should make sure that the kernel is allowed to act as a router, and will forward packets between interfaces. This is disabled by default for safety. You can enable it temporarily with the following command:

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

This setting will be lost when the system is rebooted. Most systems have an **`/etc/sysctl.conf`** file which defines default values for kernel variables like this. You can make the change permanent by adding the following line:

```
net.ipv4.ip_forward = 1
```

and remove any existing line that refers to **`net.ipv4.ip_forward`**.

The normal state of each chain is empty, but some Linux distributions configure a simple firewall automatically during installation. You can list the rules in each chain with the following **`iptables`** commands:

```
# iptables -L INPUT -n
# iptables -L FORWARD -n
# iptables -L OUTPUT -n
```

Each rule has a ***match condition***, which specifies the packets that match the rule, and a ***target***, which is activated for each packet that matches the conditions. For example, **`-p tcp -j DROP`** matches TCP packets and discards them.

Until you are more familiar with netfilter, you should only configure your firewall from the physical console of the machine, and never over the network. It is very easy to accidentally block your own access to the machine when configuring netfilter remotely.

The chains in the filter table, INPUT, OUTPUT, and FORWARD, are required to make a final decision for every packet that passes through the system. Therefore, they have a ***policy*** which is applied to each packet that does not match any rule in the chain. This can simply be thought of as the default target for the entire chain. The policy must be set to ACCEPT or DROP, and the default is ACCEPT. It is normally considered that a default ACCEPT policy is not secure, and that you should change the default policy to be DROP. This is known as ***deny by default***, and fits with the "German" network security model. The following commands change the default policy for each chain to DROP:

```
# iptables -P INPUT DROP
# iptables -P OUTPUT DROP
# iptables -P FORWARD DROP
```

To create your own firewall, you should delete any existing rules in the filter chain, using the following command:

```
# iptables -F
```

The filter table is used so often that it is implied when no explicit table is used. This is functionally equivalent to the command:

```
# iptables -t filter -F
```

...but is shorter to type. The **`-F`** option stands for ***flush***. You can also Flush individual chains:

```
# iptables -F INPUT
# iptables -F OUTPUT
# iptables -F FORWARD
```

Now all traffic is blocked on the firewall, regardless of its source or destination. To allow packets to pass through the firewall, you need to add rules which match them using the ACCEPT target. The **`iptables -A`** command appends a new rule to the end of a chain. With a deny by default policy, it is common practice to allow packets that are part of, or associated with, an established connection. Since the connection was established in the first place, we assume that subsequent packets in the same connection are also allowed.

The following commands allow such traffic into and through the firewall:

```
# iptables -A INPUT   -m state --state ESTABLISHED,RELATED -j ACCEPT
# iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
# iptables -A OUTPUT  -m state --state ESTABLISHED,RELATED -j ACCEPT
```

With established connections permitted, you can now specify the new TCP connections that should be permitted into the firewall. These should correspond to services running on the firewall itself that should be permitted. In this case, we will allow access to SSH (port 22), HTTP (port 80), SMTP (port 25), and TCP DNS (port 53) services on the firewall. Note that no new connections may be made through the firewall yet, as these rules only open connections for services on the firewall itself.

```
# iptables -A INPUT -p tcp --dport 22 -m state --state NEW -j ACCEPT
# iptables -A INPUT -p tcp --dport 80 -m state --state NEW -j ACCEPT
# iptables -A INPUT -p tcp --dport 25 -m state --state NEW -j ACCEPT
# iptables -A INPUT -p tcp --dport 53 -m state --state NEW -j ACCEPT
```

These rules each have multiple conditions that must be met in order to be accepted. For example, the first rule matches only if:

- The protocol of the packet is TCP, **and**
- The TCP destination port is 22 (SSH), **and**
- The state is NEW.

When a rule contains multiple conditions, they must all match before the target will be executed.

The rules above also make use of netfilter's ***stateful inspection*** features, also known as ***connection tracking***. The rules match on the state associated with the packet. The state is not part of the packet as transmitted over the Internet. Instead, they are determined by the firewall itself, after comparing the packet with each connection that it knows about:

- If the packet is part of an existing connection, its state is ESTABLISHED; otherwise:
- If the packet is related to an existing connection, such as an ICMP destination unreachable reply to a TCP packet sent earlier, its state is RELATED; otherwise:
- Its state is NEW.

Every NEW or RELATED packet adds a temporary entry to the connection tracking table, and every ESTABLISHED packet extends the life of the entry. If a connection is established through the firewall, but it sees no activity for some period of time, the entry in the table will expire, and further packets will be regarded as NEW again, rather than ESTABLISHED.

You cannot use the **`--sport`** and **`--dport`** options (which match the source and destination ports) unless you specify the protocol with **`-p tcp`** or **`-p udp`**. It is not possible to write a single rule that matches both TCP and UDP packets and specifies port numbers.

Some types of UDP traffic are very useful. For example, DNS requests are normally made over UDP rather than TCP. If the firewall runs a DNS server, then you will probably want to allow UDP port 53 traffic into it, with the following command:

```
# iptables -A INPUT -p udp --dport 53 -j ACCEPT
```

If your users use a DNS server out on the Internet, then you will need to give them access to it. Replace 10.20.30.40 with the DNS server's IP address in the following command:

```
# iptables -A FORWARD -p udp --dport 53 -d 10.20.30.40 -j ACCEPT
```

If you use globally routable IP addresses throughout your organisation, you may want to allow your users to have direct access to the Internet. This can be enabled with this command:

```
# iptables -A FORWARD -j ACCEPT
```

Alternatively, you may wish to force them to use a proxy to access the Internet. The proxy server should be the only computer that's allowed to make connections out onto the Internet. If the proxy server runs on the firewall itself, then the firewall must be allowed to make outgoing connections:

```
# iptables -A OUTPUT -j ACCEPT
```

If the proxy server is another computer inside your network, you will need a rule like this, replacing 10.10.10.20 with the proxy server's IP address:

```
# iptables -A FORWARD -s 10.10.10.20 -j ACCEPT
```

The options **`-s`** and **`-d`** in the commands above match source and destination addresses respectively. You can specify a range of addresses with a network mask, for example **`-s 10.10.50.0/24`** matches all source addresses from 10.10.50.0 to 10.10.50.255.

You can make your rules more explicit by specifying interfaces. The **`-i`** option matches the incoming interface, and **`-o`** matches the outgoing interface. You cannot specify **`-o`** for rules in the INPUT chain, because packets passing through INPUT will not leave the firewall, so they have no outbound interface. Similarly, you cannot specify **`-i`** for rules in the OUTPUT chain, but you can specify either or both for rules in the FORWARD chain. For example, if you want to allow traffic from 10.10.10.30 through the firewall, but only if it comes in on the eth0 interface and leaves on eth1:

```
# iptables -A FORWARD -s 10.10.10.30 -i eth0 -o eth1 -j ACCEPT
```

You may wish to allow ICMP packets in both directions, to allow the **`ping`** and **`traceroute`** commands to work to and from the firewall itself:

```
# iptables -A INPUT -p icmp -j ACCEPT
# iptables -A OUTPUT -p icmp -j ACCEPT
```

And you may wish to allow users on your inside network (e.g. eth0) to run **`ping`** and **`traceroute`** to the outside world (e.g. eth1) as well:

```
# iptables -A FORWARD -i eth0 -o eth1 -p icmp -j ACCEPT
# iptables -A FORWARD -i eth1 -o eth0 -p icmp -j ACCEPT
```

All the **`iptables`** commands above start with the **`-A`** option, which appends the specified rule to the end of the specified chain. Other useful options are:

- **`-L <chain>`** lists the rules in the specified chain, with the following useful options:

  `-n` stops the iptables command from trying to resolve IP addresses to hostnames. If running **`iptables -L`** seems to hang, you may be waiting for DNS resolution. Add **`-n`** to see the list immediately using just IP addresses.

  `-v` lists all details of the rules, including input and output interfaces and byte and packet counters

  `--line-numbers` gives the rule number next to each rule, which is useful for the **`-D`** and **`-I`** options
- **`-D <chain> <rule>`** deletes the first rule matching the specification from the specified chain
- **`-D <chain> <number>`** deletes the specified rule number from the specified chain
- **`-I <chain> <number> <rule>`** inserts the specified rule before the specified rule number in the chain

The packet and byte counters, shown with **`iptables -L -v`**, are very useful for debugging and optimising rules. The packet counter is increased by one every time a packet matches the rule. If you create a rule and it doesn't seem to be working, check the packet counter to see whether it is being matched. If not, then either the specification is wrong, or an earlier rule is capturing the packets that you wanted to match. You can try moving the rule higher up the chain (delete and re-insert it), or removing conditions until it starts to match.

If your firewall has very heavy traffic, you should optimise your rules so that the ones with the highest packet counters are higher up the chains. In other words, the rules should be in descending order by packet count. However, you should be careful not to violate your security policy by doing so. Rules which might match the same packets under any circumstances should not be reversed in order without careful thought.

It is also a good idea to add a LOG rule at the end of each chain, so that you can see from your firewall logs (**`/var/log/messages`**) which packets are not matching any rules. This can be useful for debugging and for spotting attacks on your firewall and network. You may wish to limit the rate of logging with the limit match, to avoid filling up your logs too fast:

```
# iptables -A INPUT -m limit --limit 10/min -j LOG
```

Other useful targets include ***REJECT***, which is like DROP in that the packet is not allowed to pass through the firewall. However, while DROP is silent, REJECT sends an ICMP destination unreachable message back to the originator. REJECT is more polite, because it tells the sender what happened to their packet, but DROP is more secure because it provides an attacker with less information and makes it much slower to scan your machine for open ports. You may want to consider dropping ICMP echo-request packets from outside your network for security reasons:

```
# iptables -A INPUT -p icmp --icmp-type echo-request -j DROP
```

You should ensure that this rule comes before any rule which allows ICMP packets in the INPUT chain (otherwise it will have no effect).

You can use NAT to rewrite the source address of packets forwarded by your firewall, to make them appear to have come from the firewall itself. This is called ***masquerading***, and is very useful if your internal network uses a private IP address range. For example, if your internal network uses 192.168.1.0/24, and the external interface is eth1, then you could use the following command:

```
# iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -o eth1 -j MASQUERADE
```

Once you are finished adding exceptions, test each of the rules from inside and outside your network. You can make connections manually, or use a security scanning tool such as ***nmap*** (*http://insecure.org/nmap/*).

The ruleset will be lost when the system is rebooted unless you save it first. On Debian (and Ubuntu) systems, you will need to install the iptables **`initscript`**:

```
# zcat /usr/share/doc/iptables/examples/oldinitscript.gz > /etc/init.d/iptables
# chmod a+x /etc/init.d/iptables
# ln -s /etc/init.d/iptables /etc/rcS.d/S39iptables
```

No other distribution makes it this difficult to use iptables. Normally you just have to make sure that iptables is started at boot:

```
# chkconfig iptables on
```

Whenever you update your ruleset and want to save your changes, use the following command on Debian/Ubuntu:

```
# /etc/init.d/iptables save active
```

And on other distributions:

```
# /etc/init.d/iptables save
```

You can discard the current ruleset and reload the last saved one with:

```
# /etc/init.d/iptables restart
```

You can find more about netfilter from the following resources:

- iptables manual page. EIther run **`man iptables`** or see: *http://www.linuxguruz.com/iptables/howto/maniptables.html*
- Packet Filtering HOWTO: *http://www.netfilter.org/documentation/HOWTO/packet-filtering-HOWTO-7.html*
- Network Address Translation HOWTO : *http://www.netfilter.org/documentation/HOWTO/NAT-HOWTO.html*
- Netfilter documentation: *http://www.netfilter.org/documentation/*

BWM Tools

BWM Tools (*http://freshmeat.net/projects/bwmtools/*) is a Linux utility which provides a wrapper to **`iptables`**, allowing all of its features to be configured using an easy XML syntax. Firewalling, bandwidth shaping, logging, and graphing are all supported.

BWM Tools consists of two main utilities. **`bwm_firewall`** is used for building the firewall rules, and **`bwmd`** is used for traffic shaping. BWM Tools works by queueing packets in userspace, where they are inspected, queued, rate limited, and given the go-ahead to pass. Live bandwidth graphs can be generated from any flow by using RRDtool file formats along with **`rrd-cgi`**, which provides for a good overall health check on networks.

More information about configuring and installing BWM Tools can be found at *http://bwm-tools.pr.linuxrulz.org/doc/*.

Below is a simple stateful firewalling example that allows SSH, SMTP, and DNS traffic to be received.

```
<firewall>
  # Global configuration and access classes
  <global>
    <modules>
      # Track FTP connections
      <load name="ip_nat_ftp"/>
      <load name="ip_conntrack_ftp"/>
    </modules>

    # Setup traffic classification classes
    <class name="ssh">
      <address proto="tcp" dport="22" />
    </class>

    <class name="smtp">
      <address proto="tcp" dport="25"/>
    </class>

    <class name="http">
      <address proto="tcp" dst-port="80"/>
    </class>

    <class name="dns_tcp">
      <address proto="tcp" dst-port="53"/>
    </class>

    <class name="dns_udp">
      <address proto="udp" dst-port="53"/>
    </class>

    <class name="new_connections">
      <address proto="tcp" cmd-line="--syn -m state --state NEW"/>
    </class>

    <class name="valid_traffic">
      <address cmd-line="-m state --state ESTABLISHED,RELATED"/>
    </class>
  </global>

  # Access list
  <acl>
    # Filter table
    <table name="filter">
      # Allow only SYN packets here...
      <chain name="new_connections">
        <rule target="ACCEPT">
          ssh;
          smtp;
          http;
          dns_tcp;
        </rule>
      </chain>

      # Deny-all default policy
      <chain name="INPUT" default="DROP">
        # Accept valid traffic, and UDP as its stateless
        <rule target="ACCEPT>
          valid_traffic;
          dns_udp;
        </rule>
        # Match SYN packets (tcp) and jump to new_connections
        <rule target="new_connections">
          new_connections;
        </rule>
      </chain>

      # Accept output  from ourselves
      <chain name="OUTPUT" default="ACCEPT">
      </chain>

      # Allow valid forwarded traffic, if we had any
      <chain name="FORWARD" default="DROP">
        <rule target="ACCEPT>
          valid_traffic;
        </rule>
      </chain>
    </table>
  </acl>
</firewall>
```

Shorewall

Shorewall (*http://shorewall.net/*) is a tool that can make setting up a firewall easier than using pure **`iptables`** commands. It is not a daemon or service, but is simply a tool that is used to configure netfilter.

Rather than dealing with the sometimes confusing tables, chains, and rules of netfilter, Shorewall abstracts the firewall configuration into a number of easy-to-read files that define interfaces, networks, and the sort of traffic that should be accepted. Shorewall then uses these files to generate the applicable netfilter rules.

There is an excellent configuration guide and basic introduction to networking concepts at: *http://shorewall.net/shorewall_setup_guide.htm*

Building a firewall in BSD

There are three firewalling systems in BSD, which are not compatible with each other. ***IPFW*** is the oldest and most efficient, but does not have as many features as the others. ***IPF*** was created to enhance IPFW, but the author decided to restrict its license, and so it is not completely free. ***PF*** was created as a free replacement for IPF. PF offers the most features, including packet shaping, but is claimed to be less efficient than IPFW. We will give simple examples of using IPFW and PF.

You can find out more about the relative benefits of each firewall here:

- *http://lists.freebsd.org/pipermail/freebsd-ipfw/2004-December/001583.html*
- *http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/firewalls.html*

In order to create a useful firewall, you will need to enable packet forwarding through your machine. This can be done with the following command:

```
sysctl net.inet.ip.forwarding=1
```

To make this change permanent, assuming that packet forwarding is disabled, add the following line to **`/etc/sysctl.conf`**:

```
net.inet.ip.forwarding=1
```

To build a firewall with **IPFW**, first enable IPFW functionality in the kernel. The procedure varies between different BSDs. The method described here is for FreeBSD.

```
# cd /usr/src/sys/i386/conf/
# cp GENERIC MYFIREWALL
```

Open MYFIREWALL with your favorite text editor. At the bottom of the file add these lines:

```
options         IPFIREWALL
options         IPFIREWALL_VERBOSE
options         IPFIREWALL_FORWARD
options         IPFIREWALL_VERBOSE_LIMIT=100
options         IPFIREWALL_DEFAULT_TO_ACCEPT
```

Save your work and compile the kernel with the following commands:

```
# /usr/sbin/config MYFIREWALL
# cd ../compile/MYFIREWALL
# make cleandepend
# make depend
# make
# make install
```

To ensure that the firewall is activated at boot time, edit the file **`/etc/rc.conf`** and add the following lines:

```
gateway_enable="YES"
firewall_enable="YES"
firewall_type="myfirewall"
firewall_quiet="NO"
```

Now reboot the computer to ensure that the firewall is active.

Edit the **`/etc/rc.firewall`** file and adapt it to your needs. You can apply the new rules with the command **`sh /etc/rc.firewall`**.

For more information on configuring IPFW, please refer to the manual page (**`man ipfw`**) and the FreeBSD Website:

http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/firewalls-ipfw.html

To activate **PF**, you should first enable it in the kernel. The procedure varies between different BSDs. The method described here is for FreeBSD.

```
# cd /usr/src/sys/i386/conf/
# cp GENERIC MYFIREWALL
```

Then add the following at the bottom of MYFIREWALL:

```
device pf
device pflog
device pfsync
```

Save the file and then recompile your kernel.

To ensure that PF is activated at boot time and logging is enabled, add this to **`/etc/rc.conf`**:

```
gateway_enable="YES"
pf_enable="YES"
pf_rules="/etc/pf.conf"
pf_flags=""
pflog_enable="YES"
pflog_logfile="/var/log/pflog"
pflog_flags=""
```

Now reboot the machine to ensure that PF is loaded. You should be able to run the following commands:

- **`pfctl -e`** to enable PF

- **`pfctl -s`** all to show list all firewall rules and status
- **`pfctl -d`** to disable PF

Edit the **`/etc/pf.conf`** file and uncomment all the lines that apply to your setup. You can apply the new rule set with the command **`pfctl -f /etc/pf.conf`**.

You can find out more about configuring PF in the manual page (**`man pf.conf`**) and the OpenBSD website:

http://www.openbsd.org/faq/pf/

Transparent bridging firewall in FreeBSD

Chances are good that if you are low on bandwidth, you are also short on IP addresses. Many organisations only have a few public IP addresses - perhaps one for the router and another for the proxy server. So how do you introduce a firewall without having to restructure the network or use yet another valuable IP address?

One way to do this is to use a ***transparent bridging firewall***. Such a firewall does not require an IP address, and is able to to protect your LAN, route traffic, and be integrated seamlessly into the network. As your network grows, you can add capacity by simply adding more network cards. FreeBSD has a special feature in its kernel that allows it to function as a bridge, after which you can use any of the firewall programs available in FreeBSD (including ***IPFW***, ***PF***, or ***IPF***). To build a transparent firewall with ***IPFW***, first enable IPFW and bridging functionality in the kernel.

```
# cd /usr/src/sys/i386/conf/
# cp GENERIC MYFIREWALL
```

Open MYFIREWALL with your favorite text editor. At the bottom of the file add these lines:

```
options         IPFIREWALL                      #firewall
options         IPFIREWALL_VERBOSE              #enable logging to syslogd(8)
options         IPFIREWALL_FORWARD              #transparent proxy support
options         IPFIREWALL_VERBOSE_LIMIT=100    #limit verbosity
options         IPFIREWALL_DEFAULT_TO_ACCEPT    #allow everything by default
options         BRIDGE
```

Save your work and compile the kernel with the following commands:

```
# /usr/sbin/config MYFIREWALL
# cd ../compile/MYFIREWALL
# make cleandepend
# make depend
# make && make install
```

To ensure that the firewall is activated at boot time, edit the file **`/etc/rc.conf`** and add the following lines:

```
gateway_enable="YES"
firewall_enable="YES"
firewall_type="myfirewall"
firewall_quiet="NO"
```

Next, we need to tell FreeBSD which interfaces will be bridged together. This is done by editing **`/etc/sysctl.conf`** to include the following:

```
net.link.ether.bridge.enable=1
net.link.ether.bridge.config=if1,if2
net.link.ether.bridge.ipfw=1
```

Replace if1 and if2 with your network interfaces. Finally, you can edit **`/etc/rc.firewall`** and insert your desired filtering rules.

To activate bridging with ***PF***, you should first enable PF and bridging support in the kernel.

```
# cd /usr/src/sys/i386/conf/
# cp GENERIC MYFIREWALL
```

Then add the following at the bottom of **`MYFIREWALL`**:

```
device pf
device pflog
device pfsync
options BRIDGE
```

Save the file and then recompile your kernel. To ensure that PF is activated at boot time and logging is enabled, add this to **`/etc/rc.conf`**:

```
gateway_enable="YES"
pf_enable="YES"
pf_rules="/etc/pf.conf"
pf_flags=""
pflog_enable="YES"
pflog_logfile="/var/log/pflog"
pflog_flags=""
```

For PF, you also need to create a bridge interface with the names of the network cards you want to use. Add the following to **`/etc/rc.conf`**, replacing **`xl0`** and **`xl1`** with the network devices on your computer:

```
cloned_interfaces="bridge0"
ifconfig_bridge0="addm xl0 addm xl1 up"
```

To ensure the two network interfaces are activated at boot, add the following to **/etc/rc.local**:

```
ifconfig xl0 up
ifconfig xl1 up
```

Then activate filtering on the two interfaces by adding these lines to **/etc/sysctl.conf**:

```
net.link.bridge.pfil_member=1
net.link.bridge.pfil_bridge=1
net.inet6.ip6.forwarding=1
net.inet.ip.forwarding=1
```

Finally, you can build a transparent bridging firewall using IPF. IPF support is already active on a FreeBSD install, but bridging is not. To use IPF, first enable bridging in the kernel:

```
# cd /usr/src/sys/i386/conf/
# cp GENERIC MYFIREWALL
```

Add these lines to **MYFIREWALL**:

```
options BRIDGE
options IPFILTER
options IPFILTER_LOG
options IPFILTER_DEFAULT_BLOCK
```

Then rebuild the kernel as described above. To activate IPF at boot time, edit **/etc/rc.conf** and add these lines:

```
ipfilter_enable="YES"             # Start ipf firewall
ipfilter_program="/sbin/ipf"
ipfilter_rules="/etc/ipf.rules"   # loads rules definition text file
```

Now edit **/etc/sysctl.conf** in order to create the bridged interfaces. Add the following lines, replacing **xl1** and **fxp0** for your Ethernet devices.

```
net.link.ether.bridge.enable=1
net.link.ether.bridge_ipf=1
net.link.ether.bridge.config=xl1:0,fxp0:0,xl0:1,rl0:1
net.inet.ip.forwarding=1
```

You can also create separate VLANs with this method. If you have four network cards, you can bridge them in pairs to create two separate networks. In the example on the next page, **xl1:0** and **fxp0:0** will bridge one network segment while **xl0:1** and **rl0:1** will bridge another.

```
net.link.ether.bridge.enable=1
net.link.ether.bridge_ipf=1
net.link.ether.bridge.config=xl1:0,fxp0:0,xl0:1,rl0:1
net.inet.ip.forwarding=1
```

Transparent bridging firewall in Linux

The Linux kernel also supports bridging and firewalling. In version 2.4, this required an optional patch, but in version 2.6 it is enabled by default. You only need to make sure that the **`BRIDGE_NETFILTER`** option is enabled. To check this, run the **`make menuconfig`** command in the kernel source directory and select the following options:

- Device Drivers
- Networking support
- Networking options
- Network packet filtering
- Bridged IP/ARP packets filtering

Ensure that the last option is enabled, with an asterisk (*) in the box. If you had to change any options, recompile your kernel with the **`make`** command, and install it with **`make install`**. Finally, reboot the machine to load the new kernel.

Bring down the network devices that you want to bridge together, e.g. eth0 and eth1:

```
# ifconfig eth0 down
# ifconfig eth1 down
```

Create a new bridge interface with the following command:

```
# brctl addbr br0
```

Add the network devices that you want to bridge together to the bridge device:

```
# brctl addif br0 eth0
# brctl addif br0 eth1
```

Bring all the interfaces up:

```
# ifconfig br0 up
# ifconfig eth0 up
# ifconfig eth1 up
```

Manually assign an IP address to the bridge, e.g.:

```
ifconfig br0 10.20.30.40 netmask 255.255.255.0
```

Now you should be able to write **`iptables`** firewall rules that control traffic through the bridge by specifying **`-i br0`** and/or **`-o br0`**. For example, to allow all traffic through the bridge:

```
# iptables -A FORWARD -i br0 -o br0 -j ACCEPT
```

To block all packets on port 80 from passing through the bridge:

```
# iptables -A FORWARD -i br0 -o br0 -p tcp --dport 80 -j DROP
```

You can also use **`physdev-in`** and **`physdev-out`** to match the actual physical device on which the packet entered or left the firewall:

```
iptables -A FORWARD -i br0 -o br0 -m physdev --physdev-in eth0 \
  --physdev-out eth1 -p tcp --dport 80 -j DROP
```

The above rule will drop packets to TCP port 80 (HTTP) going from eth0 to eth1, but not the other way around. The bridge configuration will be lost when you reboot your machine, so you may wish to add the commands to create the bridge to **`/etc/rc.d/rc.local`**, or the equivalent file on your distribution.

Note that once you add an interface to a bridge with **`brctl addif`**, netfilter will see packets through that interface as coming from or going to the bridge device (br0) instead of the real physical interface. You will need to adjust any existing firewall rules that refer to real physical devices, such as **`-i eth0`** or **`-o eth1`**, replacing the device name with **`br0`**.

Summary

Firewalls are an important way to control access to and from your network. You could think of them as being like locks on the doors of a house: they are necessary, but not sufficient for high security. Also, firewalls normally work on IP packets and connections, but there are times when you want more control over what can be done at the application layer, for example blocking certain websites and other content. This can be done through the use of an application layer firewall, such as a web proxy with access control (e.g. Squid).

Caching

It takes time to retrieve information from sources on the Internet. The amount of time it takes depends on a variety of factors, such as the distance to the destination and how many other people are requesting data at the same time. Greater distances between the server and client mean longer delays when sending and receiving data, since (as far as we know) electrical signals are bounded by the speed of light. When you consider the route a single packet may need to travel in order to reach California from Nairobi (up 35 000 Km to a satellite, 35 000 Km back down again, then across the ocean, possibly through another satellite trip, and across an entire continent halfway around the world, and then back again) it's no wonder that there is an upper limit to how fast information can travel across the net.

While we can't do much about changing the speed of light, we can definitely address the problem of too many people requesting information at once. Much of the information requested from the Internet is web traffic. If we can keep a local copy of data retrieved from the web, and intercept subsequent requests for the same information and serve the local copy instead, we can free up significant bandwidth for other uses, and greatly improve the overall feel of the network. Serving images from a local cache may take a few milliseconds, as compared with hundreds (or thousands) of milliseconds to retrieve the same data from distant Internet sites.

Web traffic is the most obvious service that can benefit from a cache, but just about any service that doesn't deal with realtime data can be cached. Prerecorded video streams can be cached, but live video cannot. Voicemail systems can be cached, but ***Voice over IP*** (***VoIP***) cannot. One service that should definitely be cached is DNS. Since virtually every other service makes use of hostname-to-IP address lookups, caching DNS will help "speed" up nearly everything else on the network, while saving bandwidth for other uses.

In this section, we will see specific examples that illustrate how to implement both web and DNS caching on your network.

Web caching

As mentioned earlier, ***caching web proxies*** work in much the same way as does a local browser cache. By saving requested information to a local disk, data is then served on subsequent requests to anyone on the network who needs it. How much can you really save with a web cache such as ***Squid***? While this varies depending on your traffic profile, you can expect to save, on average, between 20% and 30% of your bandwidth. As your users request more cacheable data, and the more those requests overlap, the more bandwidth you will save.

There are literately hundreds of web caches made available by the open source community and commercial vendors. Squid is undoubtedly the most popular web caching software available. It is mature, robust, very fast, and completely open source. In this book we will mainly concentrate on the Squid proxy server, but if you are running another proxy server the general concepts will still apply.

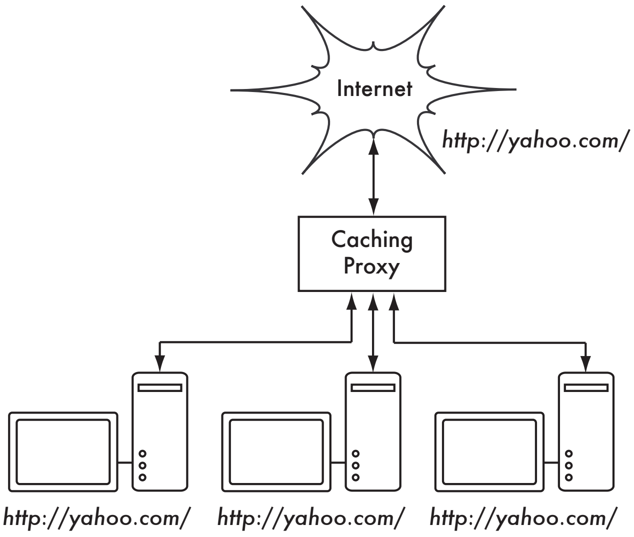


Figure 4.7: Caching proxy servers act as a shared web browser disk cache for all of the users in your network.

Besides saving Internet bandwidth, ***authenticating cache servers*** can provide you with increased control over the ways in which the network is used. When network usage is logged per user, you can then implement various kinds of behaviour modifying techniques, such as quotas or billing. The cache server also gives you a central place from which to watch the browsing habits of your users. The logs that are produced by the proxy server are useful when combined with a log file analysis tool such as Webalizer or Analog (page **81**). Other features include:

- **Bandwidth limiting / traffic shaping**. By implementing delay pools in Squid (page **189**), you can prioritise traffic in ways that make use of network resources in an equitable manner.
- **Redirection and content filtering**. This allows you to block advertisements, resample images, or make other changes to the content before it reaches the client. You can also redirect users to a page of your choice based on certain criteria.
- **Advanced Access Control List (ACL) processing**. ACLs allow you to perform firewall-like features at the application layer (for example, blocking some users while letting others through based on their credentials, IP address, or even time of day).
- **Peer caching**. This allows large sites to use multiple caches and share the cached files between them. By using peer caching you can scale your cache services to networks of any size.

When properly implemented, cache servers can have a significant impact on your bandwidth usage (even before traffic shaping and other access modification tools have been applied).

The caching server and the Firewall

The more a cache server is used, the more effective it will become. Your web cache is not fully effective unless it is used consistently throughout your organisation. If your users can easily bypass the proxy, they will certainly do so for a variety of reasons. They may not like the idea of all of their network requests being logged and monitored. If you have implemented delay pools or other access restrictions, they may notice that bypassing the proxy increases their performance. This is equivalent to cutting in line at a crowded theater. It may work for one user, but it is cheating, and can't work for everyone.

A proxy server necessarily goes hand-in-hand with a good firewall (page **114**). Configuring the firewall to only allow web access via the proxy is one way to ensure that the caching proxy is the only way that users can access the web. Of course, having a firewall doesn't do much good if it isn't properly configured.

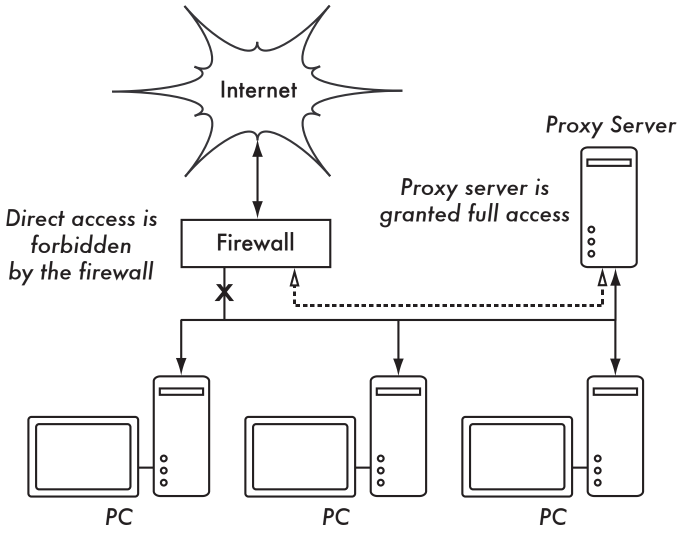

Internet
Proxy Server
Proxy server is granted full access
Direct access is forbidden by the firewall
Firewall
X
PC
PC
PC

Figure 4.8: Firewalls should be used to enforce use of the caching proxy. With a good firewall in place, the only way to access web content is via the proxy.

It is a common mistake to install a firewall for security, but to allow outbound access to the HTTP/FTP ports (TCP ports 80 and 21). Although you are protecting your network with the firewall, you are not forcing your users to use the cache server. Since your users are not forced to use the cache server, users that access the web directly are potentially wasting bandwidth. This also becomes a problem when you authenticate users on your cache server and rely on those logs to determine network usage.

Smaller networks can often install a cache server on the same server as the firewall. In larger networks, the cache server is placed somewhere on the internal network. On very busy networks, multiple cache servers may be used.

Transparent versus manual caches

A ***transparent cache*** is a caching server that works without any explicit proxy configuration on the client. All web traffic is silently redirected to the cache, and responses are returned to the client. This configuration is sometimes called an ***interception caching*** server.

With a traditional cache, the client is configured to use the cache server by either manually setting the proxy settings in the browser preferences, or by using an automatic configuration URL (page **140**). Transparent caches require no configuration on the client side. As far as the client knows, they are receiving web pages directly from the originating servers.

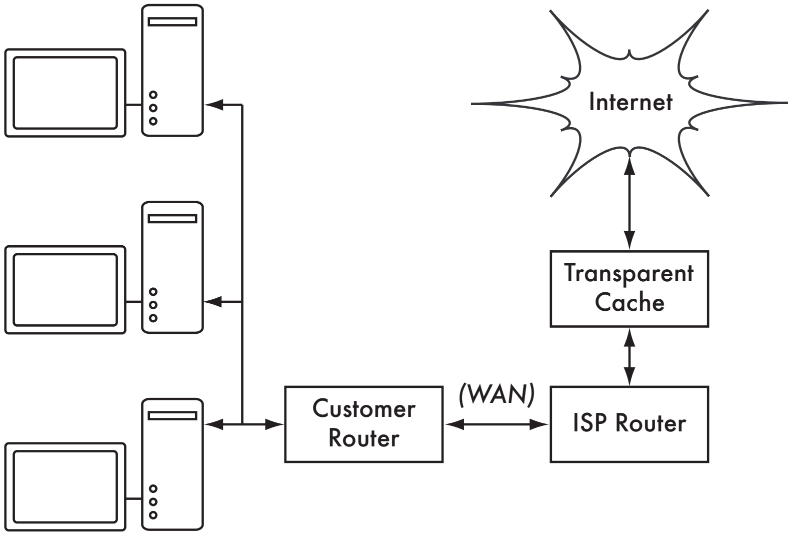


Figure 4.9: Transparent caches can be used by ISPs to save bandwidth by locally caching requests made by their customers.

Transparent caching can make problem solving difficult, as the user may not be aware that their requests are being served by the cache server and not from the original website. Transparent caching also makes proxy authentication impossible, since no credentials are presented to the cache server. This removes the ability to log and create "accountability" reports. While you can still log based on source IP address, reports at the user level can only be achieved by using authentication. The only real advantage in using a transparent cache is that the clients do not need to be configured. For installations with casual users who bring their own equipment (as is the case in cafes or public labs) this can help to reduce support requests. Transparent caches are often used as upstream caches in the ISP environment.

Transparent caching can be an involved process as it depends on specific features in your firewall and kernel. To learn how to setup a transparent cache with Squid, and various other firewalls, see the documentation at: *http://www.squid-cache.org/Doc/FAQ/FAQ-17.html*

Running Squid

Squid is the premier open source web caching proxy server originally developed by Duane Wessels. Squid has been around for over ten years and is probably the most popular web caching server in use today.

To quote from the Squid wiki;

> "*Squid is a high-performance proxy caching server for web clients, supporting FTP, gopher, and HTTP data objects. Unlike traditional caching software, Squid handles all requests in a single, non-blocking, I/O-driven process. Squid keeps meta data and especially hot objects cached in RAM, caches DNS lookups, supports non-blocking DNS lookups, and implements negative caching of failed requests. Squid supports SSL, extensive access controls, and full request logging. By using the lightweight Internet Cache Protocol, Squid caches can be arranged in a hierarchy or mesh for additional bandwidth savings. Squid consists of a main server program squid, an optional Domain Name System lookup program dnsserver (Squid nowadays implements the DNS protocol on its own by default), some optional programs for rewriting requests and performing authentication, and some management and client tools.*"

You can download Squid from *http://www.squid-cache.org/*. It is also available as a standard package in most Linux distributions. A Windows port is also available at *http://www.acmeconsulting.it/SquidNT/*. For most installations, the default Squid build included in your system package is probably sufficient. In some circumstances you may want to rebuild Squid from source.

Some options, such as delay pools and a few of the authentication helpers, require a recompile. You may also want to recompile Squid, for large installations, to increase the number of available file descriptors (the default is 1024). You should monitor your Squid box with the ***cachemgr*** interface to determine if you have sufficient file descriptors. Detailed instructions on how to increase your file descriptors are available in the Squid FAQ at *http://www.squid-cache.org/Doc/FAQ/FAQ-11.html#ss11.4*.

A Squid configuration file can be as simple as four lines:

```
http_port 3128
cache_mgr your@email.address.here
acl our_networks src 192.168.1.0/24
http_access allow our_networks
```

Change the network ACL to reflect your IP address range, add a contact email address, and that's it. Save the file as **`/etc/squid.conf`**.

When you run Squid for the first time, you will have to initialise the cache directory. To do this, run Squid with the **`-z`** switch:

```
# squid -z
```

Finally, run **`squid`** with no other parameters to launch the Squid daemon.

```
# squid
```

Congratulations, you now have a caching web proxy! All that's left now is to configure your users' browsers to use the proxy. If you only administer a few computers, you can do that manually. For larger installations, it can be easier to enable automatic proxy configuration (see the next section).

While this example will get you started, Squid can be fine-tuned to optimise networks of just about any size. Full documentation is available at *http://www.visolve.com/squid/*. For more advanced features, see the **Performance Tuning** chapter on page **177**. Many more tips and techniques are also available on the Squid Wiki at *http://wiki.squid-cache.org/*.

Automatic web proxy configuration

As mentioned earlier, the main advantage of transparent proxies is that the cache can be used automatically without changing browser settings. The trade-off for this convenience is that some proxy features, such as authentication, cannot be used. One alternative to using transparent proxies is an ***automatic proxy configuration***. This technique allows browsers to automatically detect the presence of the proxy, and works with all proxy features (including authentication).

When configured for automatic proxy discovery, the browser requests a file that defines which proxy to use for a given URL. The file contains javascript code that defines the function **`FindProxyForURL`**. It looks like this:

```
function FindProxyForURL(url, host)
         {
             ...
         }
```

This script should return a string that defines the proxy server to be used (in the format **`PROXY host:port`**), or if a SOCKS proxy is used, it should return the string **`SOCKS host:port`**. The string **`DIRECT`** should be returned if a proxy is not needed. This function is run for every request, so you can define different values for different URLs or hosts. There are also several functions that can be called in the script to help determine what action to take. For example, **`is-PlainHostName(host)`** returns true if there is no domain specified for the given host, and **`isResolvable(host)`** returns true if the host name resolves to an IP address.

The file containing the **`FindProxyForURL`** function is usually given a **`.pac`** extension (short for ***Proxy Auto Configuration***) and is placed in a directory on a local web server.

Here is an example **`config.pac`** file that returns **`DIRECT`** for plain host names and local sites, and returns a proxy for everything else:

```
function FindProxyForURL(url, host)
    {
        if (isPlainHostName(host) ||
            dnsDomainIs(host, ".mydomain.com"))
            return "DIRECT";
        if (isInNet(host, "10.1.0.0", "255.255.0.0"))
            return "DIRECT";
        else
            return "PROXY proxy.mydomain.com:8080";
    }
```

You may also need to tell your web server to associate **`.pac`** with the proper mime type. Make sure there is an entry like this in the **`mime.types`** file for your web server:

```
application/x-ns-proxy-autoconfig pac
```

A full description of **`FindProxyForURL`** can be found online at *http://wp.netscape.com/eng/mozilla/2.0/relnotes/demo/proxy-live.html*.

Now that we have a working **`config.pac`**, how do we tell our clients to use it? There are three ways to do it: by direct configuration in the browser, with a DNS entry, or via DHCP.

Most browsers have a place in their preferences or settings where the URL to the **`config.pac`** can be defined. In Firefox this is the ***Automatic proxy configuration URL***, and in Internet Explorer it is called ***Use automatic configuration script***. To configure the location manually, check these boxes and insert the URL to the **`config.pac`** script in the appropriate boxes. While this may seem counterintuitive (why do you need to manually define a URL for automatic configuration?) this method still gives you one significant benefit: once your browsers are configured, they never need to be changed again. The entire network can be configured to use new proxy settings by changing the single **config.pac** file.

The DHCP and DNS methods allow you to define the location of the automatic configuration file without hard coding a URL on every browser. This is implemented by the ***Web Proxy Auto Discovery*** (***WPAD***) protocol. The protocol provides a number of methods for generating a URL that refers to a Proxy Auto Configuration file. This option is enabled in Firefox with the ***Auto-detect proxy settings for this network*** setting. In Internet Explorer, the option is called ***Automatically detect settings in IE***.

The DHCP method is tried first, followed by a series of DNS lookups. To use the DHCP method, define the URL to the **config.pac** using DHCP option 252. In ISC's dhcpd, the configuration looks like this:

```
# Config for ISC DHCP server
option wpad code 252 = text;
option wpad "http://myserver.mydomain/config.pac";
```

When clients request a DHCP lease, they are also given the URL pointing to the **config.pac**. Your clients may need to be rebooted (or their leases released and renewed) in order to pick up the change.

If no DHCP option 252 is provided, the WPAD protocol next attempts to find the proxy server using DNS. If the client has a domain name of *mydomain.com*, the browser will try to look up *wpad.mydomain.com*. If the DNS request is successful, the browser makes an HTTP connection to that address on port 80 and requests the file **/wpad.dat**. This file should have the same contents as the **config.pac** file. To use the DNS method, you will need to configure your DNS server to reply to requests for *wpad.mydomain.com* with the IP address of your web server. You should copy (or rename) the **config.pac** file to **wpad.dat** and place it in the root directory of your web server. Make sure your web server returns the proper mime type for .dat files with an entry like this in your **mime.types**:

```
application/x-ns-proxy-autoconfig dat
```

Now that your browsers can automatically find the proxy server, you can make network-wide changes in a single place. Be sure that the web server that serves the configuration file is always available. If that file cannot be found, then your clients will not be able to find the proxy, and they won't be able to browse.

DNS caching

DNS caching is used to save bandwidth on DNS queries and improve response times. Improvements to DNS can make every other service (including web browsing and email) "feel" faster, since virtually all services make use of host lookups. Each DNS record has a ***Time To Live*** (***TTL***) that defines how long the entry is valid. All name server responses include a TTL along with the record requested. DNS caches keep a record in memory or on disk until the TTL expires, thus saving bandwidth by not having to do queries for frequently requested records.

Most DNS servers will act as a caching proxy. Here are examples of how to implement a cache using dnsmasq, BIND, and djbdns.

dnsmasq

Dnsmasq is a lightweight, easy to configure DNS forwarder and DHCP server. It is available for BSD and most Linux distributions, or from *http://freshmeat.net/projects/dnsmasq/*. The big advantage of dnsmasq is flexibility: it acts as both a caching DNS proxy and an authoritative source for hosts and domains, without complicated zone file configuration. Updates can be made to zone data without even restarting the service. It can also serve as a DHCP server, and will integrate DNS service with DHCP host requests. It is lightweight, stable, and flexible. Bind is likely a better choice for large networks (more than a couple of hundred nodes), but the simplicity and flexibility of dnsmasq makes it attractive for small to medium sized networks.

Setting up Dnsmasq is pretty easy. If you already have **`/etc/hosts`** and **`/etc/resolv.conf`** set up, run dnsmasq and point other computers to the server's IP address (using DHCP or manually configuring the DNS server). For more complex configuration, the **`/etc/dnsmasq.conf`** file contains documentation explaining all the various variables.

BIND (named)

The ***Berkeley Internet Name Domain*** (***BIND***) is capable of serving zones, acting as a slave, performing caching and forwarding, implementing split horizon (page **212**), and doing just about anything else that is possible with DNS. BIND is used by a vast majority of the Internet community to serve DNS, and considered to be stable and robust. It is provided by the Internet Software Consortium at *http://www.isc.org/sw/bind/*. Virtually all versions of Linux and BSD include a package for BIND.

To run BIND as a caching DNS server, make sure there is an entry for the root zone in your **named.conf**:

```
# Hints file, listing the root nameservers
zone "." {
        type hint;
        file "root.cache";
};
```

That's all there is to it. Make sure your clients use this server for DNS instead of your ISP's DNS server, and you're good to go.

djbdns

DJBDNS is a DNS server implemented by D. J. Bernstein. It is focused on security, and is claimed to be free of security holes. Unfortunately, djbdns has a restrictive license which prevents patching and redistribution of patched executables, making it difficult for distribution maintainers to package it. To use djbdns, you will need to download it from *http://cr.yp.to/djbdns.html* and compile it yourself.

Once djbdns has been built and installed, it can be configured as follows to provide DNS caching. First, create the users and configuration we need:

```
$ useradd Gdnscache
$ useradd Gdnslog
$ dnscache-conf Gdnscache Gdnslog /etc/dnscache 127.0.0.1
```

Next, install it as a service:

```
$ ln -s /etc/dnscache /service
$ sleep 5
$ svstat /service/dnscache
```

DNS queries made on this server should now be cached.

Mirroring

Mirroring is a technique used to synchronise files between hosts. This can be used to store local copies of software updates on a server, provide redundancy for load distribution between servers, or even perform backup. Mirroring saves bandwidth by providing a local copy of popular data. Users request information from the mirror rather than directly from the site on the Internet. Mirrors are updated at off-peak times to minimise the impact on other services.

Software Updates

Most packages or operating systems require software updates. If you look at the output from your proxy log file analysis, it will become apparent that one or two software update sites contribute significantly to your bandwidth consumption. While a caching proxy will take some of the load off, you can proactively populate a local mirror with software updates, and redirect your users there. Some common applications that update themselves are:

- **Windows.** Windows updates are a common problem, since machines asynchronously decide when to update themselves, and download redundant copies of the same patches. Fortunately, you can to set up a ***Microsoft Windows Server Update Services*** (***WSUS***) mirror on your local network. Clients can then be made to point to the local server for updates through a simple registry entry. Full details on setting up your own Windows Update Server can be found at *http://www.microsoft.com/wsus*
- **Adobe software.** If you have Adobe Acrobat installed on several machines, it will frequently look for and download updates. Keep local, up to date copies and encourage users to disable automatic updates.
- **Linux distributions.** Some Linux distributions can download vast amounts of updates automatically at regular intervals. You should decide on a few distributions that you want to support and try to keep a local mirror, even if it is only used for updates. Data repositories for Linux distributions can be mirrored locally using rsync or wget, and configured to be used by system update tools (such as yum and apt).

There are many other software packages that "call home" to look for updates. Some of these include anti-virus software, most shareware, and even the Mac OS X operating system. By watching your proxy logs (page **81**), you can get a clear idea of what information is frequently requested, and make local copies to remove some of the load on the Internet connection.

Some services lend themselves well to mirroring, while others do not. Large software repositories (such as Linux and BSD distributions, public FTP servers, software update sites, etc.). are ideal for local mirroring. Dynamic sites (such as news sites and web mail services) cannot be mirrored, but may benefit from caching (page **135**). While there isn't a standard method for mirroring arbitrary sites, several tools can help you create local copies of Internet content.

GNU wget

GNU ***wget*** is a free utility for non-interactive download of files from the web. It supports HTTP, HTTPS, and FTP, as well as retrieval through HTTP proxies. By limiting the transfer rate to any value, wget can shape usage to fit the available bandwidth. It can be run manually or in the background as a non-interactive process, downloading as many web pages as you like from a particular site. This allows you to start a retrieval and disconnect from the system, letting wget finish the work.

Wget supports a full-featured recursion mechanism, through which it is possible to retrieve large parts of the web automatically. The level of recursion and other mirroring parameters can be specified. It respects the ***robot exclusion standard*** (specified by the file **`robots.txt`** in the root directory of a web server).

Wget has been designed for robustness over slow or unstable network connections. If a download fails due to a network problem, it will keep retrying until the whole file has been retrieved. If the server supports resumed downloads, it will instruct the server to continue the download from where it left off. Wget can even convert the links in downloaded HTML files to point to local files. This feature facilitates off-line viewing.

There are quite a few command-line options for Wget. From a terminal, **`man wget`** or **`wget --help`** will list all of them. Here are some examples to get you started.

This will create a mirror of *www.yahoo.com*, retrying each request once when errors occur. The result will have the same directory structure as the original website. Results are logged to the file **`wget.log`**.

```
wget -r -t1 http://www.yahoo.com/ -o wget.log
```

You can schedule wget to run at any time using cron. This crontab will mirror the Xemacs ftp server every Sunday at midnight:

```
 0 0 * * 0 wget --mirror ftp://ftp.xemacs.org/pub/xemacs/ -o /home/me/xfer.log
```

The **`--mirror`** option turns on recursion and time stamping, sets infinite recursion depth and keeps ftp directory listings.

Wget is so popular that it is included by default in many Linux distributions. You can download the latest version from *http://www.gnu.org/software/wget/*.

Curl

Like wget, ***curl*** is a command line tool for downloading data from URLs, but is far more flexible. From the *http://curl.haxx.se/* website:

> *curl is a command line tool for transferring files with URL syntax, supporting FTP, FTPS, TFTP, HTTP, HTTPS, TELNET, DICT, FILE and LDAP. curl supports SSL certificates, HTTP POST, HTTP PUT, FTP uploading, HTTP form based upload, proxies, cookies, user+password authentication (Basic,*

> *Digest, NTLM, Negotiate, kerberos...), file transfer resume, proxy tunneling and a busload of other useful tricks.*

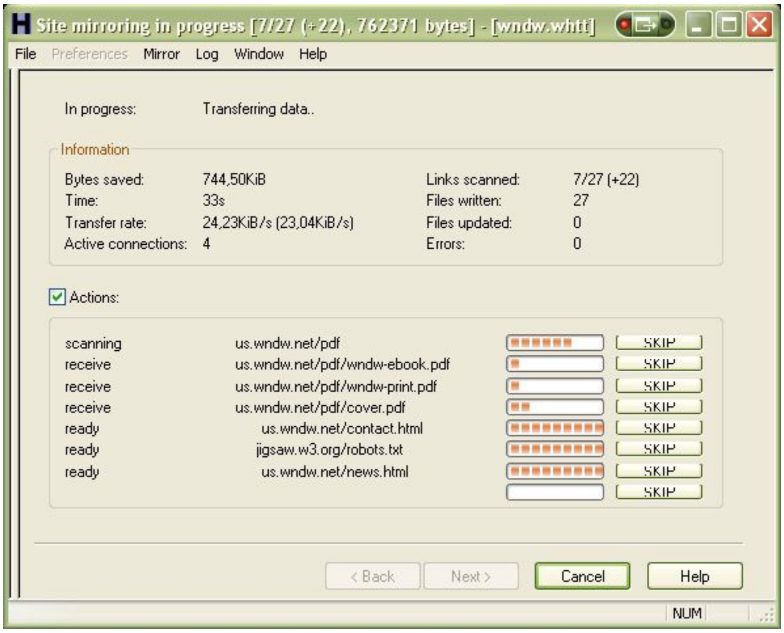
One difference between wget and curl is that curl assumes that you want to use standard out for the retrieved data. To retrieve a web page and save it to **`file.html`**, use the **`-o`** switch:

```
curl -o file.html http://www.yahoo.com
```

Unfortunately, one feature that curl does **not** have is recursive downloading. But this is supported using a wrapper script for curl, such as **`curlmirror`** (*http://curl.haxx.se/programs/curlmirror.txt*). Run **`curlmirror -?`** for a list of options. Curl can be downloaded from the main web site, and is included in many modern Linux distributions.

HTTrack

HTTrack (*http://www.httrack.com*) is a website mirroring and offline browser utility. It allows you to recursively download and store entire websites, rewriting URLs to facilitate offline browsing.



Site mirroring in progress [7/27 (+22), 762371 bytes] - [wndw.whtt]

In progress: Transferring data..

Information

Bytes saved:	744,50KiB	Links scanned:	7/27 (+22)
Time:	33s	Files written:	27
Transfer rate:	24,23KiB/s (23,04KiB/s)	Files updated:	0
Active connections:	4	Errors:	0

Actions:

```
scanning   us.wndw.net/pdf                         SKIP
receive    us.wndw.net/pdf/wndw-ebook.pdf          SKIP
receive    us.wndw.net/pdf/wndw-print.pdf          SKIP
receive    us.wndw.net/pdf/cover.pdf               SKIP
ready      us.wndw.net/contact.html                SKIP
ready      jigsaw.w3.org/robots.txt                SKIP
ready      us.wndw.net/news.html                   SKIP
                                                   SKIP
```

Figure 4.10: HTTrack can save entire websites for offline browsing.

It can also update an existing mirrored site without downloading the entire contents again, as well as resume interrupted downloads. There are versions available for Windows 9x/NT/2000/XP, as well as Linux, BSD, and Mac OS X.

HTTrack has a friendly, intuitive interface and an extensive help system. It is much easier to run than other mirroring software, but lacks some features (such as download scheduling). It is open source and released under the GPL.

rsync

There are far more efficient ways to mirror content than over HTTP. If you have shell access on a remote file server, you can use ***rsync*** instead. This lets you synchronise files and directories between systems much faster, without tying up public web server resources. You can run rsync over its own TCP port, or tunnel it over SSH for added security.

It can be quicker to run rsync rather than using a web mirroring utility, since rsync can transfer only changes within a file rather than the entire file. This can help when mirroring large files, such as compressed backups, or file archives. You can only use rsync on systems that run a public rsync server, or servers on which you have shell access.

Just about all versions of Linux and BSD include rsync, and it will also run on Windows and Mac OS X. You can download it at *http://samba.org/rsync/*.

Email

Email was the most popular application of the Internet before Tim Berners-Lee invented the World Wide Web, and is still extremely popular today. Most Internet users have at least one email address. Email's long history of development has made it cheap, fast, resilient, and almost universally supported. All modern computers, mobile phones, PDAs, and many other electronic devices have email support, making it nearly as ubiquitous as the telephone. In many organisations, email has even surpassed the telephone as the standard means of communication, where it is used for everything from internal information exchange to customer-facing sales and support. Many companies provide free email services (e.g. Microsoft, Yahoo, and Google).

Email Basics

Email senders use a program such as Thunderbird or Outlook (or even a Web browser) to create their message. When they click the "Send" button, the message is passed on to a local ***Mail Transfer Agent*** (***MTA***), often located in their company or at their Internet Service Provider. The MTA's task is to deliver the email to the recipient's MTA via the Internet.

Email is usually transferred across the Internet using a protocol known as the ***Simple Mail Transfer Protocol*** (***SMTP***). The current version is defined in RFC2821 (*http://www.rfc.net/rfc2821.html*). Earlier RFCs related to email date back as far as 1980, and primordial electronic messaging systems existed for years before that.

When the recipient's MTA receives an email, it often cannot be delivered directly to the recipient's computer. This could be because the recipient is not online, their mail client is not running, or they do not wish to store their mail on their own machine. Therefore, the recipient's MTA usually delivers the email to a storage service instead. This is known as a ***Mail Delivery Agent*** (***MDA***). The client can collect their mail from the MDA whenever they like, using protocols such as POP3 and IMAP. The program that ultimately retrieves the mail and displays it to the user is called the ***Mail User Agent*** (***MUA***).

Email Security

Unfortunately, SMTP is a simple protocol that does not offer much security. Due to its popularity and widespread support, it has proved impossible to replace. This lack of security has made it trivial for unscrupulous users to send vast amounts of ***unsolicited bulk email*** (a.k.a. ***spam***). The senders are often called ***spammers***. Most users do not want to receive spam, and it wastes their time and bandwidth.

It is in your best interests to limit the amount of junk mail that passes through your mail server. Your organisation is ultimately responsible for the email that emanates from your network. If you allow the entire world to relay mail through your email server, you will not only add to the growing spam problem, but you can easily get your mail server or IP address range blacklisted. If this happens, you will not be able to send or receive email at all, as major mail servers around the world will refuse to communicate with you. Acting as a spam relay may even violate the terms of service of your ISP, and could get you disconnected. If your bandwidth is charged by the byte, you are likely to receive a bill for all of the spam that came from your network before it was shut off!

There are four critical steps to keep in mind when running an Internet email service from your network. These steps are:

1. **Choose a good MTA.** There are many available, so you should choose one that is secure, fast, and flexible enough to support mail services at your organisation. (page **151**)
2. **Eliminate all open relays.** Users should be authenticated (by IP address, username and password, or some other credentials) before your server accepts the email for relaying. (page **166**)
3. **Implement anti-spam and anti-virus measures for inbound (and, if possible, outbound) email.** For inbound email, this will help reduce complaints from your users. For outbound email, this can help you to find malicious users and machines infected with viruses. (page **152**)

4. **Closely monitor the volume and profile of mail sent from your network.** While they can catch more than 99% of unwanted email traffic, no email filter is completely infallible. A sudden increase in email volume, or mail being sent at unusual hours, can indicate that spam is getting around your filter and leaving your network. (page **82**)

By following these four steps, you can go a long way towards reducing the bandwidth spent on handling email, and keeping your network off the blacklists. But since email is so widely abused, there are a few other subtle points to keep in mind. If your organisation uses dynamic IP space (page **37**), you may run into unexpected trouble with your mail server. Some organisations that compile blacklists also collect lists of known dynamic IP ranges. These IP lists are then published in the form of a ***DNS Black List*** (***DNSBL***) and can be used by large ISPs for rejecting mail. Since dynamic IPs are relatively anonymous, they are often havens for prolific spammers.

It is also common practice for large ISPs to block all traffic destined for port 25 (the SMTP port) originating from their dynamic IP space, in an effort to prevent outbound spam. Whenever possible, use a static IP address for your email server. If this is not possible, you should configure your email server to relay all outbound mail to the SMTP server at your ISP. This is required by some ISPs in order to track email abuse.

Choosing an MTA

On UNIX-like operating systems such as Linux and FreeBSD, there are four popular MTAs: ***Sendmail***, ***qmail***, ***Postfix***, and ***Exim***. All of these are quite powerful and reliable, but choosing the right one for your needs is vital, as there are some important differences between them.

- **Sendmail** (*http://www.sendmail.org/*) is the oldest MTA still in common use. When it was released in the early 1980s, it was responsible for delivering almost all of the e-mail sent over the Internet. Its popularity has declined due to its reputation for being difficult to configure, as well as the discovery of a number of critical security flaws over the years. Despite this, it is still included by default with many UNIX operating systems. If properly configured, Sendmail is efficient and has many useful features, but unless you are already familiar with the program or it has been set up for you already, you should probably consider one of the other, more easy-to-use MTAs. Sendmail is being completely re-written and the new version, known as Sendmail X, may resolve some of the issues with the original program.
- **qmail** (*http://www.qmail.org/*) was written specifically because of security concerns with Sendmail. It has an unusual design that has proven very successful at repelling attacks, and is also known for being very fast, stable, and minimal (that is, it has a relatively small set of features compared to the other

MTAs). The author of the program has not updated it since 1997, and the licensing conditions prevent any modifications from being officially released, although unofficial patches are available. Because of its unconventional nature, it is not recommended that you use qmail unless you are already familiar with it or are willing to put the effort into learning more about it. For more information, see *http://www.lifewithqmail.org/*.

- **Postfix** (*http://www.postfix.org/*) was developed by a security expert at IBM. It has a significantly better security record than Sendmail, and is considered easier to use than qmail. Like qmail, and unlike Sendmail and Exim, Postfix emphasises security over features. However, it has a more conventional design than qmail and its configuration is easier to understand. It is the only MTA besides Sendmail to support Sendmail's mail filters (***milters***). It is also well documented and has good community support. Postfix is very efficient in memory usage, speed, and bandwidth.
- **Exim** (*http://www.exim.org/*) was developed at Cambridge University (UK) before the release of qmail or Postfix, to provide them with a replacement for Sendmail. The configuration file is easy to read and understand, and the documentation is very good. Exim is extendable and easy to configure, making it simple to do clever and strange things with mail systems. The author has never claimed that Exim is secure, and early versions had many security problems. But since 2000 its security record has been quite good.

In summary, Postfix and Exim are likely to be the most appropriate choices for inexperienced users. Postfix seems to have a better security record, while Exim is more flexible. Both are relatively easy to use. Postfix may have better performance for larger sites. Exim is recommended for smaller sites, people new to mail systems, and sites that need strange/unusual mail configurations. For exceptionally large sites, qmail may provide better performance, but it comes with a very high learning curve.

If you are stuck with a Windows server, there are a few free mail servers that you can try. All of these include MTA and MDA functionality (in other words, they can receive, send, and store mail):

- **Mercury Mail** (*http://www.pmail.com/*) is a feature-rich mail server. It was designed to work with the Pegasus Mail client (which is an MUA), but it is also very standards-compliant and should work with most MUAs. Two versions are available: one for Novell NetWare and the other for Windows.
- **Macallan** (*http://macallan.club.fr/MMS*) runs on Windows XP and 2000. It provides all the basic mail server features, and includes built-in webmail. The free version supports up to 128 mailboxes.
- **MailEnable** (*http://www.mailenable.com/*) is a Windows mail server that supports SMTP and POP3. The free version does not support IMAP. It has a

mailing list manager that makes it easy to administer subscription-based distribution lists.

- **BaSoMail** (*http://www.baso.no/*) is a compact and easy-to-use SMTP/POP3 server for Windows. It does not support IMAP.

Securing your mail server

The war on spam is constantly evolving as developers find new and clever ways to outwit the spammers, and spammers find devious and subtle ways to outwit the developers. A strong email server implementation will include the following components:

- **A secured MTA installation**. The choice of MTA is up to you, but it should authenticate outbound email before sending to eliminate the possibility of open relays. Follow the installation instructions for your MTA to be sure that authentication is properly enabled (page **152**). You can test your own email server for holes using abuse.net's **Mail relay testing tool**, available at *http://www.abuse.net/relay.html* . Another popular testing tool is the ORDB at *http://www.ordb.org/submit/*, provided by the Open Relay Database.
- A spam filtering package, such as **SpamAssassin** or **DSPAM**. SpamAssassin provides a way to reduce, if not completely eliminate, unsolicited junk mail from your incoming email. It uses a scoring system based on local and network defined rules to identify messages which appear to be spam. It then adds headers to the message that identify the total spam "score," allowing the messages to be easily filtered by the user's mail client. DSPAM approaches the problem differently, by relying almost completely on automated language analysis and deobfuscation techniques instead of user-contributed rulesets and blacklists. While we use SpamAssassin for the examples in this book, both packages are open source and are worth considering. Spamassassin can be downloaded from *http://spamassassin.apache.org/*. DSPAM is available at *http://dspam.nuclearelephant.com/*.
- **ClamAV** (*http://www.clamav.net/*). Clam Antivirus is fast and robust open source antivirus package. It includes a multi-threaded daemon (for speed) and a simple command line scanner. Its virus definition files are automatically updated from the Internet, keeping it up to date as new viruses are discovered. When used in combination with an MTA, ClamAV allows you to scan for viruses and other harmful content embedded in email, before it is delivered to the user.
- **Amavisd-new** (*http://www.ijs.si/software/amavisd/*). This package acts as an interface between your MTA and content filters such as SpamAssassin and ClamAV. It works with every popular MTA including Postfix, Sendmail, Qmail, and Exim.
-

From the AMaViS website:

> *amavisd-new is a high-performance interface between mailer (MTA) and content checkers: virus scanners, and/or SpamAssassin. It is written in Perl for maintainability, without paying a significant price for speed. It talks to MTA via (E)SMTP or LMTP, or by using helper programs.*

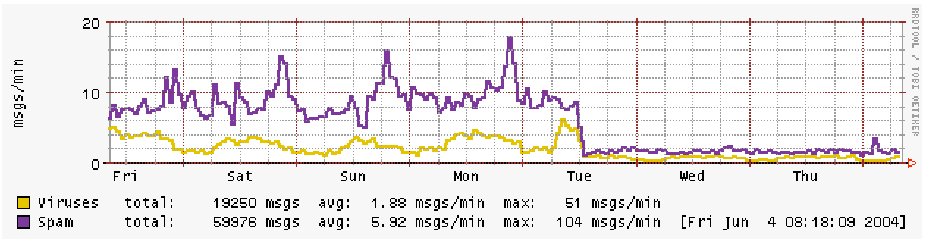
By combining amavisd-new with your content filters (SpamAssassin and ClamAV), it is possible to build a robust email solution capable of handling very large email loads, even on relatively modest hardware. All of these components are open source.

You should combine all of these, if possible, to defend your mail server and your users against spam and viruses.

Greylisting

While blacklists provide explicit criteria used to refuse mail, and whitelists explicitly permit mail, ***greylists*** provide the functionality of both. Each time a given mailbox receives an email from an unknown IP address, the mail is automatically rejected with a "try again later" message. This happens at the SMTP layer, and is transparent to the sender or recipient. Since most spamming software is quite simple and does not comply with established RFCs, will not try again later and will simply drop the message. This can provide an immediate and dramatic reduction in spam without using content filtering.

If the MTA tries again after the greylisting period (typically 30 minutes), a combination of the sender's email address, IP address, and the recipient's email address is whitelisted for 3 days. If email from the same sender and IP address is sent to the same recipient is within 3 days, it is sent through without delay and the whitelisting period is reset for a further 3 days.



```
■ Viruses   total:   19250 msgs  avg:  1.88 msgs/min  max:   51 msgs/min
■ Spam      total:   59976 msgs  avg:  5.92 msgs/min  max:  104 msgs/min  [Fri Jun  4 08:18:09 2004]
```

Figure 4.11: Greylisting can show an immediate improvement in deterring spam, while consuming modest resources.

While some users report very good results using greylists, others have had mixed results.

The author of one package, exim-greylist, writes on his website:

> *I no longer maintain this because I no longer use greylisting on my server, it proved to be little efficient when having a lot of other SPAM checks and often deferred valid email for me and my users. I'm leaving this here for reference only.*

If you decide to implement greylisting, you should evaluate its performance to determine how effectively it combats the sort of spam that is directed at your network.

Greylisting must be implemented in the MTA, and is typically achieved by using third party software. The following is a list of freely available, open source greylisting software packages for various MTAs.

For **Postfix**:

- GLD (*http://www.gasmi.net/gld.html*) - Greylist server with MySQL database by Salim Gasmi.
- SQLgrey (*http://sqlgrey.sourceforge.net/*) - Greylist policy server with auto-whitelisting in Perl with support for PostgreSQL, MySQL and SQLite storage by Lionel Bouton.
- GPS (*http://mimo.gn.apc.org/gps/*) - Greylist policy server in C++ using MySql, Postgres, or SQLite by Michael Moritz.
- PostGrey (*http://isg.ee.ethz.ch/tools/postgrey/*) - Greylist policy server in Perl by David Schweikert.
- Policyd (*http://policyd.sourceforge.net/*) - Policy server in C which provides greylisting, sender (envelope, SASL, or host/ip) based throttling (messages and/or volume per hour) and spam traps by Cami Sardinha.
- TumGreySPF (http://www.tummy.com/Community/software/tumgreyspf/) - Greylisting and SPF policy server by Sean Reifschneider. This uses the file system (instead of a database file) for storing greylist data and configuration information.

For **Sendmail**:

- milter-greylist (*http://hcpnet.free.fr/milter-greylist/*) - milter-greylist is a stand-alone milter written in C that implements the greylist filtering method.
- SMFS (*http://smfs.sourceforge.net/*) - Lightweight, fast, and reliable Sendmail milters for SPAM and virus filtering. SMFS includes support for greylists, SPF, SpamAssassin, and other filtering techniques.

For **Exim**:

- greylistd (*http://www.tldp.org/HOWTO/Spam-Filtering-for-MX/exim.html*) - a Python implementation by Tor Slettnes.
- Direct support in Exim configuration file (no external programs) using MySQL (*http://theinternetco.net/projects/exim/greylist*) or Postgres (*http://raw.no/personal/blog/tech/Debian/2004-03-14-15-55_greylisting*)
- exim-greylist (*http://johannes.sipsolutions.net/Projects/ex*), advanced greylisting including statistics and manual whitelisting capability with Exim 4 and MySQL. No longer maintained.

For **Qmail**:

- qgreylist (*http://www.jonatkins.com/page/software/qgreylist*)
- denysoft_greylist (*http://www.openfusion.com.au/labs/dist/denysoft_greylist*)

DomainKeys

DomainKeys are designed not only to verify the authenticity of an email sender, but also to ensure that the email has not been tampered with in transit. Under the DomainKeys system, the sender generates two cryptographic keys: the public key and the private key. Anything encoded with the public key can be decoded using the private key, and vice versa. The public key is published in the sender's DNS records.

The sender signs all of their messages with a signature that has been generated based on the contents of the email, and encoded using the sender's private key. When the email arrives, the receiver can look up the public key in the DNS record for the domain from which the email claims to be. If the sender is authentic and really has access to the private key, then the public key can decode the encrypted signature, and hence the receiver can be confident that the sender is authentic and that the email has not been tampered with in transit.

More details on how to set up DomainKeys with various MTAs can be found at *http://antispam.yahoo.com/domainkeys*. Yahoo! is one of the chief proponents of DomainKeys.

Sender Policy Framework (SPF)

Sender Policy Framework (***SPF***) is designed to fight email forgery, which is often used in scam emails. SPF allows you to verify that mail apparently from a particular domain (say, a financial institution or government office) was sent from an authorised mail server. SPF verifies the path that email took to arrive at your server, and can discard email that originated at unauthorised MTAs before the message is transmitted, thus saving bandwidth and reducing spam. It works by publishing DNS records that indicate valid mail servers for your domain. Before mail is accepted, the receiving MTA does a DNS lookup to verify that the sender is an authorised email server for the domain. If not, it terminates the connection. If SPF records for the domain match the sender's MTA, the mail is accepted.

While this can do little to help with email sent from bogus addresses (or domains that do not publish SPF records), it promises to reduce email forgeries as more organisations implement it. SPF documentation is available from the main site at *http://www.openspf.org/*.

As with greylisting and domain keys, SPF is implemented in your MTA. The following is a list of packages that provide SPF support for various MTAs.

For **Postfix**:

- libspf2-1.2.x Patch (*http://www.linuxrulz.org/nkukard/postfix/*) - Patch to Postfix to allow native SPF support by Nigel Kukard.
- TumGreySPF (*http://www.tummy.com/Community/software/tumgreyspf/*) - Greylisting and SPF policy server by Sean Reifschneider. This uses the file system (instead of a database file) for storing greylist data and configuration information.

For **Sendmail**:

- SMFS (http://smfs.sourceforge.net/) - Lightweight, fast and reliable Sendmail milters for SPAM and virus filtering.
- SPFMilter (http://www.acme.com/software/spfmilter/)

Resources

- DomainKeys entry on Wikipedia, (*http://en.wikipedia.org/wiki/DomainKeys*)
- Firewalls in FreeBSD,
  http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/firewalls.html
- MTA comparisons:
  http://shearer.org/MTA_Comparison
  http://www.python.org/cgi-bin/faqw-mm.py?req=show&file=faq04.002.htp

- Netfilter Packet Filtering HOWTO: *http://www.netfilter.org/documentation/HOWTO/packet-filtering-HOWTO-7.html*
- Netfilter documentation: *http://www.netfilter.org/documentation/*
- Network Address Translation HOWTO : *http://www.netfilter.org/documentation/HOWTO/NAT-HOWTO.html*
- Squid documentation, *http://www.visolve.com/squid/*
- Squid FAQ, *http://www.squid-cache.org/Doc/FAQ/FAQ-17.html*
- Squid Wiki, *http://wiki.squid-cache.org/*
- Wessels, Duane. *Squid: The Definitive Guide*. O'Reilly Media (2004). *http://squidbook.org/*

5 Troubleshooting

How you establish the support infrastructure for your network is as important as what type of equipment you use. Once your network is connected to the Internet, or opened up to the general public, considerable threats may come from the Internet, or from your users themselves. These threats can range from the benign to the outright malevolent, but all will have an impact on your network if it is not properly configured.

This chapter will show you some common problems that are frequently encountered on modern networks, as well as simple methods for quickly determining the source of the trouble. But before you jump straight into the technical details, it is a good idea to get into the mindset of a good troubleshooter. You will solve your problems much faster if you cultivate an attitude of lateral thinking and problem solving. If networks could be fixed by simply trying random tactics without understanding the problem, then we would program a computer to do it for us.

Proper troubleshooting technique

No troubleshooting methodology can completely cover all bandwidth problems you will encounter when working with networks. Having a clear methodology for both preparing and responding to the problems that you encounter will keep you working the right direction.

But often, problems come down to one of a few common mistakes. Here are a few simple points to keep in mind that can get your troubleshooting effort working in the right direction.

Preparing for problems

- **Make regular backups.** Over time your network configuration will grow and expand to suit your particular network. Remembering the intricate details will become impossible making it very difficult to reproduce the same configuration if it is lost. Making regular backups ensures that you can rebuild your configuration from scratch if required. Having multiple backups means you can roll back to a previous known working state if a configuration change goes awry.
- **Disaster plan.** Technology is not always as reliable as we hope, and it is a certainty that at some point major problems will strike your network. By planning for these and having a procedure in place for dealing with them you will be in a far better situation when the lights go off!
- **Fallback network mode.** It is useful to prepare a basic network configuration state, which only allows a minimum set of services on a network. When a problem occurs which stops the network from functioning effectively you can implement this fallback mode, allowing others to use essential services whilst you are troubleshooting the problem.

Responding to a problem

- **Don't panic.** Don't panic, inform your users of the problem, and set about solving it in a methodical and guided manner.
- **Understand the problem.** If you are troubleshooting a system, that means that it was working at one time, and probably very recently. Before jumping in and making changes, survey the scene and assess exactly what is broken. If you have historical logs or statistics to work from, all the better. Be sure to collect information first, so you can make an informed decision before making changes.
- **Is it plugged in?** This step is often overlooked until many other avenues are explored. Plugs can be accidentally (or intentionally) unplugged very easily. Is the lead connected to a good power source? Is the other end connected to your device? Is the power light on? It may sound silly, but you will feel even sillier if you spend a lot of time checking out an antenna feed line only to realise that the AP was unplugged the entire time. Trust me, it happens more often than most of us would care to admit.
- **What was the last thing changed?** If you are the only person with access to the system, what is the last change you made? If others have access to it, what is the last change they made and when? When was the last time the system worked? Often, system changes have unintended consequences that may not be immediately noticed. Roll back that change and see what effect it has on the problem.

- **Make a backup.** This applies before you notice problems, as well as after. If you make a complicated software change to a system, having a backup means that you can quickly restore it to the previous settings and start again. When troubleshooting very complex problems, having a configuration that "sort-of" works can be much better than having a mess that doesn't work at all (and that you can't easily restore from memory).
- **The known good.** This idea applies to hardware, as well as software. A ***known good*** is any component that you can replace in a complex system to verify that its counterpart is in good, working condition. For example, you may carry a tested Ethernet cable in a tool kit. If you suspect problems with a cable in the field, you can easily swap out the suspect cable with the known good and see if things improve. This is much faster and less error-prone than re-crimping a cable, and immediately tells you if the change fixes the problem. Likewise, you may also pack a backup battery, antenna cable, or a CD-ROM with a known good configuration for the system. When fixing complicated problems, saving your work at a given point lets you return to it as a known good, even if the problem is not yet completely solved.
- **Change one variable at a time.** When under pressure to get a failed system back online, it is tempting to jump ahead and change many likely variables at once. If you do, and your changes seem to fix the problem, then you will not understand exactly what led to the problem in the first place. Worse, your changes may fix the original problem, but lead to more unintended consequences that break other parts of the system. By changing your variables one at a time, you can precisely understand what went wrong in the first place, and be able to see the direct effects of the changes you make.
- **Do no harm.** If you don't fully understand how a system works, don't be afraid to call in an expert. If you are not sure if a particular change will damage another part of the system, then either find someone with more experience or devise a way to test your change without doing damage. Putting a penny in place of a fuse may solve the immediate problem, but it may also burn down the building.

A basic approach to a broken network

It happens all the time. Suddenly, the network isn't working at all. What do you do? Here are some basic checks that will quickly point to the cause of the problem.

First make sure the problem is not just with the one web server you want to contact. Can you open other websites such as *www.google.com*? If you can open popular web sites but not the one you requested, the problem is likely with the site itself, or with the network between you and the other end. If your web browser cannot load pages from the Internet, next try to browse to a server on the local network (if any). If local sites are shown quickly, then that may indicate a problem with the Internet connection or the proxy server. If not, then you most likely have a problem with the local network.

If you suspect a problem with the proxy server, try accessing information with a different program, such as an email client. If you can send and receive email, but web browsing still doesn't work, then this may further indicate problems with the proxy server.

Your web browser and mail client can only provide so much information. If nothing seems to be working at all, it's time to switch to a more useful diagnostic tool such as **ping**.

Try pinging a well know server such as *google.com*.

```
$ ping www.google.com
PING www.l.google.com (66.102.9.99) 56(84) bytes of data.
64 bytes from 66.102.9.99: icmp_seq=1 ttl=243 time=30.8 ms
64 bytes from 66.102.9.99: icmp_seq=2 ttl=243 time=31.6 ms
64 bytes from 66.102.9.99: icmp_seq=3 ttl=243 time=30.9 ms

--- www.l.google.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2016ms
rtt min/avg/max/mdev = 30.865/31.176/31.693/0.395 ms
```

This indicates that your network connection is working just fine, and the problem is very likely with your proxy (or your web browser's proxy settings). Sometimes, **ping** will hang while waiting for a DNS reply:

```
$ ping www.google.com
```

In this case, DNS resolution isn't working. Without DNS, just about everything else on the network will seem to be broken. Check your local machine settings to be sure that it is using the correct DNS server. If you can't ping a server using the domain name, the next step would be to ping a known IP address - 4.2.2.2 is an easy to remember address.

```
$ ping 4.2.2.2
PING 4.2.2.2 (4.2.2.2) 56(84) bytes of data.
64 bytes from 4.2.2.2: icmp_seq=1 ttl=247 time=85.6 ms
64 bytes from 4.2.2.2: icmp_seq=2 ttl=247 time=86.3 ms
64 bytes from 4.2.2.2: icmp_seq=3 ttl=247 time=84.9 ms
64 bytes from 4.2.2.2: icmp_seq=4 ttl=247 time=84.8 ms

--- 4.2.2.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3012ms
rtt min/avg/max/mdev = 84.876/85.436/86.330/0.665 ms
```

If you can ping an IP address but not a domain name, then the network is fine but your computer is unable to convert the domain name (*www.google.com*) into an IP address (66.102.9.99). Check to make sure that your DNS server is running and is reachable.

If you can't ping an Internet IP address, then it's a good idea to make sure that your local network connection is still working. Does your computer have a valid IP address? Use **`ifconfig`** on UNIX or **`ipconfig`** on Windows to make sure your IP settings are correct.

If you don't have an IP address then you are definitely not connected to the Internet. Check the cables from your computer (or the wireless settings if using wireless). Also check that your DHCP server is up and running, if you use DHCP. If you have an IP address but it is incorrect, then there are only two possibilities. Either your machine is using the wrong settings, or there is a rogue DHCP server on the local network (page **170**). Either change your local settings or track down the bad DHCP server.

If you do have a valid IP address, try pinging the gateway's IP address.

```
$ ping 192.168.0.1
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.
64 bytes from 192.168.0.1: icmp_seq=1 ttl=64 time=0.489 ms
64 bytes from 192.168.0.1: icmp_seq=2 ttl=64 time=0.496 ms
64 bytes from 192.168.0.1: icmp_seq=3 ttl=64 time=0.406 ms
64 bytes from 192.168.0.1: icmp_seq=4 ttl=64 time=0.449 ms

--- 192.168.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2999ms
rtt min/avg/max/mdev = 0.406/0.460/0.496/0.035 ms
```

If you can't ping your gateway then the problem is definitely in your local network - maybe the switch or router needs to be restarted. Check all the cables (both network cables and power cables).

If you can ping your gateway, then you should next check the Internet connection. Maybe there is a problem upstream. One good test is to log into your gateway router and try to ping the gateway at your ISP. If you cannot ping your ISP's gateway, then the problem is with your Internet connection. If you can ping your ISP's gateway but no other Internet hosts, then the problem may exist beyond your ISP. It's probably a good idea to phone your ISP and check if they have a problem.

Is everything still not working? Then it's time to roll up your sleeves and get to work. You should reread the **Troubleshooting Technique** section (page **159**) and settle down for some slow and methodical work, checking each part of your network bit by bit.

Common symptoms

It can be difficult at times to diagnose a problem by simply looking at utilisation graphs or running spot check tools like **`ping`**. This is why establishing a baseline is so important. If you know what your network should "look like," then you can usually find the problem by looking for anything out of the ordinary. Here are examples of common, but often subtle problems that may cause anything from performance degradation to complete network outages.

Automatic updates

Automatic updates for virus scanners, spyware scanners, and Microsoft Windows are very important to a healthy, virus-free network environment. However, most auto-update systems can cause very heavy network usage at inopportune times, especially when a new, critical patch is made available. Many auto-update systems make use of HTTP to download updates, so keep an eye out for specific auto-update URLs. It may be worth compiling a list of your own auto-update URLs by monitoring your logs, and checking widely-installed software. Some common URLs are *http://update.microsoft.com/* and *http://update.adobe.com/*.

You may want to configure your auto-update software to download updates outside of business hours, as some software allows you to set at which time to check and download updates. You may also want to research if your vendor offers products to cache the automatic updates on a server on your network. This way, you only need to transfer the update once over the Internet, and the patch or signature can be quickly sent to all hosts using the much faster local network.

Sometimes vendors do not offer cost effective solutions, or such solutions may not even exist. In this case, you may want to consider mirroring these sites using split horizon DNS (page **212**). Sometimes, you may be able to cache these updates on your web proxy.

Simply blocking the update site on the proxy server is not a good solution because some update services (such as Windows automatic updates) will simply retry more aggressively. If all workstations do that at once, it places a heavy load on the proxy server. The extract below is from the proxy log (Squid access log) where this was done by blocking Microsoft's cabinet (**`.cab`**) files. Much of the Squid log was full of lines like this:

```
2003.4.2 13:24:18 192.168.1.21 http://windowsupdate.microsoft.com/ident.cab
*DENIED* Banned extension .cab GET 0
```

While this may be tolerable for a few PC clients, the problem grows significantly as hosts are added to the network. Rather than forcing the proxy server to serve requests that will always fail, it makes more sense to redirect the Software Update clients to a local update server.

Spyware

Often when users browse sites, these sites will install little programs on the users' PCs. Some of these programs will collect information from the PC and send it to a site somewhere on the Internet. By itself this might not much of a threat to your Internet connection, but if you have hundreds of machines doing this it can be a problem as they will quickly fill your bandwidth and perhaps flood your proxy server. The symptom of this particular problem would be a slow client machine and lots of similar requests for machines in your proxy log files. It is very important to run anti-virus and anti-spyware software on all machines, and to ensure that it is updated regularly, to counter this threat.

P2P

Peer-To-Peer applications are used for the sharing of media, software, and other content between groups of users. They are often left running, and can use up bandwidth downloading and uploading data even when the user believes it to be closed.

Detecting P2P software can be difficult, other than monitoring the volume of traffic. P2P applications often use protocols designed to bypass firewalls and proxies, disguising the data as normal web traffic, for example.

It is possible to detect P2P traffic by monitoring the number of connections that a client machine is making to a remote server. P2P software often tries to make multiple connections to speed up transfers. Alternative techniques involve the detailed inspection of packets passing through a server and marking them as P2P traffic where detected. This is possible by using an application layer firewall such as L7-filter (*http://l7-filter.sourceforge.net/*). However, this kind of filtering is processor intensive and not 100% foolproof.

It is important that P2P traffic is not blocked, only rate limited heavily. Most software will attempt to jump ports and disguise itself more effectively if a block is detected. By rate limiting the traffic, the traffic can be kept under control.

Email

Monitoring the baseline percentage of email utilisation on your connection will give you a good understanding of typical email usage. If the volume of email traffic rises significantly then further inspection will be required to ascertain why.

Users restricted in other areas may attempt to use email to bypass the restrictions, perhaps sending large files. Limiting the size of email attachments can help alleviate the load.

Viruses will often use email to deliver their payload, and rising email volume can be a good indicator of a virus problem on the network.

Open email relay hosts

An SMTP server which allows a computer from any address to send email to any other address is called an ***open relay***. This kind of server can be compromised and made use of by spammers to send large quantities of email, thus consuming large amounts of bandwidth. They do this to hide the true source of the spam, and avoid getting caught. This kind of behaviour will show up in analyses of network communications, with increases in uses of email. However, the best policy is to ensure that your SMTP servers will only accept email to or from domains you control, avoiding the problem from ever occurring.

To test for an open relay host, the following test should be carried out on your mail server (or on the SMTP server that acts as a relay host on the perimeter of the campus network). Use **`telnet`** to open a connection to port 25 of the server in question (with some Windows versions of telnet, it may be necessary to type **`set local_echo`** before the text is visible):

```
telnet mail.uzz.ac.zz 25
```

Then, if an interactive command-line conversation can take place (for example, as follows), the server is an open relay host:

```
MAIL FROM: spammer@waste.com
250 OK - mail from <spammer@waste.com>
RCPT TO: innocent@university.ac.zz
250 OK - rcpt to spammer@waste.com
```

Instead, the reply after the first MAIL FROM should be something like:

```
550 Relaying is prohibited.
```

An online tester is available at sites such as *http://www.ordb.org/*. There is also information about the problem at this site. Since bulk emailers have automated methods to find such open relay hosts, an institution that does not protect its mail systems is almost guaranteed to be found and abused. Configuring the mail server not to be an open relay consists of specifying the networks and hosts that are allowed to relay mail through them in the MTA (e.g., Sendmail, Postfix, Exim, or Exchange). This will likely be the IP address range of the campus network.

Email forwarding loops

Occasionally, a single user making a mistake can cause a problem. For example, a user whose university account is configured to forward all mail to her Yahoo account. The user goes on holiday. All emails sent to her in her absence are still forwarded to her Yahoo account, which can grow to only 2 MB. When the Yahoo account becomes full, it starts bouncing the emails back to the university account, which immediately forwards it back to the Yahoo account. An email loop is formed that might send hundreds of thousands of emails back and forth, generating massive traffic and crashing mail servers.

There are features of mail server programs that can recognise loops. These should be turned on by default. Administrators must also take care that they do not turn this feature off by mistake, or install an SMTP forwarder that modifies mail headers in such a way that the mail server does not recognise the mail loop.

Open proxies

A proxy server should be configured to accept only connections from the university network, not from the rest of the Internet. This is because people elsewhere will connect and use open proxies for a variety of reasons, such as to avoid paying for international bandwidth. The way to configure this depends on the proxy server you are using. For example, you can specify the IP address range of the campus network in your **`squid.conf`** file as the only network that can use Squid (page **272**). Alternatively, if your proxy server lies behind a border firewall, you can configure the firewall to only allow internal hosts to connect to the proxy port.

Programs that install themselves

There are programs that automatically install themselves from the Internet and then keep on using bandwidth - for example, the so-called Bonzi-Buddy, the Microsoft Network, and some kinds of worms. Some programs are spyware, which keep sending information about a user's browsing habits to a company somewhere on the Internet. These programs are preventable to some extent by user education and locking down PCs to prevent administrative access for normal users. In other cases, there are software solutions to find and remove these problem programs, such as Spychecker (*http://www.spychecker.com/*), Ad-Aware (*http://www.lavasoft.de/*), or xp-antispy (*http://www.xp-antispy.de/*).

Programs that assume a high bandwidth link

In addition to Windows updates, many other programs and services assume that bandwidth is not a problem, and therefore consume bandwidth for reasons the user might not predict. For example, anti-virus packages (such as Norton Antivirus), periodically update themselves automatically and directly from the Internet. It is better if these updates are distributed from a local server.

Other programs, such as the RealNetworks video player, automatically download updates and advertisements, as well as upload usage patterns back to a site on the Internet. Innocuous looking applets (like Konfabulator and Dashboard widgets) continually poll Internet hosts for updated information. These can be low bandwidth requests (like weather or news updates), or very high bandwidth requests (such as webcams). These applications may need to be throttled or blocked altogether.

The latest versions of Windows and Mac OS X also have a time synchronisation service. This keeps the computer clock accurate by connecting to time servers on the Internet. It is better to install a local time server and distribute accurate time from there, rather than to tie up the Internet link with these requests.

Windows traffic on the Internet link

Windows computers communicate with each other via ***NetBIOS*** and ***Server Message Block*** (***SMB***). These protocols work on top of TCP/IP or other transport protocols. It is a protocol that works by holding elections to determine which computer will be the ***master browser***. The master browser is a computer that keeps a list of all the computers, shares, and printers that you can see in ***Network Neighborhood*** or ***My Network Places***. Information about available shares are also broadcast at regular intervals.

The SMB protocol is designed for LANs and causes problems when the Windows computer is connected to the Internet. Unless SMB traffic is filtered, it will also tend to spread to the Internet link, wasting the organisation's bandwidth. The following steps might be taken to prevent this:

- **Block outgoing SMB/NetBIOS traffic on the perimeter router or firewall.** This traffic will eat up Internet bandwidth, and worse, poses a potential security risk. Many Internet worms and penetration tools actively scan for open SMB shares, and will exploit these connections to gain greater access to your network. To block this traffic, you should filter TCP and UDP ports 135-139, and TCP port 445.
- **Install ZoneAlarm on all workstations (not the server).** A free version can be found at *http://www.zonelabs.com/*. This program allows the user to determine which applications can make connections to the Internet and which ones cannot. For example, Internet Explorer needs to connect to the Internet, but Windows Explorer does not. ZoneAlarm can block Windows Explorer from doing so.

- **Reduce network shares.** Ideally, only the file server should have any shares. You can use a tool such as SoftPerfect Network Scanner (from *http://www.softperfect.com/*) to easily identify all the shares in your network.

Streaming media / Voice over IP

Streaming audio and video comes in many forms, Internet radio and video are popular on the web, whilst people can communicate through video and audio using Voice over IP and instant messaging tools. All these types of media streaming use large amounts of bandwidth, and can reduce availability for other services. Many of these services use well known ports, which can be detected and limited or blocked by firewalls.

Service	TCP	UDP
Realtime Streaming Protocol (RTSP) - for Quicktime 4, Real Video etc.	554	5005
Realtime Transport Protocol (RTP) - used by iChat for audio & video		16384-16403
Real Audio & Video	7070	6970-7170
Windows Media	1755	1755
Shoutcast Audio	8000	
Yahoo Messenger (voice)	5000-5001	5000-5010
AIM (AOL Instant Messenger) (video)	1024-5000	1024-5000
Yahoo Messenger (video)	5100	
Windows Messenger (voice)		2001-2120, 6801, 6901
MSN file transfers	6891-6900	
MSN Messenger (voice)	6901	6901

Your main line of defense is user education, as most users rarely consider bandwidth as a limited resource. If streaming continues to be a problem, you may want to consider traffic shaping or blocking of these ports.

Sometimes streaming is required and may be part of your organisation's vision in multimedia services. In this case, you may want to research using multicast as a more cost effective way of video distribution. When using multicast properly, streams are only sent to those who request it, and any additional requests for the same feed will not result in any increase in bandwidth.

Most streaming can be tunneled through a proxy server, so here again an authenticated proxy server (or firewall) is your best defense.

Skype and other VoIP services can be difficult to block, as they use various techniques to bypass firewalls. You can block the SIP port, which is UDP port 5060 for many VoIP clients, but almost all VoIP traffic is sent on randomized high UDP ports. An application layer firewall (such as l7-filter, *http://l7-filter.sourceforge.net/*) can help detect and filter it.

Denial of Service

Denial of Service (***DoS***) attacks occur when an attacker sends a large number of connection requests to a server, flooding it and effectively disabling it. This will show up in your firewall logs, but by this point it will be too late. The only effective defense is to cut off the traffic further upstream; doing this will require the cooperation of your ISP.

Rogue DHCP servers

A misconfigured DHCP server, either by accident or intentionally malicious, can wreak havoc on a local area network. When a host sends a DHCP request on the local network, it accepts whichever response it receives the fastest. If the rogue DHCP server hands an incorrect address faster than your own DHCP server, it can potentially blackhole some of your clients.

Most of the time, a rogue DHCP server is either a misconfigured server or wireless router. Rogue DHCP servers are difficult to track down, but here are some symptoms to look for:

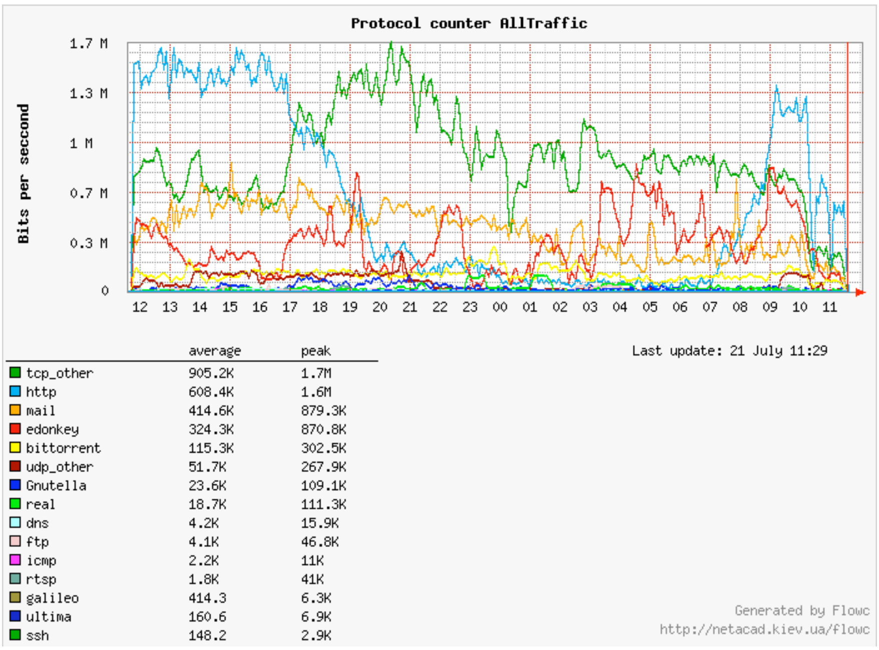
- Clients with improper IP addresses, netmasks, or gateways, even though your DHCP server is configured correctly.
- Some clients can communicate on the network, others cannot. Different IP addresses are being assigned to hosts on the same network.
- While sniffing network traffic, you see a DHCP response from a server IP address that you do not recognise.

Once you have determined the rogue DHCP server's MAC address from a packet trace, you can then make use of various layer 2 tracing techniques to determine the location of the rogue DHCP server, and isolate it.

There are several ways to prevent rogue DHCP servers from appearing on your network. First, educate your users on the dangers of misconfiguring or enabling DHCP services on your local LAN. Windows and UNIX systems engineers and users setting up access points on your local LAN should be careful not to place such a service on the local LAN. Second, some switching hardware platforms have layer 2 filtering capabilities to block DHCP responses from network interfaces that should never be connected to a DHCP server. On Cisco switching platforms, you may want to use the "dhcp snooping" feature set to specify trusted interfaces were DHCP responses can be transmitted. Apply these to server access ports and all uplink ports on your switch fabric.

Port analysis

There are several programs around that graphically display for you the network ports that are active. You can use this information to identify which ports need to be blocked at your firewall, proxy or router. However, some applications (like peer-to-peer programs) can masquerade on other ports, rendering this technique ineffective. In this case you would need a deep packet reader such as BWM Tools to analyse the application protocol information regardless of the port number. Figure 5.1 shows a graph from a protocol analyser known as FlowC.



	average	peak
🟩 tcp_other	905.2K	1.7M
🟦 http	608.4K	1.6M
🟧 mail	414.6K	879.3K
🟥 edonkey	324.3K	870.8K
🟨 bittorrent	115.3K	302.5K
🟫 udp_other	51.7K	267.9K
🟦 Gnutella	23.6K	109.1K
🟩 real	18.7K	111.3K
⬜ dns	4.2K	15.9K
⬜ ftp	4.1K	46.8K
🟪 icmp	2.2K	11K
⬛ rtsp	1.8K	41K
🟫 galileo	414.3	6.3K
🟦 ultima	160.6	6.9K
🟩 ssh	148.2	2.9K

Figure 5.1: Traffic utilisation broken down by protocol.

Browser prefetch

Some web browsers support a "prefetch" facility, which makes the browser download links on a web page before they are clicked on by the user. This functionality means that links can be displayed immediately, since the download has already taken place in the background. Browser prefetching will fetch many pages that the user will never view, thus consuming larger amounts of bandwidth than the user would otherwise require. Other than a marked increase in bandwidth use, this kind of behaviour is very difficult to detect. The only real response to this problem is to educate users, and explain that these tools can absorb large quantities of network bandwidth.

Benchmark your ISP

It is important to be sure that your Internet Service Provider has provided for you the level of service that you are paying for. One method of checking this is to test your connection speed to locations around the world. A list of servers around the world can be found at *http://www.dslreports.com/stest* . Another popular speed tester is *http://speedtest.net/*.

It is important to note that these tests have limitations. The tests are impacted by network conditions, both locally and across the entire route, at a particular moment in time. To obtain a full understanding, multiple tests should be run at different times of day, and with an understanding of local network conditions at the time.

Large downloads

A user may start several simultaneous downloads, or download large files such as 650MB ISO images. In this way, a single user can use up most of the bandwidth. The solutions to this kind of problem lie in training, offline downloading, and monitoring (including real-time monitoring, as outlined in chapter six). Offline downloading can be implemented in at least two ways:

- At the University of Moratuwa, a system was implemented using URL redirection. Users accessing *ftp://* URLs are served a directory listing in which each file has two links: one for normal downloading, and the other for offline downloading. If the offline link is selected, the specified file is queued for later download and the user notified by email when the download is complete. The system keeps a cache of recently downloaded files, and retrieves such files immediately when requested again. The download queue is sorted by file size. Therefore, small files are downloaded first. As some bandwidth is allocated to this system even during peak hours, users requesting small files may receive them within minutes, sometimes even faster than an online download.

- Another approach would be to create a web interface where users enter the URL of the file they want to download. This is then downloaded overnight using a ***cron job*** or scheduled task. This system would only work for users who are not impatient, and are familiar with what file sizes would be problematic for download during the working day.

Large uploads

When users need to transfer large files to collaborators elsewhere on the Internet, they should be shown how to schedule the upload. In Windows, an upload to a remote FTP server can be done using an FTP script file, which is a text file containing FTP commands, similar to the following (saved as **`c:\ftpscript.txt`**):

```
open ftp.ed.ac.uk
gventer
mysecretword
delete data.zip
binary
put data.zip
quit
```

To execute, type this from the command prompt:

```
ftp -s:c:\ftpscript.txt
```

On Windows NT, 2000 and XP computers, the command can be saved into a file such as **`transfer.cmd`**, and scheduled to run at night using the Scheduled Tasks (Start -> Settings -> Control Panel -> Scheduled Tasks). In Unix, the same can be achieved by using **`at`** or **`cron`**.

Users sending each other files

Users often need to send each other large files. It is a waste of bandwidth to send these via the Internet if the recipient is local. A file share should be created on a local file server, where a user can put the large file for others to access.

Alternatively, a web front-end can be written for a local web server to accept a large file and place it in a download area. After uploading it to the web server, the user receives a URL for the file. He can then give that URL to his local or international collaborators, and when they access that URL they can download it. This is what the University of Bristol has done with their ***FLUFF*** system. The University offers a facility for the upload of large files available from *http://www.bristol.ac.uk/fluff/*. These files can then be accessed by anyone who has been given their location. The advantage of this approach is that users can give external users access to their files, whereas the file share method can work only for users within the campus network. A system like this can easily be implemented as a CGI script using Python and Apache.

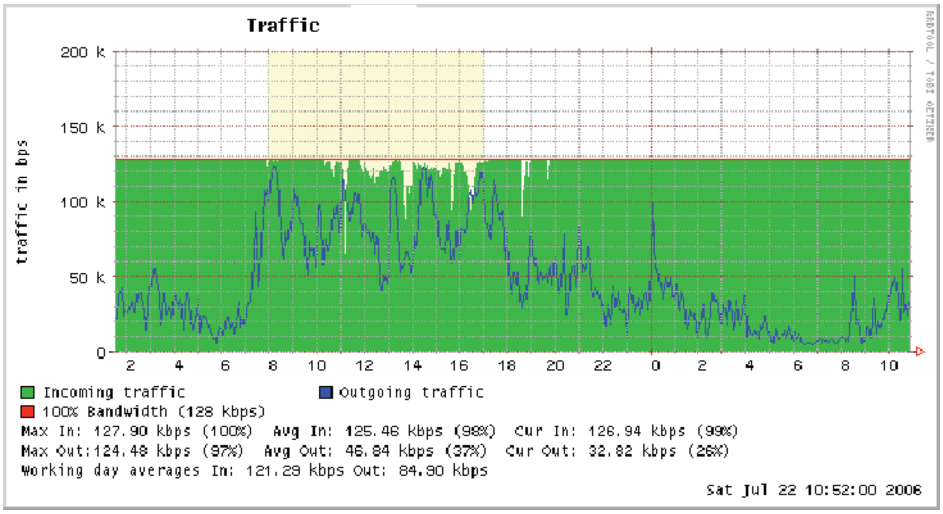
Viruses and worms

Viruses are self-replicating computer programs that spread by copying themselves into (or ***infecting***) other files on your PC. Viruses are just one type of ***malware*** (or malicious software). Other types include ***worms*** and ***trojan horses***. Often, the term "virus" is used in the broader sense to include all types of malware. Network viruses spread by using popular network programs and protocols, such as SMTP, to spread themselves from one computer to another. Worms are similar to viruses in that they are spread from machine to machine, but worms typically do not have a malicious payload. While the goal of a virus is to steal data or damage computers, worms are simply intent on replicating as fast and as widely as possible. Trojan horses include software that masquerades as something useful (such as a utility or game), but secretly installs malware on your machine. Viruses and worms sometimes use trojan horses as a ***vector*** for infection. If a user can be tricked into double-clicking an email attachment or other program, then the programs can infect the user's machine.

Viruses can saturate a network with random traffic that can slow down a local area network and bring an Internet connection to a standstill. A virus will spread across your network much like the common flu will spread around a city. A network virus will typically start on a single PC, and then scan the entire local network looking for hosts to infect. It is this traffic that typically will kill the network. A single infected PC may not cause a big problem for the network, but as the number of infected PCs grows, the network traffic will grow exponentially. Another strategy viruses use is to send themselves through email. This type of virus will typically attempt to email itself to everyone in your address book with the intention of infecting them with the virus.

Recently, more diabolical viruses have been found that create vast ***bot networks***. These are used to send spam, perform DDoS attacks, or simply log traffic and send it back to a central location. These bots often use IRC as a control channel, where it receives further instructions. If your organisation does not rely on IRC for communication, it is prudent to filter IRC traffic using a firewall, proxy server, or with l7-filter.

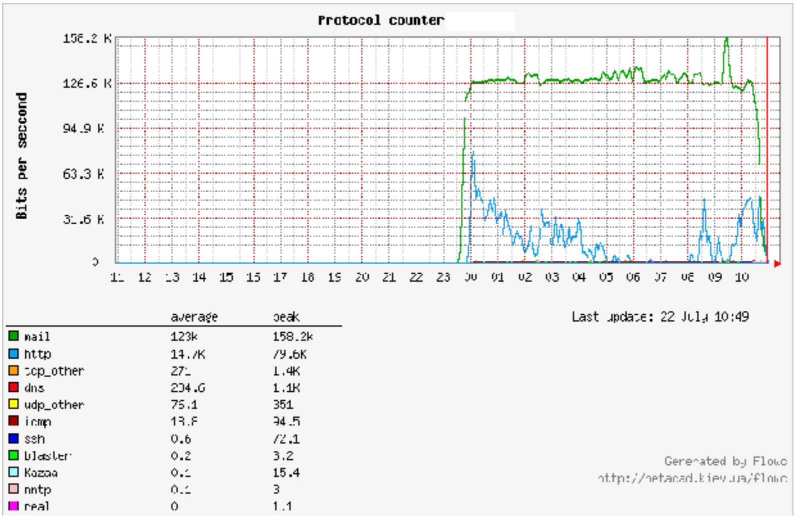
So if a virus can be so detrimental to your network, how do you spot them? By keeping an eye on your bandwidth graph with programs like MRTG or Cacti, you can detect a sudden unexpected increase in traffic. You can figure out what type of traffic they are generating by using a protocol analyser.



```
🟩 Incoming traffic                    🟦 Outgoing traffic
🟥 100% Bandwidth (128 kbps)
Max In: 127.90 kbps (100%)   Avg In: 125.46 kbps (98%)   Cur In: 126.94 kbps (99%)
Max Out:124.48 kbps (97%)   Avg Out: 46.84 kbps (37%)   Cur Out: 32.82 kbps (26%)
Working day averages In: 121.29 kbps Out: 84.90 kbps
                                                          Sat Jul 22 10:52:00 2006
```

Figure 5.2: Something has been using all 128k of the inbound bandwidth for a period of several hours. But what is causing the problem?

Figure 5.2 shows an MRTG graph taken from an ISP with a 128 Kbps link that has been saturated by a certain virus. The virus was sending out massive amounts of traffic (the light grey area incoming), seen as inbound traffic to the ISP. This is a sign of trouble, and a protocol analyser confirms it.

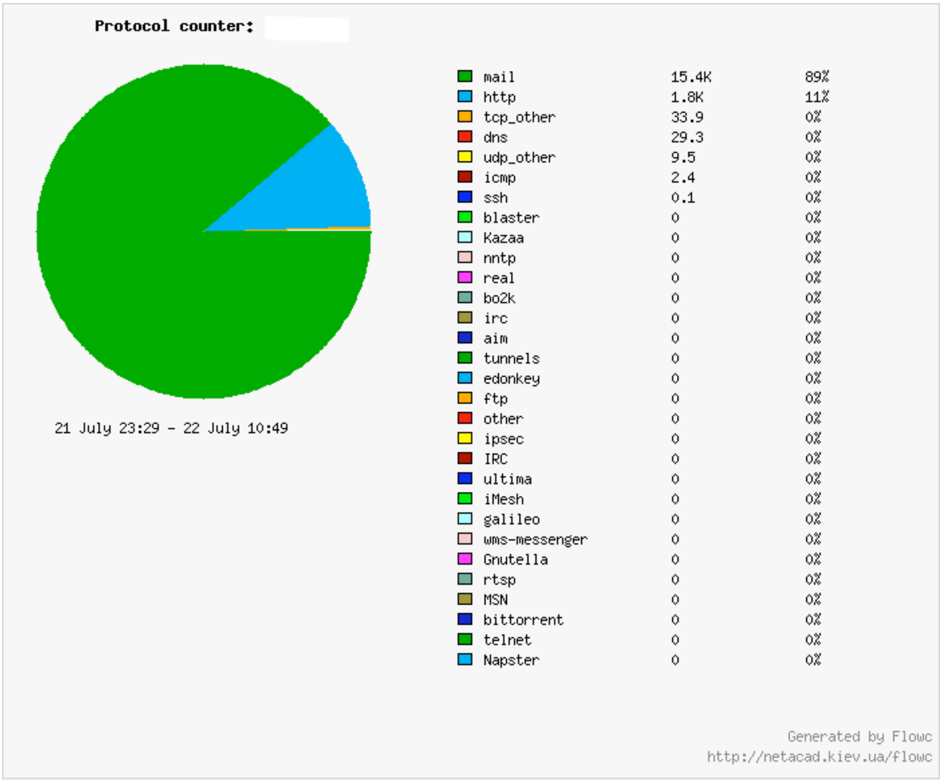


Last update: 22 July 10:49

	average	peak
🟩 mail	123k	158.2k
🟦 http	14.7K	79.6K
🟧 tcp_other	271	1.4K
🟥 dns	204.6	1.1K
🟨 udp_other	75.1	351
🟥 icmp	13.8	34.5
🟦 ssh	0.6	72.1
🟩 blaster	0.2	3.2
🟦 Kazaa	0.1	15.4
🟫 nntp	0.1	3
🟪 real	0	1.1

Figure 5.3: The highest line represents email utilisation. That definitely does not fit with the typical baseline usage for this site.

It is a mail virus known as Bagle. It is now sending out mass emails and may result in the IP getting blacklisted. The pie chart in figure 5.4 also reveals the damage done to browsing speeds.



🟩	mail	15.4K	89%
🟦	http	1.8K	11%
🟧	tcp_other	33.9	0%
🟥	dns	29.3	0%
🟨	udp_other	9.5	0%
🟥	icmp	2.4	0%
🟦	ssh	0.1	0%
🟩	blaster	0	0%
⬜	Kazaa	0	0%
⬜	nntp	0	0%
🟪	real	0	0%
▨	bo2k	0	0%
🟫	irc	0	0%
🟦	aim	0	0%
🟩	tunnels	0	0%
🟦	edonkey	0	0%
🟧	ftp	0	0%
🟥	other	0	0%
🟨	ipsec	0	0%
🟥	IRC	0	0%
🟦	ultima	0	0%
🟩	iMesh	0	0%
⬜	galileo	0	0%
⬜	wms-messenger	0	0%
🟪	Gnutella	0	0%
▨	rtsp	0	0%
🟫	MSN	0	0%
🟦	bittorrent	0	0%
🟩	telnet	0	0%
🟦	Napster	0	0%

Figure 5.4: Email represents 89% of the total observed traffic on the link. Since typical usage for this site is < 5%, this is a clear indication of a virus.

6
Performance Tuning

Car manufacturers often develop for the mass market, and strive to develop a "best fit" product that will be acceptable to the majority of customers. The product (in this case a car) will operate flawlessly for 90% of the people and give years of service. Sometimes, customers wish to tune their cars to work better under specific operating conditions. This may include the addition of snow tires, a supercharger or an engine additive. This optimisation will ensure that the car delivers the best performance for the customer relative to their environment. Networking hardware and software are a bit like this. Most times the default settings will be optimal, but there will be times when some optimisation is required to get the best performance for your specific environment.

The key to optimisation is understanding why you are optimising in the first place, and knowing the ramifications of the changes you are making. It was C.A.R. Hoare that said, "Premature optimization is the root of all evil in programming." If you upgrade your car by installing a massive supercharger because you are having trouble getting around corners, you are probably not going to achieve your goal. The car will go faster, but you will probably have an accident on your first corner when your woefully inadequate suspension decides to quit. If you understand the problem first, that the bad cornering was caused by poor suspension and not by engine output, then you can avoid an unnecessary (and expensive) upgrade as well as a potentially bad accident. There are a couple of things to remember when embarking on any optimisation:

1. **Practice change control.** Make backups of any configuration files you alter, so you can revert to an older version should it prove necessary.
2. **Take a baseline of the performance before and after changing any settings.** How else will you know if your settings have had a positive effect? This also may not be an exact numeric measurement, but could be the results of an operation. For example, if a user gets an error message when

they attempt to do something, making a change should make the error go away, or at least change to something more informative.

3. **Make sure you understand the problem you are trying to address.** If it is unclear precisely where the problem lies, it is even more important to make backups before you make any significant changes.
4. **Document the change.** This helps others understand why a change was made. Keeping a system change log can help you build a bigger picture of the state of the system, and may indicate long-term problems before they occur.
5. **Don't make a change unless you have a reason to.** If everything is working well enough that utilisation is acceptable and the users are happy, then why make changes?

There are system administrators who will swear blind that a particular optimisation change has fixed a problem, but they often cannot give actual proof when asked. This is usually because they do not understand what they have done or cannot measure the effect of the change. Defaults are there for a reason, so unless you have a valid reason to change the defaults, don't do it!

Of course, even when you fully understand the nature of the problem and the limitations of the system you are working with, optimisation can only take you so far. When you have exhausted optimisation techniques, it may be time to upgrade your hardware or software.

Squid cache optimisation

The default Squid settings are sufficient for most smaller networks. While these settings may work well for many installations, maximum performance can be achieved in large installations by making some changes to the defaults. Since software authors cannot know ahead of time how aggressively Squid may use system resources, the default settings are intentionally conservative. By fully optimising Squid to fit your server hardware, you can make the best possible use of the server resources and squeeze the maximum performance from your network.

Of course, no amount of configuration tweaking can help Squid run on hardware that simply can't handle the network load. Some of the parameters that may need to be changed to match your particular network include the server hardware itself, the disk cache and memory cache sizes, and even ACL lists. You will also need to know when (and how) to use multiple caches effectively when a single caching server simply isn't enough. This section will show you how to make the best possible caching solution for your network.

Squid optimisation is, of course, a complex subject in itself. For a more detailed explanation of this topic we suggest you refer to Duane Wessels' excellent book, *Squid: The Definitive Guide*.

Cache server hardware

The type of hardware you need to dedicate to your Squid cache depends on the amount of traffic that flows through it. Even moderately busy networks require little more than a typical desktop PC for use as the proxy server. Monitoring your CPU utilisation over time will help you to determine if you need to upgrade the processing power of your cache server. Squid runs as a single process, so using a multiprocessor machine will not give you much benefit. In terms of disks, the cache should ideally reside on a separate disk with its own data channel (e.g. the secondary IDE master with no slave on the same channel, or on its own SCSI or SATA bus) to give you the greatest performance benefit. It is very common to run the operating system on one disk and install the cache on another disk. This way, regular system activities (such as writing to the log files) do not interfere with access to the cache volumes. You will get better performance with a fast disk of adequate size, rather than a large disk with slower access times.

If you use RAID in your server, installing your cache on a RAID0 or RAID1 is acceptable. Squid already spreads the data load across multiple disks, so there is really no advantage to using RAID0. You definitely want to avoid using RAID5 for your cache. From the Squid FAQ:

> *Squid is the worst case application for RAID5, whether hardware or software, and will absolutely kill the performance of a RAID5. Once the cache has been filled Squid uses a lot of small random writes which [is] the worst case workload for RAID5, effectively reducing write speed to only little more than that of one single drive.*
>
> *Generally seek time is what you want to optimise for Squid, or more precisely the total amount of seeks/s your system can sustain. Choosing the right RAID solution generally decreases the amount of seeks/s your system can sustain significantly.*

You can still place the operating system (or other volumes) on a RAID 5 disk set for fault tolerance. Simply install the cache volumes on their own disks outside the RAID. The cache volumes are very quick to re-create in the case of a disk failure.

Squid also loves to use system RAM, so the more memory that is installed, the faster it will run. As we will see, the amount of RAM required increases as your disk cache size increases. We will calculate precisely how much is needed on page **181**.

There is also such a thing as having too many resources. An enormous disk and RAM cache is useless unless your clients are actually making cacheable requests. If your cache is very large and full of unusable data, it can increase cache access times. You should adjust the size of your cache to fit the amount of data your clients request.

Tuning the disk cache

A question often asked is, "How big should my cache volume be?" You might think that simply adding disk space will increase your hit rate, and therefore your performance will improve. This is not necessarily true. There is a point where the cache hit rate does not climb significantly even though you increase the available disk space. Cached objects have a ***TTL*** (***Time To Live***) that specifies how long the object can be kept. Every requested page is checked against the TTL, and if the cached object is too old, a new copy is requested from the Internet and the TTL is reset.

This means that huge cache volumes will likely be full of a lot of old data that cannot be reused. Your disk cache size should generally be between 10 and 18 GB. There are quite a few cache administrators that are running large cache disks of 140 GB or more, but only use a cache size of around 18GB. If you specify a very large cache volume, performance may actually suffer. Since Squid has to manage many concurrent requests to the cache, performance can decrease as the requests per second rise. With a very large cache, the bus connected to the disk may become a bottleneck. If you want to use a very large disk cache, you are better off creating smaller cache volumes on separate disks, and distributing the disks across multiple channels.

The **cache_dir** configuration directive specifies the size and type of disk cache, and controls how it is arranged in directories on the disk.

For a very fast link of 8 Mbps or more, or for a large transparent proxy at an ISP, you might want to increase the cache size to 16 GB. Adding this line to your **squid.conf** will create a 16 GB disk cache.

```
cache_dir aufs /var/spool/squid 16384 32 512
```

The **aufs** argument specifies that we want to use the new Squid on-disk storage format, formally referred to as Squid's async IO storage format. The cache will be stored in **/var/spool/squid**. The third argument (**16384**) specifies the maximum size of the cache, in megabytes. The last two arguments (**32** and **512**) specify the number of first and second level directories to be created under the parent directory of the cache. Together, they specify a total of x * y directories used by Squid for content storage. In the above example, the total number of directories created will be 16384 (32 top level directories, each containing 512 subdirectories, each containing many cached objects). Different filesystems have different characteristics when dealing with directories that contain large numbers of files. By spreading the cache files across more directories, you can sometimes achieve a performance boost, since the number of files in a given directory will be lower. The default value of 16 and 256 is nearly always sufficient, but they can be adjusted if desired.

If you are using Linux as your operating system, the best file systems to use are ext2fs/ext3fs or reiserfs. Cache filesystems using ext2fs/ext3fs or reiserfs should be mounted with the **`noatime`** option (specified in **`/etc/fstab`**). If you are using reiserfs, you should add the **`notail`** mount option as well. The **`noatime`** option tells the operating system not to preserve the access times of the file, and so saves overhead. The **`notail`** tells the file system to disable packing of files into the file system tree, which is used for saving space. Both these settings have a significant impact on disk performance.

For example, to mount the reiserfs filesystem on **`/dev/sdb1`** on **`/cache1`**, add this line to your **`/etc/fstab`**:

```
/dev/sdb1       /cache1 reiserfs notail,noatime 1 2
```

Remember that if you replace a cache volume, you will need to start Squid with the **`-z`** option once in order for the cache directories to be created. Also remember to make sure the cache directory is owned by the Squid user, otherwise Squid will not be able to write to the disk.

The size of your disk cache has a direct impact on the amount of memory needed on your server. In the next section, we will see how to calculate the required amount of system RAM for a given disk cache size.

Memory utilisation

Memory is very important to Squid. Having too little RAM in your machine will certainly reduce the effective size of your cache, and at worst can bring a system to its knees. If your Squid server is so low on RAM that it starts using swap space, the Squid process will quickly consume all available resources. The system will "thrash" as it attempts to allocate memory for Squid that causes the rest of the system, and eventually the Squid process itself, to be swapped out to disk.

While more RAM is nearly always better, a good rule of thumb is to allocate 10 MB of RAM per GB of cache specified by your **`cache_dir`** directive, plus the amount specified by the **`cache_mem`** directive (page **182**), plus another 20 MB for additional overhead. Since this memory is dedicated to Squid, you should also add enough additional memory to run your operating system.

For example, a 16 GB disk cache would require 160MB of memory. Assuming that **cache_mem** is set to 32 MB, this would require:

```
        160 MB + 32 MB + 20 MB = 212 MB
```

Squid itself will use approximately 212 MB of RAM. You should add enough additional RAM to accommodate your operating system, so depending on your needs, 512 MB or more of RAM would be a reasonable total amount for the cache server.

The default Squid settings allocate 100 MB for the disk cache and 8 MB for the memory cache. This means that you will need roughly 30 MB of RAM, plus enough for your operating system, in order to run Squid "out of the box." If you have less than this, Squid will likely start to swap at busy times, bringing network access to a crawl. In this case, you should either adjust your settings to use less than the default, or add more RAM.

Tuning the hot memory cache

The **cache_mem** directive specifies the maximum amount of RAM to be used for ***in-transit***, ***hot***, and ***negative-cached*** objects. In-transit objects represent incoming data, and take the highest priority. Hot objects are those that receive many on-disk cache hits. These are moved to RAM to speed up future responses when they are requested. Negative cached objects are objects that returned an error on retrieval, such as a refused connection or a **404 Not Found**. Note that the **cache_mem** directive does NOT specify the maximum size of the Squid server process, but only applies as a guideline to these three parameters, and may occasionally be exceeded. If you have a lot of memory available on your box you should increase this value, since it is much faster for Squid to get a cached file from memory rather than from the disk.

```
cache_mem 16 MB
```

The amount of memory is specified in 4 kilobyte blocks, and should not be too large. The default is 8 MB. Squid performance degrades significantly if the process begins to use swap space, so be sure there is always sufficient free memory when the cache server is being used at peak times. Use the equation above to calculate precisely how much RAM is required for your settings.

Cacheable content limits

The configuration directive **maximum_object_size** specifies the largest object that will be cached. Setting it low will probably increase the responsiveness of your cache, while setting it high will increase your byte hit ratio. Experience has shown that most requests are under 10 KB (you can confirm this for yourself with calamaris, page **81**). If you set this value too high (say, 1 GB or more to cache movies and other large content) then your hit rate will probably decline as your disk gets filled up with large files. The default is 4 MB.

```
maximum_object_size 64 MB
```

You can also configure the maximum size of objects in memory using the **`maximum_object_size_in_memory`** directive. Setting this to a low value is generally a good idea, as it prevents large objects from holding precious system memory. Typically around 90% of your requests will fit under 20K.

```
maximum_object_size_in_memory 20 KB
```

You can determine the optimum value for this setting by looking at your Squid logs once it is up and running for some time. An analysis of the Squid log files at my university shows that 88% of the requests are under 10K. The default maximum memory object size is 8 KB.

Access Control List (ACL) optimisation

Squid has two types of ACL components: ***ACL elements*** and ***ACL rules***. Elements allow you to match particular attributes of a given request (such as source IP, MAC address, user name, or browser type). You then use rules to determine whether access is granted or denied to requests that match a particular element.

ACL **elements** are processed with ***OR logic***. This means that Squid will stop processing the ACL as soon as a match occurs. To optimise the processing of your ACLs, you should arrange the list so that the most likely matches appear first. For example, if you define an element called **`badsites`** that contains a list of sites that are to be blocked, you should place the most likely site to be visited first. If you need to block *www.ccn.com* and *www.mynewsite.com* the ACL should be:

```
acl badsites dstdomain www.cnn.com www.mynewsite.com
```

If you find yourself making a large list of users (or any other information, such as IP or MAC addresses), you should place the items that have more chance of being matched at the front of the list.

When matching ACL **rules**, Squid uses ***AND logic***. Therefore, you should list the least-likely-to-match rules first. For example, if you have a ACL called **`localnet`** that matches any host on your local network, and an ACL called **`badsites`** that lists forbidden sites, you would define a rule this way:

```
http_access deny badsites localnet
```

This way, as soon as Squid determines that **badsites** does not match, it will immediately skip to the next **http_access** rule.

Some ACLs also require more processing time than others. For example, using a ***regex*** (***regular expression***) takes longer than matching against a literal list, since the pattern must be examined more closely. ACLs such as **src_domain** require a host name lookup, and must wait for a DNS response from the network. In such cases, using a caching DNS server (page **143**) can help to improve your Squid response times. If you can, avoid regex matches and other "expensive" ACLs whenever possible. Large access control lists are not generally a problem as Squid has an efficient way of storing them in memory. Since Squid can perform a literal search much faster than performing pattern matching, large lists can actually yield better performance than tiny pattern matches.

Examples of some commonly used elements and rules are available in **Appendix B** on page **269**.

Redirectors

Redirectors are typically used to direct Squid to serve content other than what was requested. Rather than simply denying access to a particular site or object with an abrupt error message, a redirector can send the user to a more informative web page or take some other action. Redirectors are often used to block sites based on content (e.g. games, pornography, or advertisements) or they can be used to redirect users to a page informing them of network status or that their access is disabled.

Redirectors are run as an external process, which can significantly increase processing time and resource consumption on your cache server. You should therefore make good use of ACLs to delay redirector processing until it is actually needed. For example, if you are blocking advertisements, you typically need to check every request for a match. If the redirector is slow, this could affect the overall performance of your cache. If you move advertisement blocking to a later stage (say, after all local and approved web sites are permitted without filtering), then you can reduce the server load significantly.

There are only a finite number of child processes available to answer all redirector requests. If your redirector takes too long to complete, you could run out of redirector processes. This happens when requests come in faster than they can be processed. This will be reflected in your **cache.log** as "FATAL: Too many queued redirector requests," and will cause your Squid process to prematurely exit. Make sure you allocate enough children to process your requests by using the **redirect_children** directive.

```
redirect_children 10
```

The default value is 5 children. If you specify too many, you will consume more system RAM and CPU. This will be reflected in your system monitoring logs (page **80**) or using a spot check tool such as **`top`** (page **74**).

You can also tell Squid to bypass the redirector should it run out of available children. This should obviously not be used in access control rules, but could be appropriate when using redirectors to block advertisements on a very busy network. This will allow you to use the redirector most of the time, but bypass it when the network load grows too large. You can enable this with the **`redirector_bypass`** directive.

```
redirector_bypass on
```

A good example of a real world redirector is ***adzapper***. Adzapper is a redirector that will intercept advertising and replace it with smaller static graphic files. This obviously saves bandwidth, and can help creating a more pleasing, advertisement-free environment. Adzapper matches each request using a pattern matching approach. If the request is matched, it will replace the request with a smaller static graphic which is often much smaller than the original, and is much faster to serve since it comes from the local machine.

A colleague at another university runs adzapper on his Squid servers in an interesting way. He has journalism students who want to see the adverts. In order to cater to their needs, he runs Squid on two different ports using the **`http_port`** directive. He then has a redirector ACL that will only match against requests on one of the ports using a **`myport`** ACL element. This way most users see no advertising, but if you connect to Squid on the alternate port you will get the full, unmodified page.

```
http_port 8080 8082
acl advertport myport 8082
redirector access deny advertport
redirector_access allow ALL
```

Adzapper is available from *http://adzapper.sourceforge.net/*.

Squidguard is another popular redirector program that is often used in conjunction with "blacklists" to block categories of sites, for example pornography. It is a very powerful and flexible script that offers a fine-grained level of access control. You can download it at *http://www.squidguard.org/*.

DansGuardian

An alternative to using a redirector to filter content is to use the very popular ***DansGuardian*** program. This program runs on your Squid (or other proxy) server, and intercepts the requests and responses between the browser and proxy. This allows you to apply very sophisticated filtering techniques to web requests and responses. DansGuardian is very often used for schools to block undesirable content, and has the side affect of saving bandwidth.

DansGuardian can be downloaded from *http://dansguardian.org/*.

Authentication helpers

Authentication introduces accountability to any service, and is a particularly important part of bandwidth control. By authenticating your Squid users, a unique identity is assigned to all requests passing through the proxy server. Should any particular user consume excessive bandwidth or otherwise violate the network access policy, access can be immediately revoked and the nature of the problem will be reflected in the system logs. Examples of how to implement authentication in Squid are provided in the **Implementation** chapter, on page **101**. More details on how to setup Squid authentication are covered in the Squid FAQ at *http://www.squid-cache.org/Doc/FAQ/FAQ-23.html* .

In Squid, authentication is not done for every page, but rather it expires after a specified amount of time. After the TTL expires, the client will again be asked for authentication. Two critical parameters that affect the performance of **basic authentication** are:

```
auth_param basic children 5
```

The **`children`** parameter tells Squid how many helper processes to use. The default value is 5, which is a good starting point if you don't know how many Squid needs to handle the load. If you specify too few, Squid warns you with messages in **`cache.log`**. Specifying more will allow you to accommodate a higher system load, but will require more resources. Another important parameter for basic authentication is **`credentialsttl`**:

```
auth_param basic credentialsttl 2 hours
```

A larger **`credentialsttl`** value will reduce the load on the external authenticator processes. A smaller value will decrease the amount of time until Squid detects changes to the authentication database. Think of this value as the maximum amount of time a user can continue to use the network before another authentication check is required. Note that this only affects positive results (i.e., successful validations). Negative results aren't cached by Squid. The default TTL value is two hours.

If you are using **digest authentication**, there are three variables to consider:

```
auth_param digest children 5
auth_param digest nonce_garbage_interval 5 minutes
auth_param digest nonce_max_duration 30 minutes
```

The **`children`** parameter for digest authentication is used the same way as it is for basic authentication.

The **`nonce_garbage_interval`** parameter tells Squid how often to clean up the nonce cache. The default value is every 5 minutes. A very busy cache with many Digest authentication clients may benefit from more frequent nonce garbage collection.

The **`nonce_max_duration`** parameter specifies how long each nonce value remains valid. When a client attempts to use a nonce value older than the specified time, Squid generates a **401 (Unauthorized)** response and sends along a fresh nonce value so the client can re-authenticate. The default value is 30 minutes. Note that any captured Authorisation headers can be used in a replay attack until the nonce value expires, and using a smaller value will limit the viability of this kind of attack. Setting the **`nonce_max_duration`** too low, however, causes Squid to generate 401 responses more often. Each 401 response essentially wastes the user's time as the client and server renegotiate their authentication credentials.

Note that if you are utilising a central authentication resource, you should ensure that this server can handle all the authentication requests that may be made of it. If the authentication source is unavailable, then users will not be able to browse the web at all.

Hierarchical caches

If you run a number of Squid cache servers you may want them to cooperate with each other. This can help to maximise the amount of cache hits, and eliminate redundant downloads.

Hierarchies can help when the volume of proxy traffic becomes too high for a single server to handle. For example, imagine a university that has a 5 Mb congested Internet connection. They originally ran two independent cache servers to provide fault tolerance and to split the load. The users were directed to the cache servers by using a DNS round-robin approach. If a user requested an object from cache A, and it did not have a cached copy of the object, it would then be fetched from the original site. The fact that this object may have existed on server B was never considered. The problem was that Internet bandwidth was wasted by fetching the object from the origin when it already existed on the local network.

A better solution would be to create a hierarchy between the two caches. That way, when a request is received by one of the caches, it would first ask the other cache if it has a local copy before making a request directly to the original server. This scenario works because the two servers are on a LAN, and access between the two servers is much faster than making requests from the Internet. A hierarchical cache configuration is shown in Figure 6.1.

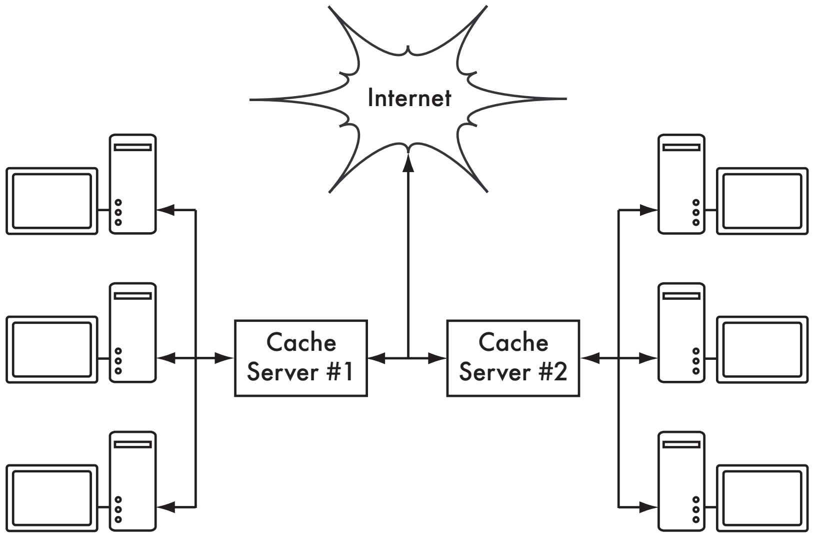


Figure 6.1: Multiple caches can distribute the load on a busy network across multiple servers, improving efficiency and response times.

One pitfall to watch out for when implementing cache hierarchies is the accidental creation of ***forwarding loops***. Forwarding loops can happen when two caches consider each other as the parent cache. When cache A requests an object from cache B, the request is forwarded back to cache A, then to cache B, and so on, until the cache servers run out of resources. For this reason, it is critically important that parent caches are never defined in a reciprocal manner.

Squid caches communicate with each other using the ***Internet Cache Protocol*** (***ICP***), as defined in RFC2186 and RFC2187. To optimise inter-cache performance and cut down on ICP queries, you should use ***cache digests***. Cache digests provide a very compact summary of the available objects in a particular cache, eliminating the need for caches to request individual objects just to determine whether or not they are available from the peer. This can dramatically speed up communication between your caches. There are no runtime configuration options to tune cache digests; simply compile Squid after passing the **`–enable-cache-digests`** to the **`configure`** script, and your cache hierarchies will use digests. There is a great deal of technical detail about cache digests in the Squid FAQ at: *http://www.squid-cache.org/Doc/FAQ/FAQ-16.html*

Since a hierarchy can improve your hit rate by 5% or more, it can be beneficial to overutilised and congested Internet links. The golden rule to remember with hierarchical caches is that your neighbour must be able to provide the data faster than the origin server for them to be worthwhile.

Configuring delay pools

Squid can throttle bandwidth usage to a certain amount through the use of ***delay pools***. Delay pools utilise the ***Token Bucket Filter*** (***TBF***) algorithm to limit bandwidth to a particular rate while still allowing short full-speed bursts.

Conceptually, delay pools are "pools" of bandwidth that drain out as people browse the web, and fill up at the rate specified. This can be thought of as a pot of coffee that is continually being filled. When there is bandwidth available in the "pot," requests are served at the best possible speed. Once the pot has emptied, it will only "refill" at a constrained rate as the coffee machine brews more coffee. To the user, this creates a small amount of fast bursting (for quickly loading a page or two). As requests continue, this is followed by a period of slower access, ensuring fairness and discouraging excessive use. As the user's requests are reduced, the coffee pot again has a chance to fill at the specified rate. This can be useful when bandwidth charges are in place, to reduce overall bandwidth usage for web traffic while continuing to provide quick responses to reasonable requests.

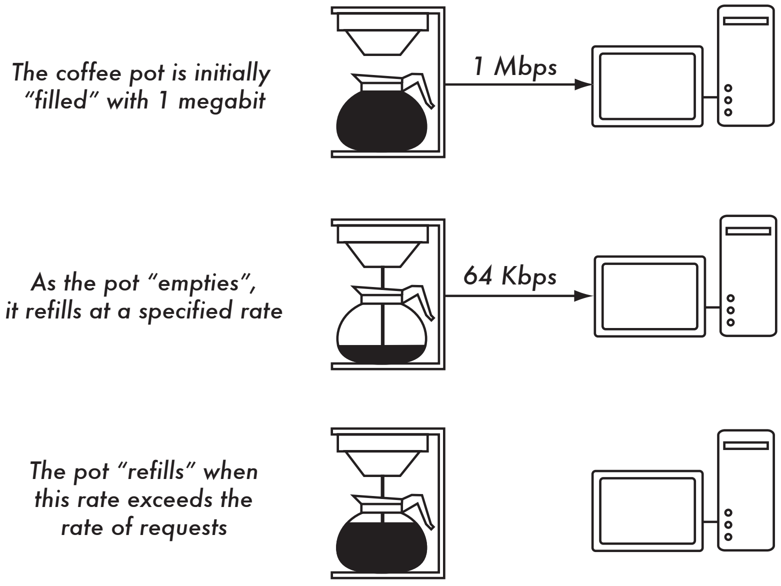

The coffee pot is initially "filled" with 1 megabit
1 Mbps
As the pot "empties", it refills at a specified rate
64 Kbps
The pot "refills" when this rate exceeds the rate of requests

Figure 6.2: Bandwidth is available as long as there is "coffee" in the "pot." When the delay pool is empty, requests are filled at the specified rate.

Delay pools can do wonders when combined with ACLs. This allows us to limit the bandwidth of certain requests based on any criteria. Delay behaviour is selected by ACLs (page **269**). For example, traffic can be prioritised based on destination, staff vs. student requests, authenticated vs. unauthenticated requests, and so on. Delay pools can be implemented at ISPs to improve the quality of service on a particular network. To enable delay pool support, Squid needs to be configured with the **`--enable-delay-pools`** option.

There are five classes of delay pools as of Squid 3.0:

1. Use one aggregate bucket for all traffic. This allows you to absolutely limit all traffic to a particular rate across the entire network.
2. One aggregate bucket for all traffic, and 256 additional buckets for all hosts in the class C network. In addition to setting an absolute rate, you can set individual rates for each IP address within the same class C network as the cache server.
3. One aggregate bucket for all traffic, 256 network buckets, and 256 individual buckets for each network (for a total of 65,536 individual buckets). With the class 3 delay pool, you have all of the limiting functionality of classes 1 and 2, and you can further limit traffic based on the source network.
4. Everything in the class 3 delay pool, with an additional limit for each user. This allows you to further limit the data rate based on the unique authenticated username.
5. Requests are grouped according to their tag (when using external authentication with the **`external_acl_type`** feature).

For example, this limits everyone to a global rate of 64 kbps, using a class 1 delay pool:

```
# Replace the network below with your own
acl delay_pool_1_acl src 192.168.1.0/255.255.255.0
# Define 1 delay pool, class 1
delay_pools 1
delay_class 1 1
# Manage traffic from our network with the delay pool
delay_access 1 allow delay_pool_1_acl
delay_access 1 deny all
# Lock users to 64kbps
delay_parameters 1 8000/8000
```

The last line specifies the rate at which the bucket is filled, and the maximum size of the bucket. Sizes are specified in bytes, **not** bits. In this example, the fill rate is 8000 bytes (64 000 bits) per second, and the maximum size of the bucket is also 8000 bytes. This defines a hard rate of 64 kbps with no bursting.

This example uses a class 2 delay pool to limit the overall rate to 128 kbps, restricting each IP address a maximum of 64 kbps:

```
# Replace the network below with your own
acl delay_pool_1_acl src 192.168.1.0/255.255.255.0
# Define 1 delay pool, class 2
```

```
delay_pools 1
delay_class 1 2
# Manage traffic from our network with the delay pool
delay_access 1 allow delay_pool_1_acl
delay_access 1 deny all
# Lock everyone to 128kbps and each IP to a maximum of 64kbps
delay_parameters 1 16000/16000 8000/8000
```

To use a class 3 delay pool, let's assume that each department or lab uses its own class-C IP block. This will limit the entire network to a maximum of 512 kbps, each class-C network will be limited to 128 kbps, and each individual IP address will be limited to a maximum of 64 kbps.

```
# Replace the network below with your own
acl delay_pool_1_acl src 192.168.0.0/255.255.0.0
# Define 1 delay pool, class 3
delay_pools 1
delay_class 1 3
# Manage traffic from our network with the delay pool
delay_access 1 allow delay_pool_1_acl
delay_access 1 deny all
# Lock everyone to 512kbps, each class-c to 128kbps,
# and each IP to 64kbps
delay_parameters 1 64000/64000 16000/16000 8000/8000
```

Finally, this example uses a class 4 delay pool to limit individual authenticated users to 64 kbps, no matter how many lab machines they are logged into:

```
# Replace the network below with your own
acl delay_pool_1_acl src 192.168.0.0/255.255.0.0
# Define 1 delay pool, class 4
delay_pools 1
delay_class 1 4
# Manage traffic from our network with the delay pool
delay_access 1 allow delay_pool_1_acl
delay_access 1 deny all
# Lock everyone to 512kbps, each class-c to 128kbps,
# each IP to 64kbps, and each user to 64kbps
delay_parameters 4 64000/64000 16000/16000 8000/8000 8000/8000
```

When using multiple delay pools, remember that the user will be placed in the first delay pool that matches them. If you define a fast ACL for sites that all users must be able to access quickly, and a slow ACL for everything else, you should place the **`delay_access`** match for the fast pool before that of the slow pool. Also note the **`deny all`** at the and of each delay pool match statement. This causes Squid to stop searching for a match for that particular pool.

More information

Squid is like other complex software, in that you should understand what you are doing before implementing it on a live network. It can be very handy to have a test Squid server that you can use to test potential changes before rolling the changes out on your production network. Change control is also very important when tuning Squid, as it is very easy to make a simple change that completely breaks the cache.

Squid is a tremendously powerful piece of software, and it is not possible do justice to all the different ways that Squid can be configured in this text. For more in-depth knowledge of Squid, we highly recommend *Squid: the Definitive Guide* by Duane Wessels, published by O'Reilly Media. The Squid FAQ (*http://wiki.squid-cache.org/*) and Google in general have plenty of Squid examples that you can use to make Squid perfectly fit your environment.

Monitoring your Squid performance

Squid provides two interfaces for monitoring its operation: SNMP and the cache manager.

The ***cachemgr*** interface is a command line interface that is accessed via the **`squidclient`** program (which ships with Squid). There is also a web-based interface called ***cachemgr.cgi*** that displays the cachemgr interface via a web page.

The SNMP interface is nice because it is easy to integrate into your existing system management package, such as Cacti (page **84**) or MRTG (page **83**). Squid's SNMP implementation is disabled by default, and must be enabled at compile time using the **`--enable-snmp`** option. Once you have an SNMP-enabled Squid, you also need to enable SNMP in your **`squid.conf`** using an ACL:

```
acl snmppublic snmp_community public
snmp_access allow snmppublic localhost
snmp_access deny all
```

Change the "public" community string to something secret. The only drawback to using SNMP is that it cannot be used to monitor all of the metrics that are available to the cachemgr interface.

All SNMP community strings in our examples start with the prefix **`enterprises.nlanr.squid.cachePerf`**. This prefix has been omitted in these examples for clarity. For example, the full SNMP community string in the next example is:

```
enterprises.nlanr.squid.cachePerf.cacheSysPerf.cacheSysPageFaults
```

For a full list of all available SNMP community strings, see the comprehensive list at: *http://www.squid-cache.org/SNMP/snmpwalk.html*

Some of the metrics you should monitor include:

- **Page Fault Rate**. Page faults occur when Squid needs to access data that has been swapped to disk. This can cause severe performance penalties for Squid. A high page rate (> 10 per second) may cause Squid to slow considerably.

  To monitor the page fault rate with the cachemgr interface:

```
# squidclient mgr:info | grep 'Page faults'
   Page faults with physical i/o: 3897
```

  The SNMP community string for tracking page faults is:

```
.cacheSysPerf.cacheSysPageFaults
```

- **HTTP request rate**. This is simply a measure of the total number of HTTP requests made by clients. If you monitor this number over time, you will have a good idea of the average load on your Squid cache throughout the day.

```
# squidclient mgr:info | grep 'Number of HTTP requests'
Number of HTTP requests received:       126722
# squidclient mgr:info | grep 'Average HTTP requests'
Average HTTP requests per minute since start:   340.4
```

  The SNMP community string is:

```
.cacheProtoStats.cacheProtoAggregateStats.cacheProtoClientHttpRequests
```

- **HTTP and DNS Service time**. These two metrics indicate the amount of time HTTP and DNS requests take to execute. They are valuable to monitor as they can indicate network problems beyond the cache server. A reasonable HTTP service time should be between 100 and 500 ms. If your service requests rise above this, you should take a look at your network connection as it may indicate a congested network. High DNS service times may indicate a potential problem with your caching DNS server (page **143**) or upstream network problems. These times may also rise if the Squid server itself is overloaded.

```
# squidclient mgr:5min | grep client_http.all_median_svc_time
client_http.all_median_svc_time = 0.100203 seconds
# squidclient mgr:5min | grep dns.median_svc_time
dns.median_svc_time = 0.036745 seconds
```

  Squid can also warn you if the HTTP service time goes over a specified threshold. It will log the warning to disk, which can notify you immediately if you are using Nagios (page **88**) or another log watching package (page **80**).

To enable high response time warnings, add this to your **squid.conf**. Times are specified in milliseconds.

```
high_response_time_warning 700
```

For SNMP, you can request the average response times in the last 1, 5, or 60 minutes. The relevant community strings are:

```
.cacheProtoStats.cacheMedianSvcTable.cacheMedianSvcEntry.cacheHttpAllSvcTime.1
.cacheProtoStats.cacheMedianSvcTable.cacheMedianSvcEntry.cacheHttpAllSvcTime.5
.cacheProtoStats.cacheMedianSvcTable.cacheMedianSvcEntry.cacheHttpAllSvcTime.60

.cacheProtoStats.cacheMedianSvcTable.cacheMedianSvcEntry.cacheDnsSvcTime.1
.cacheProtoStats.cacheMedianSvcTable.cacheMedianSvcEntry.cacheDnsSvcTime.5
.cacheProtoStats.cacheMedianSvcTable.cacheMedianSvcEntry.cacheDnsSvcTime.60
```

- **Open File Descriptors**. A Squid server that runs out of file descriptors will perform like a sick puppy. If you start running out of file descriptors, you will need to increase the number of file handles available to Squid.

```
# squidclient mgr:info | grep -i 'file desc'
File descriptor usage for squid:
        Maximum number of file descriptors:   8192
        Largest file desc currently in use:    815
        Number of file desc currently in use:  268
        Available number of file descriptors: 7924
        Reserved number of file descriptors:   100
```

On Linux, you may need to increase the overall number of file descriptors available to the system, as well as the number allowed per process. To increase the total number of available file descriptors on a Linux system, write the new value to **/proc/sys/fs/file-max**. For example:

```
echo 32768 > /proc/sys/fs/file-max
```

You should add that command to your **/etc/rc.d/rc.local** (or the equivalent in **/etc/sysctl.conf**) to preserve the change across boots. To increase the number of descriptors available to an individual process, use **ulimit**:

```
# ulimit -n 16384
```

This sets the number of file descriptors for the current process. You can call this in the initialisation script that runs Squid, just prior to launching the daemon.

The SNMP community string for available file descriptors is:

```
 .cacheSysPerf.cacheCurrentUnusedFDescrCnt
```

- **Hit ratio**. The hit ratio gives you an idea of the effectiveness of your cache. A higher the ratio means that more requests are served from the cache rather than the network.

```
#squidclient mgr:info | grep 'Request Hit Ratios'
        Request Hit Ratios:     5min: 28.4%, 60min: 27.4%
```

  As with HTTP and DNS service response time, you can request the 1 minute, 5 minute, or 60 minute average using the proper SNMP community string:

```
.cacheProtoStats.cacheMedianSvcTable.cacheMedianSvcEntry.cacheRequestHitRatio.1
.cacheProtoStats.cacheMedianSvcTable.cacheMedianSvcEntry.cacheRequestHitRatio.5
.cacheProtoStats.cacheMedianSvcTable.cacheMedianSvcEntry.cacheRequestHitRatio.60
```

- **CPU Utilisation**. While you are likely already monitoring overall CPU utilisation, it can be useful to know what percentage is being used by Squid. Periods of constant 100% utilisation may indicate a problem that needs investigation. The CPU can also become a hardware bottleneck, and constant high utilisation may mean that you need to optimise your Squid, upgrade your server, or lighten the load by bringing up another cache box (page **187**).

```
# squidclient mgr:5min | grep cpu_usage
cpu_usage = 1.711396%
```

  The SNMP community string for CPU usage is:

```
enterprises.nlanr.squid.cachePerf.cacheSysPerf.cacheCpuUsage
```

Graphing Squid metrics

Since the values can be extracted via SNMP or the cachmgr interface, they can also be easily graphed. This can show you trends that will help you predict how your cache will perform in the future.

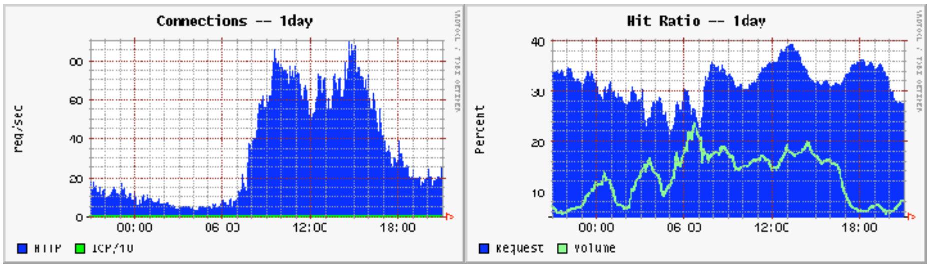


Figure 6.3: The graph on the left shows the number of connections per second to your Squid cache. The graph on the right shows the cache hit ratio as expressed by the number of requests (the dark area) and volume (the light line).

Examples of how to set up RRDtool to graph your Squid cache can be found at *http://www.squid-cache.org/~wessels/squid-rrd/*.

Traffic shaping

Traffic shaping is used to strictly enforce a particular data rate based on some predefined criteria. Let's say you are controlling your HTTP usage via a Squid proxy with delay pools, your mail has spam controls, and you enforce usage quotas for your users through proxy authentication and accounting. One day while running Ntop (page **76**), you notice that there is surprisingly large amount of traffic on port 22 (SSH). It appears that some users have figured out how to tunnel their P2P requests via the SSH port. You can't disable SSH access throughout your entire organisation, and while you could simply disable access for these users, you would need to repeat the process each time users figure out a new way to circumvent the firewall.

This is where traffic shaping can be very useful. By using traffic shaping techniques, you can limit all traffic using port 22 to a low rate, such as 16 Kbps. This will permit legitimate SSH shell traffic, while limiting the impact that tunneling and large file transfers have on the network.

Traffic shaping is best using in conjunction with traffic monitoring (page **83**) so that you have a clear idea of how much bandwidth is used by various services and machines. Before you can decide how to shape your traffic, it is paramount that you first understand how bandwidth is consumed on your network.

It is important to remember that traffic can only be shaped on transmit, not on receive. By the time inbound packets have been received, they have already traversed the network, and delaying their delivery further is usually pointless.

Linux traffic control and QoS tools

Linux has very good tools for managing bandwidth use, which are unfortunately not well known or widely used. This is probably because the tools to manipulate them are complex and poorly documented. You may well find BWM tools easier to use, but we will describe the kernel tools because they are more powerful.

To perform traffic shaping in Linux, you will need the ***iproute2*** package. It is available from *http://linux-net.osdl.org/index.php/Iproute2* and is a standard package in most distributions. This gives you the **`tc`** command, which is used for all traffic control and ***Quality of Service*** (***QoS***) configuration under Linux. Another useful component is the **`ipt_CLASSIFY`** kernel module, which comes with recent Linux 2.6 kernels. This integrates **`iptables`** with traffic control, allowing you to write netfilter rules that group traffic into certain classes.

Every network interface in Linux has a ***queuing discipline*** (***qdisc***) associated with it. The queuing discipline controls when and how the interface is allowed to send packets. You can define a queuing discipline on the external network interface of your router in order to control traffic that you send to the Internet. You can also define a queuing discipline on the internal interface to control traffic that your users receive.

None of these queuing disciplines will help you unless the queue is "owned" by the firewall itself. For example, if your firewall is connected to an ADSL modem by Ethernet, then the Ethernet link to the modem is much faster than the ADSL line itself. Therefore, your firewall can send traffic to the modem at a rate faster than the modem can send out. The modem will happily queue traffic for you, but this means that the queue is on the modem and not on the firewall, and therefore it cannot easily be shaped.

Similarly, your internal LAN is usually much faster than your Internet connection, and so packets destined for your network will build up on the Internet provider's router, rather than on the firewall. The way to prevent this is to ensure that the firewall sends less traffic to the ADSL line than the line's maximum upload speed, and less traffic to your LAN than the line's maximum download speed.

There are two types of queuing discipline: ***classful*** and ***classless***. Classful qdiscs can have other qdiscs attached to them. They modify the behaviour of all attached qdiscs. Classless qdiscs cannot have any other qdiscs attached, and are much simpler.

Various queuing disciplines are available. The simplest is ***pfifo_fast***, which separates traffic into three priority classes based on the ***Type Of Service*** (***TOS***) field in the packet header. High priority packets are always dequeued (sent) first, followed by medium and low. The pfifo_fast qdisc is classless, and the only thing that can be configured is the mapping between TOS field values and priority levels. It does not allow you to throttle your traffic in order to take ownership of the queue. See page **199** for a more detailed discussion of the TOS field.

Another useful queuing discipline is the ***Token Bucket Filter*** (***TBF***). This qdisc is also classless, but it provides a simple way to throttle traffic on a given interface. For example, the following command throttles outbound traffic on the first Ethernet interface (eth0) to 220 kbps, and limits the maximum latency to 50 ms:

```
# tc qdisc add dev eth0 root tbf rate 220kbit latency 50ms burst 1540
```

If you want more control over your bandwidth, you will need to use a classful qdisc such as ***Hierarchical Token Buckets*** (***HTB***). This allows you to place arbitrary traffic in classes and restrict the amount of bandwidth available to each class. You can also allow classes to borrow unused bandwidth from other classes. A very simple example is given below. It will throttle outgoing traffic to 50 kbps.

```
# tc qdisc add dev eth0 handle 1 root htb default 10
# tc class add dev eth0 classid 1:1 htb rate 100kbit ceil 100kbit
# tc class add dev eth0 classid 1:10 parent 1:1 htb rate 50kbit ceil 50kbit
```

To remove the current qdisc setup from an interface and return to the default, use the following command:

```
# tc qdisc del dev eth0 root
```

The following example uses the netfilter CLASSIFY module to place HTTP and SSH traffic into their own classes (1:20 and 1:30 respectively), and places individual limits on them. If you have tried another example, delete the root qdisc before running this one.

```
# tc qdisc add dev eth0 handle 1 root htb default 10
# tc class add dev eth0 classid 1:1 htb rate 100kbit ceil 100kbit
# tc class add dev eth0 classid 1:10 parent 1:1 htb rate 50kbit ceil 50kbit
# tc class add dev eth0 classid 1:20 parent 1:1 htb rate 30kbit ceil 30kbit
# tc class add dev eth0 classid 1:30 parent 1:1 htb rate 20kbit ceil 100kbit
# iptables -t mangle -F
# iptables -t mangle -A OUTPUT -p tcp --sport 80 -j CLASSIFY --set-class 1:20
# iptables -t mangle -A OUTPUT -p tcp --sport 22 -j CLASSIFY --set-class 1:30
```

The above example creates a root class (1:1) which is limited to 100 kbps, and 3 classes underneath (1:10, 1:20 and 1:30), which are guaranteed 50 kbps, 30 kbps and 20 kbps respectively. The first two are also limited to their guaranteed rates, whereas the third is allowed to borrow unused bandwidth from the other two classes up to a maximum of 100 kbps.

Two common pitfalls in traffic control are misusing branch nodes and rates. If a node has any children, then it is a ***branch node***. Nodes without children are ***leaf nodes***. You may not enqueue traffic for a branch node. That means that a branch node must not be specified as the default class, nor used in an iptables CLASSIFY rule.

Because the rate on each node is a guarantee, the rate of a branch node must be exactly equal to the total rates of all its children (leaf nodes). In the above example, we could not have specified a rate other than 100 kbps for class 1:1. Finally, it does not make sense to specify a ceil (maximum bandwidth) for any class that is greater than the ceil of its parent class.

Using SFQ to promote fairness over a 56k modem

If you have a device which has an identical link speed and actual available rate, such as a normal 56k analogue modem, you may want to use ***Stochastic Fairness Queuing*** (***SFQ***) to promote fairness. SFQ is a fair queueing algorithm designed to require fewer calculations than other algorithms while being almost perfectly fair. SFQ prevents a single TCP session or UDP stream from flooding your link at the expense of others. Rather than allocating a queue for each session, it uses an algorithm that divides traffic over a limited number of queues using a hashing algorithm. This assignment is nearly random, hence the name "stochastic."

This uses tc to enable SFQ on the device **`ppp0`**:

```
# tc qdisc add dev ppp0 root sfq perturb 10
```

That's all there is to it. Traffic that leaves ppp0 is now subject to the SFQ algorithm, and individual streams (such as downloads) should not overpower other streams (such as interactive SSH sessions).

Implementing basic Quality of Service (QoS)

If your physical link is saturated and you wish to implement basic quality of service to prioritise one type of traffic over another, you can use the ***PRIO*** queueing discipline. A packet's ***Type Of Service*** (***TOS***) bits determine whether a packet should be prioritised to minimise delay (***md***), maximise throughput (***mt***), maximise reliability (***mr***), minimise monetary cost (***mmc***), or some combination of these. Applications request the appropriate TOS bits when transmitting packets. For example, interactive applications like **`telnet`** and **`ssh`** may set the "minimise delay" bits, while file transfer applications like **`ftp`** may wish to "maximise throughput."

When a packet arrives at the router, it is queued into one of three ***bands***, depending on the TOS bits. The first band is tried first, and higher bands are only used if the lower classes have no packets queued to be sent out. According to the Linux Advanced Routing & Traffic Control HOWTO, the different combinations of TOS bits result in the following assignment of bands. All of the possible combinations of TOS bits are enumerated in the table on the next page.

Service requested	Linux priority	Band
Normal	Best Effort	1
Minimise Monetary Cost	Filler	2
Maximise Reliability	Best Effort	1
MMC + MR	Best Effort	1
Maximise Throughput	Bulk	2
MMC + MT	Bulk	2
MR + MT	Bulk	2
MMC + MR + MT	Bulk	2
Minimise Delay	Interactive	0
MMC + MD	Interactive	0
MR + MD	Interactive	0
MMC + MR + MD	Interactive	0
MT + MD	Interactive Bulk	1
MMC + MT + MD	Interactive Bulk	1
MR + MT + MD	Interactive Bulk	1
MMC + MR + MT + MD	Interactive Bulk	1

The PRIO qdisc doesn't actually shape traffic to match a particular rate. It simply assigns priority to different types of traffic as it leaves the router. Therefore it only makes sense to use it on a fully saturated link where granting priority to certain kinds of traffic makes sense. On an unsaturated link, PRIO will have no discernible performance impact.

To implement basic QoS with fairness, we can use a combination of PRIO and SFQ.

```
# tc qdisc add dev eth0 root handle 1: prio
# tc qdisc add dev eth0 parent 1:1 handle 10: sfq
# tc qdisc add dev eth0 parent 1:2 handle 20: sfq
# tc qdisc add dev eth0 parent 1:3 handle 30: sfq
```

This creates the following priority decision tree:

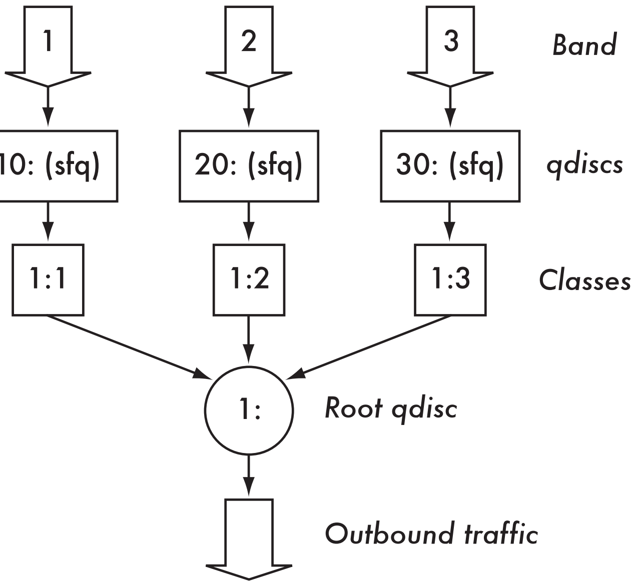


Figure 6.4: The priority decision tree used by PRIO. Note that while this example uses SFQ for each band, you could use a different qdisc (such as HTB) for every band.

Traffic for band 0 gets queued into qdisc 10:, band 1 traffic is sent to qdisc 20:, and band 2 traffic is sent to qdisc 30:. Each qdisc is released according to the SFQ algorithm, with lower numbered qdiscs taking priority.

Class Based Queueing (CBQ)

Another popular queueing discipline is ***Class Based Queueing*** (***CBQ***). CBQ is similar to the PRIO queue in that lower priority classes are polled after higher ones have been processed. While CBQ is likely the most widely known queueing algorithm, it is also one of the most complex and least accurate. But it can work well in many circumstances. Remember that it's a good idea to benchmark your network performance (page **89**) after making changes to your traffic shaping configuration to be sure that the shaper is working as you intend.

Let's assume that we have a server connected to a 2 Mbps link. We want to give web and mail a combined 1.5 Mbps of bandwidth, but not allow web traffic to exceed 1 Mbps and not allow mail to exceed 750 Kbps.

```
# Setup CBQ on the interface
tc qdisc add dev eth0 root handle 1:0 cbq bandwidth 100Mbit avpkt 1000 cell 8
# Lock us down to 1Mbit
tc class add dev eth0 parent 1:0 classid 1:1 cbq bandwidth 100Mbit \
  rate 1.5Mbit weight 150Kbit prio 8 allot 1514 cell 8 \
  maxburst 20 avpkt 1000 bounded
```

```
# Create a class for our web traffic
tc class add dev eth0 parent 1:1 classid 1:3 cbq bandwidth 100Mbit \
  rate 1Mbit weight 100Kbit prio 5 allot 1514 cell 8 maxburst 20 avpkt 1000
# Create a class for our mail traffic
tc class add dev eth0 parent 1:1 classid 1:4 cbq bandwidth 100Mbit \
  rate 750Kbit weight 75Kbit prio 5 allot 1514 cell 8 maxburst 20 avpkt 1000
# Add SFQ for fairness
tc qdisc add dev eth0 parent 1:3 handle 30: sfq
tc qdisc add dev eth0 parent 1:4 handle 40: sfq
# Classify the traffic
tc filter add dev eth0 parent 1:0 protocol ip prio 1 u32 match ip sport 80 \
  0xffff flowid 1:3
tc filter add dev eth0 parent 1:0 protocol ip prio 1 u32 match ip sport 25 \
  0xffff flowid 1:4
```

For more details about this complex qdisc, see the Linux Advanced Routing and Traffic Control HOWTO at *http://lartc.org/lartc.html* .

WonderShaper

WonderShaper (*http://lartc.org/wondershaper/*) is a relatively simple shell script that attempts to achieve the following:

- Maintain low latency for interactive traffic
- Allow web browsing at reasonable speeds while uploading or downloading
- Make sure uploads don't impact downloads, and vice-versa

The WonderShaper script can use the CBQ or HTB packet schedulers, and is configured by simply setting some variables at the top of the script. While it is intended for use with residential DSL networks, it provides a good example of CBQ and HTB queueing that can be used as the starting point for a more complex traffic shaping implementation.

BWM Tools

BWM Tools (*http://freshmeat.net/projects/bwmtools*) is a full firewall, shaping, monitoring, logging, and graphing package. It is implemented using userspace utilities, so any Linux kernel that supports the iptables **`-j QUEUE`** target will work. Shaping of traffic is easily accomplished by defining classes and creating flows.

The configuration file for BWM Tools is defined in XML format. A Class for SMTP traffic can be defined as follows:

```
<class name="smtp_traffic">
    <address name="inbound" dst="192.168.1.1" proto="tcp" dst-port="25">
    <address name="outbound" src="192.168.1.1" proto="tcp" src-port="25">
</class>
```

Change 192.168.1.1 to match your external IP address. To shape the traffic to allow an absolute 128 kbps inbound and 64 kbps outbound, use this:

```
<traffic>
    <flow name="smtp_inbound" max-rate="16384">
        inbound;
    </flow>
    <flow name="smtp_outbound" max-rate="8192">
        outbound;
    </flow>
</traffic>
```

Note that rates are specified in bytes per second, so you should multiply the rate by 8 to get the bits per second.

Traffic shaping with BSD

Packet Filter (***PF***) is the system for configuring packet filtering, network address translation, and packet shaping in FreeBSD and OpenBSD. PF uses the ***Alternate Queuing*** (***ALTQ***) packet scheduler to shape traffic. PF is available on FreeBSD in the basic install, but without queueing/shaping ability. To enable queueing, you must first activate ALTQ in the kernel. This is an example of how to do so on FreeBSD.

```
# cd /usr/src/sys/i386/conf
# cp GENERIC MYSHAPER
```

Now open **`MYSHAPER`** file with your favourite editor and add the following at the bottom of the file:

```
device pf
device pflog
device pfsync
options         ALTQ
options         ALTQ_CBQ        # Class Bases Queuing (CBQ)
options         ALTQ_RED        # Random Early Detection (RED)
options         ALTQ_RIO        # RED In/Out
options         ALTQ_HFSC       # Hierarchical Packet Scheduler (HFSC)
options         ALTQ_PRIQ       # Priority Queuing (PRIQ)
options         ALTQ_NOPCC      # Required for Multi Processors
```

Save the file and recompile the kernel by using the following commands:

```
# /usr/sbin/config MYSHAPER
# cd ../compile/MYSHAPER
# make cleandepend
# make depend
# make
# make install
```

Ensure PF is activated on boot. Edit your **/etc/rc.conf** and add this:

```
gateway_enable="YES"
pf_enable="YES"
pf_rules="/etc/pf.conf"
pf_flags=""
pflog_enable="YES"
pflog_logfile="/var/log/pflog"
pflog_flags=""
```

PF with ALTQ is now installed.

An example of how to rate limit SMTP traffic to 256 Kbps and HTTP traffic to 512 Kbps is shown below. First, create the file **/etc/pf.conf** and begin by identifying the interfaces. PF supports macros, so you don't have to keep repeating yourself.

```
#The interfaces
gateway_if  = "vr0"
lan_if      = "vr1"
```

You should replace vr0 and vr1 with your network interface card names. Next, identify the ports by using Macros:

```
mail_port = "{ 25, 465 }"
ftp_port = "{ 20, 21 }"
http_port = "80"
```

Identify the host or hosts:

```
mailsrv = "192.168.16.1"
proxysrv = "192.168.16.2"
all_hosts = "{" $mailsrv $proxysrv "}"
```

If you have several IP address blocks, a table will be more convenient. Checks on a table are also faster.

```
table <labA> persist { 192.168.16.0/24 }
table <labB> persist { 10.176.203.0/24 }
```

Now that your interfaces, ports, and hosts are defined, it's time to begin shaping traffic. The first step is to identify how much bandwidth we have at our disposal (in this case, 768 Kbps). We wish to limit mail to 256 Kbps and web traffic to 512 Kbps. ALTQ supports ***Class Based Queueing*** (***CBQ***) and ***Priority Queueing*** (***PRIQ***).

Class Based Queueing is best used when you want to define several bucket queues within primary queues. For example, suppose you have Labs A, B, and C on different subnets, and you want each individual lab to have different bandwidth allocations for mail and web traffic. Lab A should receive the lion's share of the bandwidth, then Lab B, and Lab C should receive the least.

Priority Queuing is more suitable when you want certain ports or a range of IPs to have priority, for example when you want to offer QoS. It works by assigning multiple queues to a network interface based on protocol, port, or IP address, with each queue being given a unique priority level. The queue with the highest priority number (from 7 to 1) is always processed ahead of a queue with a lower priority number. In this example. we will use CBQ.

Use your favourite editor to edit the file **`/etc/pf.conf`** and add the following at the bottom:

```
altq on $ gateway_if bandwidth 768Kb cbq queue { http, mail }
  queue http bandwidth 512Kb cbq (borrow)
  queue mail bandwidth 256Kb
```

The **`borrow`** keyword means that HTTP can "borrow" bandwidth from the mail queue if that queue is not fully utilised.

You can then shape the outbound traffic like this:

```
pass in quick on lo0 all
pass out quick on lo0 all

pass out on $gateway_if proto { tcp, udp } from $proxysrv to any \
  port http keep state queue http

pass out on $gateway_if proto { tcp, udp } from $mailsrv to any \
  port smtp keep state queue mail

block out on $gateway_if all

pass in on $gateway_if proto { tcp, udp } from any to $mailsrv \
  port smtp keep state
pass in on $gateway_if proto { tcp, udp } from any to $proxysrv \
  keep state
pass in on $lan_if proto { tcp, udp }from $mailsrv to any keep state
pass in on $lan_if proto { tcp, udp } from $proxysrv to any keep state
block in on $lan_if all
```

Farside colocation

Under many circumstances, you can save money and improve Internet speeds by moving public-facing services into a ***colocation facility***. This service can be provided by your ISP or a third party hosting service. By moving your public servers out of your organisation and into a facility closer to the backbone, you can reduce the load on your local connection while improving response times.

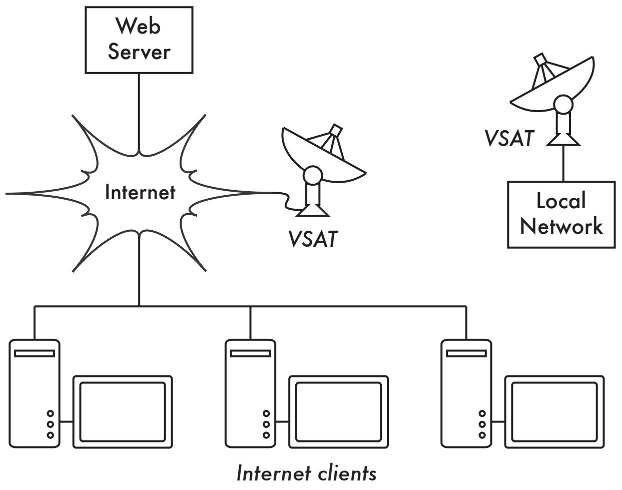


Figure 6.5: Colocation can remove load from your local Internet connection.

Colocation (often simply referred to as ***colo***) works very will with services such as:

- **Web servers**. If your public web servers require high bandwidth, colocation makes a lot of sense. Bandwidth in some countries is extremely cheap and affordable. If you have a lot of visitors to your site, choosing to host it "off-shore" or "off-site" will save you time, money, and bandwidth. Hosting services in Europe and the United States are a fraction of the cost of equivalent bandwidth in Africa.
- **Public / backup DNS servers**. DNS allows you to create redundant servers in an effort to increase service reliability. If your primary and secondary servers are hosted at the same physical location, and the network connectivity to that location goes down, your domains will effectively "fall off the Internet." Using colocation to host DNS gives you redundant name service with very little chance of both DNS servers being down at the same time. This is especially important if you're doing a lot of web hosting or if people are paying you for a hosting service, as a loss of DNS service means that all of your customer's services will be unreachable.

  Installing a public DNS server at a colo can also help improve performance, even on an unsaturated line. If your organisation uses a VSAT or other high latency connection, and your only DNS server is hosted locally, then Internet requests for your domain will take a very long time to complete.

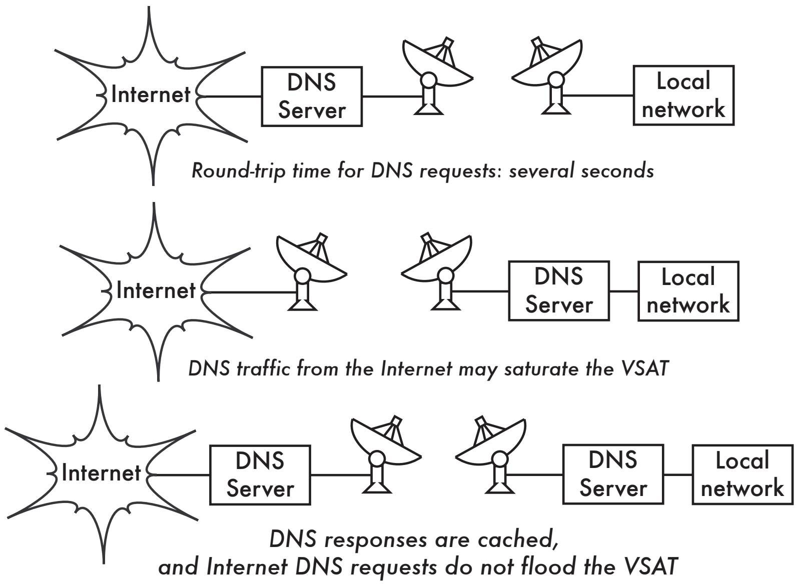

Internet
DNS Server
Local network
Round-trip time for DNS requests: several seconds
Internet
DNS Server
Local network
DNS traffic from the Internet may saturate the VSAT
Internet
DNS Server
DNS Server
Local network
DNS responses are cached, and Internet DNS requests do not flood the VSAT

Figure 6.6: Inefficient DNS configuration can cause unnecessary delays and wasted bandwidth.

If you use private IP addressing internally (i.e., you are using NAT) then your DNS server will need to send different responses depending on where the requests come from. This is called ***split horizon DNS***, and is covered on page **212**.

- **Email**. By combining anti-virus and anti-spam measures at a colo server, you can drastically reduce bandwidth wasted on undesired content. This technique is sometimes called ***far-side scrubbing***. If you rely on client-side virus scanners and spam filters, the filtering can only occur after the entire mail has been received. This means that you have needlessly downloaded the entire message before it can be classified as spam and discarded. Often times, 80% or more of all inbound mail in a normal educational setting can be spam and virus content, so saving this bandwidth is vital.

  To implement far-side scrubbing, you should set up an anti-virus and anti-spam solution as discussed in chapter four: **Implementation**, and host that server at a colocation facility. You should also install a simple internal email server, with firewall rules in place that only allow connections from the colo. After mail has been filtered at the colo, the remote email server should forward all mail to the internal server. This server can then simply deliver the mail without further filtering.

Choosing a colo or ISP

When shopping for an ISP or colocation facility, do not overlook the details of the ***Service Level Agreement*** (***SLA***). This document describes the precise level of service that will be provided, including technical support, minimum uptime statistics, emergency contact procedures, and liability for unforeseen service outages. Remember that a promise of 99.99% uptime means that an ISP can be down for about seven hours **per month** before their SLA is violated. Keep this in mind, as it can and most probably will happen at least once as every colo provider experiences denial of service attacks, hardware failure, and simple human errors. While it's almost certain that your service will get interrupted eventually, the SLA will determine what course of action is available to you when it does.

You should also pay attention to the technical specifications of a potential data centre. Do they have backup power? Is the facility well ventilated? Do they have a multi-homed Internet connection with enough capacity to meet your needs as well as the needs of the rest of their customers? Do they use trusted equipment and professional installations, or is the data centre a haphazard tangle of wires and hardware? Take a tour of the facility and be sure you are comfortable with the organisation before making an agreement.

Billing considerations

Flat rate billing is when you're allocated a certain amount bandwidth, and are capped at this rate. For instance you may be allocated 1.5 Mbps of inbound and outbound bandwidth. You can use up to this amount for as long as you want with no fear of being billed extra at the end of the month. The only drawback to this is if you wish to have the ability to ***burst***, using more than the allocated 1.5 Mbps during busy times.

The ***95th percentile*** method allows for bursting. Bandwidth rates are polled every 5 minutes. At the end of the month, the top 95% spikes are removed and the maximum value is taken as the billed rate for the entire month. This method can lead to unexpectedly large bandwidth bills. For example, if you use 10 Mbps for 36 hours straight, and you use less than 1 Mbps for the rest of the month, you will be billed as if you used 10 Mbps for the entire month. The advantage of this method is that you can occasionally burst to serve much more traffic than normal, without being billed for the bursts. As long as your peak times fall in the top 5% of overall traffic, your bill will not increase.

You may also be billed by ***actual usage***, also known as ***by-the-bit***. ISPs may choose to bill you for every byte of traffic you transmit and receive, although this isn't commonly done for colo hosting. Actual usage billing is normally associated with a dedicated connection, such as a 100 Mbit or 1 Gbit line.

Protocol tuning

Most times, the default TCP parameters provide a good balance between performance and reliability. But sometimes it is necessary to tune TCP itself in order to achieve optimal performance. This is particularly important when using high latency, high throughput networks such as VSAT. You can drastically improve performance on such a link by eliminating unnecessary acknowledgments.

TCP window sizes

The ***TCP window size*** determines the size of a chunk of data that is sent before an ACK packet is returned from the receiving side. For instance, a window size of 3000 would mean that two packets of 1500 bytes each will be sent, after which the receiving end will ACK the chunk or request retransmission (and reduce the window size at the same time).

Large window sizes can speed up high-throughput networks, such as VSAT. For example, a 60 000 byte window size would allow the entire chunk to be sent to the receiving end before an ACK reply is required. Since satellite bandwidth has such high latency (about 1 second in Africa), using a small window size greatly reduces the available throughput. The standard window size of 1500 would require an ACK for each packet to be sent, introducing an additional 1 second of latency per packet. In this case, your available throughput would be roughly 1-2 Kbps maximum, even though the available bandwidth of the VSAT is much higher.

The TCP window size and other TCP tuning parameters can be easily adjusted in Linux and BSD.

Linux

RFC1323 defines two important high performance TCP extensions. It provides the TCP "Window Scale" option to permit window sizes of greater than 64 Kb. This will enable window scale support in Linux:

```
echo "1" > /proc/sys/net/ipv4/tcp_window_scaling
```

RFC1323 also establishes a mechanism for improving ***round trip time*** (***RTT***) calculations through the use of timestamps. A more accurate RTT means that TCP will be better able to react to changing network condition. This command enables timestamp support:

```
echo "1" > /proc/sys/net/ipv4/tcp_timestamps
```

To set the maximum size of the TCP receive and transmit windows respectively:

```
echo [size] > /proc/sys/net/core/rmem_max
echo [size] > /proc/sys/net/core/wmem_max
```

You can also adjust the default size of TCP receive and transmit windows:

```
echo [size] > /proc/sys/net/core/rmem_default
echo [size] > /proc/sys/net/core/wmem_default
```

As the available bandwidth increases, the transmit queue should also be increased. This is particularly important on very high bandwidth connections. The length of transmit queue can be set with **`ifconfig`**:

```
ifconfig eth0 txqueuelen [size]
```

FreeBSD

You can activate window scaling and timestamp options (as per RFC1323) with a single command in FreeBSD:

```
sysctl net.inet.tcp.rfc1323=1
```

To set the maximum TCP window size:

```
sysctl ipc.maxsockbuf=[size]
```

The default size of the TCP receive and transmit windows are set like this:

```
sysctl net.inet.tcp.recvspace=[size]
sysctl net.inet.tcp.sendspace=[size]
```

For more information about TCP window size and other protocol tuning, see:

- *http://proj.sunet.se/E2E/tcptune.html*
- *http://www.psc.edu/networking/projects/tcptune/*
- *http://www.hep.ucl.ac.uk/~ytl/tcpip/linux/txqueuelen/*

Link aggregation

By combining two or more network connections into a single logical connection, you can increase your throughput and add a layer of redundancy to your network. There are two mechanisms available to aggregate network links in Linux: via the bonding driver, and using routing.

Bonding

Bonding is one method for combining the throughput of two or more network connections. When using bonding, two or more physical interfaces are combined to create one virtual interface capable of the combined throughput. Bonding requires both sides of the connection to support the technology.

Let's assume we have two hosts that each have two 100 Mbit interfaces, eth0 and eth1. By bonding these two interfaces together, we can create a logical device (bond0) that provides a 200 Mbit link between the two hosts.

Run the following on both hosts.

```
# Make sure the bonding driver is loaded
modprobe bonding
# Set our IP, .10 for the first host, .11 for the second
ip addr add 192.168.100.10/24 brd + dev bond0
# Bring the interface up
ip link set dev bond0 up
# Add our slave interfaces
ifenslave bond0 eth0 eth1
```

If you use bonding, you should connect the bonded machines via cross-over cables, or use a switch that supports port trunking. Since both physical devices will use the same hardware (MAC) address, this can confuse conventional switches. For more information about bonding, see:

- *http://linux-net.osdl.org/index.php/Bonding*
- *http://linux-ip.net/html/ether-bonding.html*

Aggregate routing

You can also aggregate links by using routing alone. You can either use all links in a round-robin fashion (called ***equal cost routing***), or you can fail over to a second connection when the first becomes saturated. The first option is appropriate when the monetary cost of both links is equal. The second allows you to use a less inexpensive link for most of your traffic, and only fail over to the more expensive connection when the demand is high.

To perform equal cost routing between eth1 and eth2:

```
# ip route add default dev eth1 nexthop eth2
```

To only use eth2 when the traffic on eth1 saturates the link:

```
# ip route add default dev eth1 weight 1 nexthop eth2 weight 2
```

For more examples of how and when to use aggregate routing, see the Linux Advanced Routing & Traffic Control HOWTO at *http://lartc.org/lartc.html* .

DNS optimisation

Optimising a DNS cache will provide users with fast resolution of DNS queries, thus providing fast initial "startup" times for connections. Faster DNS response times make everything else on the network seem to "go faster."

Without memory restrictions, a DNS cache can run wild and use whatever memory is available. For example, a misconfigured BIND installation can easily eat up 4 GB of RAM when operating as a DNS cache server for a small ISP. To limit RAM consumption on BIND, add a **`max-cache-size`** option to your options section:

```
options {
  max-cache-size 16M;
}
```

DJBDNS uses 1 MB of memory for its cache by default, which may be a bit small for some installations. This will change the cache size to 16 MB:

```
echo 16000000 > /service/dnscache/env/CACHESIZE
echo 16777216 > /service/dnscache/env/DATALIMIT
svc -t /service/dnscache
```

It can be difficult to find the best location for hosting your public web server. If you host it on the local LAN, then local users will be able to access it very quickly, but connections from the public Internet will consume your bandwidth. If you host it at a colocation facility (as mentioned on page **205**), then the Internet at large will be able to access it very quickly, but local users (for example, students, and faculty) will need to use the Internet connection to access it.

One approach to this problem is to use a combination of mirroring (page **144**) and ***split horizon DNS***. You can mirror the contents of the web server and have a local copy as well as a public copy hosted at a colo. You can then configure your DNS server to return the IP address of the internal server when it is requested from the local LAN, and otherwise return the IP address at the colo.

In this example, hosts with a source IP address of 192.168.0.0/24 are given private responses to DNS queries, while all other addresses are given public responses.

```
acl internal {
    192.168.0.0/24;
}
```

```
view "internal" {
  match-clients {
    localhost;
    internal;
  };
  recursion yes;

  zone "." {
    type hint;
    file "caching/root.cache";
  };

  zone "companynet" in {
    type master;
    file "master/companynet-private";
  };

  zone "0.168.192.in-addr.arpa" {
    type master;
    file "master/0.168.192.in-addr.arpa";
  };
};


view "public" {

  recursion yes;

  zone "." {
    type hint;
    file "caching/root.cache";
  };

 zone "companynet" in {
    type master;
    file "master/companynet-public";
  };
};
```

This configuration will direct traffic to the most appropriate server.

You can use split horizon in any situation where clients should be redirected based on their source IP address. For example, you may wish to direct email clients to an internal email server when they are physically at the office, but to a different server when travelling. By assigning a different view for your mail host (one pointing at an internal IP address, the other pointing at a public IP) your users' email will continue to work without the need to change settings while travelling. A network using split horizon DNS is shown in Figure 6.7.

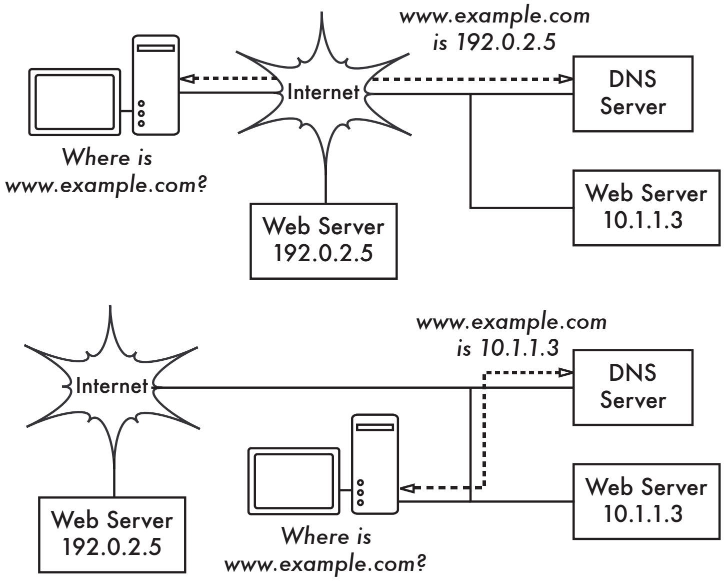


Figure 6.7: Split horizon directs users to the appropriate server depending on their source IP address.

Web access via email

While Internet continues to quickly expand, there remains a large community of users who only have access to e-mail. This can be because Internet service providers do not offer full Internet connections (due to inadequate infrastructure and low-bandwidth lines) or because the users simply cannot afford to pay for full Internet capabilities. Many of these users live in remote areas of developing countries and rely on e-mail not only for interpersonal communication, but also to access essential medical, business, and news information.

Despite the lack of broad and unlimited access to Internet in remote areas, there is still a plethora of creative and diverse content that scientists, artists, and people in general can share over, and retrieve from the net. Indeed, an important lesson learnt in recent years is that high-bandwidth access to the Internet is not essential for bridging the digital divide. To some extent, the exchange and transfer of knowledge and technology is possible using only e-mail (to retrieve general content Web pages) or Web-to-email gateways (to retrieve eJournals).

Some of these facilities, which mostly available for free, are discussed in this section. It is possible to access nearly any site on the Internet through e-mail.

There exists a moderated mailing list called ACCMAIL as a forum for communicating news, comments, and questions about e-mail only methods of accessing the Internet. To contribute to the discussion, e-mail to *accmail@listserv.aol.com* and to subscribe, send an e-mail to *listserv@listserv.aol.com* with SUBSCRIBE ACCMAIL as the message body.

www4mail

This is an open source application that allows you to browse and search the whole Web via e-mail by using any standard Web browser and any MIME (Multipurpose Internet Mail Exchange) aware e-mail program. E-mail messages sent to www4mail servers get automatically passed to the e-mail agent after selecting one or more buttons that link to other Web documents within a requested Web page. There are many options available, including user quotas, e-mail uuencoding reply, etc. All of these are described in the www4mail users' manual available at *http://www.www4mail.org/*.

By default, www4mail servers deliver web pages as attached HTML without including any images. In this way, returned e-mails are smaller, arrive faster, download quicker, and take up less space in an e-mail box. It is also possible to retrieve images via e-mail. Further information on the use of www4mail can be retrieved by sending an e-mail To: *www4mail@wm.ictp.trieste.it* and writing in the body of the message: "help" (without quotes).

To request the homepage of the International Centre for Theoretical Physics in Trieste, Italy, you would send an e-mail To: *www4mail@wm.ictp.trieste.it* and simply write in your e-mail message: *www.ictp.it* (without leading spaces). You should receive a reply in a few minutes, depending on Internet traffic.

Long URLs in the message body should be wrapped by using a backslash "\" without quotes. For example, the url:

```
http://cdsagenda5.ictp.it/full_display.php?smr=0&ida=a05228
```

...can be wrapped into:

```
http://cdsagenda5.ictp.it/full_display.php\
?smr=0&ida=a05228
```

To search in Yahoo for "peanuts" using www4mail, simply send another e-mail with the message: **`search yahoo peanuts`** .

web2mail

Just send an email to: *www@web2mail.com* with the web address (URL) of the web page you want as the Subject: of your email message. Another available web2mail e-mail addresses is: *web2mail@connectweb.de* .

PageGetter.com

Send an e-mail, including one or more URLs in the subject or body of your message, to *web@PageGetter.com* . You will automatically receive the full requested web page, complete with embedded graphics. Web pages with frames will be split into multiple e-mails, as many e-mail clients can not support frames. But if your e-mail client supports frames (i.e., Outlook Express) you can receive all frames in a single e-mail. In this case send the e-mail to *frames@PageGetter.com*. To receive a text-only version of the requested page, write instead to: *text@PageGetter.com* . This is especially useful for Personal Digital Assistants (PDAs), cell phones, and text only e-mail systems. You can also send an e-mail to: *HTML@PageGetter.com* to receive the full HTML page with no graphics.

GetWeb

GetWeb is another useful web to e-mail automatic service run by the Health-Net organisation . Send an e-mail To: *getweb@healthnet.org* with these three lines in the message body:

```
. begin
. help
. end
```

You will receive a reply with further instructions. If your mail software automatically inserts text (such as a signature) at the beginning or end of your message, an error will occur. To prevent this, GetWeb requires you to enclose retrieval commands in a "begin" and "end" block as shown above. The GetWeb server ignores any text that appears before or after this block.

To retrieve a particular website like "www.ictp.it" use the command get:

```
. begin
. get http://www.ictp.it
. end
```

For searching the words "malaria" and "Africa" with GetWeb use instead the strings:

```
. begin
. search yahoo malaria and Africa
. end
```

Time Equals Knowledge (TEK)

TEK is an open source web-to-email client that uses email as a transport mechanism for displaying web pages. It empowers low-connectivity communities by providing a full Internet experience, including web searching, caching, and indexing. Rather than viewing the web with an email client, users run TEK from within a normal web browser. The TEK distribution includes a customised version of Firefox that is pre-configured to talk to TEK rather than make requests directly from the Internet. It is also straightforward to configure other web browsers to talk to TEK.

TEK clients connect to a proxy server hosted at MIT that provides simplification and compression of pages, conversion of PS and PDF to HTML, and an interface to Google. The TEK client is released under the GNU GPL, and is free to download from *http://tek.sourceforge.net/*

Other useful web-to-email applications

Using GetWeb, the HealthNet SATELLIFE (*www.healthnet.org*) has pilot projects aimed at expanding access to health and medical information and supporting data collection and analysis through the use of handheld computers (PDAs) connected via the local GSM cellular telephone network in African regions.

The eJDS -free electronic Journal Delivery Service (*www.ejds.org*) is an application of www4mail geared to facilitate the access to current scientific literature free of cost in the fields of Physics and Mathematics. The goal is to distribute individual scientific articles via e-mail to scientists in institutions in developing countries who do not have access to sufficient bandwidth to download material from the Internet in a timely manner. Providing scientists with current literature supports their ongoing research and puts them with their peers in industrialised countries.

loband.org

A useful service offered by AidWorld that simplifies web pages in order to make them download faster over slow Internet connections is available at: *http://www.loband.org/.*

To use the service, simply type the web address of the page you want to visit into a web form and click the "Go" button. Once you start browsing through loband, the simplified page will contain the same text information as the original website and the formatting of the simplified page will be as similar to the original as possible, with colours and images removed.

Images contained in the original page are replaced by links containing an "i" in square brackets, e.g. [i-MainMap] or [i]. In either case, the original image can be viewed by clicking the link.

High Frequency (HF) networks

HF (***High-Frequency***) data communications enable radio transmission of data over a range of hundreds of miles.

HF radio waves reflect off the ionosphere to follow the curvature of the earth. The great advantage of HF is it can go the distance, leaping over obstacles in its path. Where HF wins the wireless game in range, but it loses in data capacity. Typical HF radio modems yield about 2400 bps. Two-way radio is the classic half-duplex medium of communication, so you are either transmitting or receiving, but not both at the same time. This, plus the robust error-checking protocols implemented by the modem hardware itself, means the actual link experience is on the order of 300 bps. It is not possible to use HF data communications for on-line browsing, chat, or video-conferencing. But HF is a workable solution for very remote places when using classic store-and-forward applications like text-based e-mail. One simply needs to pay close attention to the configuration and try to optimise as much as possible.

A basic HF data communications system consists of a computer, a modem, and a transceiver with an antenna. Modems used for HF radio vary in throughput and modulation technique, and are normally selected to match the capabilities of the radio equipment in use. At HF frequencies, Bell 103 modulation is used, at a rate of 300 bit/s.

Two distinct error and flow control schemes may be simultaneously used in HF networks. One is the Transport Control Protocol (TCP) which provides reliable packet delivery with flow and congestion control. The other is a radio link ***Automatic Repeat Request*** (***ARQ***) protocol which provides radio link layer frame error recovery.

The adaptive mechanisms in TCP are not optimum for HF networks, where link error rates are both higher and burstier than in the wired Internet. TCP has been designed under the assumption that packet losses are caused almost exclusively by network congestion, so TCP uses congestion avoidance mechanisms incorporating rate reduction and multiplicative increase of the retransmission timeout.

Application protocols (e.g., HTTP, FTP, and SMTP) may exchange many short commands and responses before each large file transfer (web page, file, or email message). In the course of transferring a single email message, an SMTP client will send at least four short SMTP commands and receive a similar number of short replies from the SMTP server. This high ratio of short to long messages is an important characteristic of Internet applications for the designer of a suitable wireless ARQ protocol.

The clients of a HF wireless ARQ protocol (e.g., IP) call upon the ARQ entity to deliver data reliably over a link. Just as for TCP, reliability is achieved by retransmissions. A typical sequence of events is a link setup handshake, followed by cycles of alternating data transmissions and acknowledgments from the destination. Each such cycle requires that the roles of physical transmitter and receiver be reversed twice. During each link turnaround some delay is introduced. This added cost for link turnarounds is a characteristic of wireless links, especially for HF radio links whose turnaround times range up to tens of seconds. However, the timeouts and other behaviour of a HF wireless ARQ protocol can be matched to known characteristics of a specific wireless channel for more efficient operation.

For more information about HF networks, see:

- NB6Z's Digital Ham Radio page, *http://home.teleport.com/~nb6z/about.htm*
- Introduction to Packet Radio, *http://www.choisser.com/packet/*
- Das Packet Radio Portal (German), *http://www.packetzone.de/*

Modem optimisation

If your only connection to the Internet is a 56k modem, there are a few things you can do to make the best possible use of the available bandwidth. As mentioned on page **199**, using SFQ with PRIO can help ensure that traffic is handled fairly on a congested link. Many of the other techniques in this book (such as using good caching software and doing far-side scrubbing) can help to ensure that the line is only used when it is absolutely needed. If you are using a dedicated modem connection, you will want to evaluate the performance when using various forms of compression.

Hardware compression

Standards such as V.42bis, V.44 and MNP5 are hardware compression techniques that are implemented in the modem itself. Under some circumstances, using hardware compression can significantly improve throughput. Note that the modems on both sides of the link must support the compression used.

- **V.42bis** is an adaptive data compression developed by British Telecom. The v.42bis algorithm continually estimates the compressibility of the data being sent, and whenever compression is not possible, it switches to a "transparent mode" and sends the data without compression. Files that have already been compressed (such as file archives or MP3 files) do not compress well, and would be sent without further processing. To enable V.42bis, use this modem string: **AT%C2**

- **V.44** is a data compression standard incorporated into the v.92 dial-up modem standard. V.44 can provide significantly better compression than V.42bis. V.44 was developed by Hughes Electronics to reduce bandwidth consumption on its DirecPC and VSAT products. Hughes released the V.44 algorithm in 1999, and it has been widely implemented as an alternative to V.42bis. To enable V.44, use this modem string: **`AT+DS44=3,0`**

- **MNP5** is short for Microcom Network Protocol 5. While it can provide better compression than V.42bis, it can only do so on compressible files. If you transfer data that is already compressed, MNP5 may actually decrease your throughput as it attempts to compress the data further. Many modems support the use of both MNP5 and V.42bis, and will select the appropriate compression algorithm depending on the data being sent. To enable MNP5 and V.42bis, the modem string is **`AT%C3`**. To enable just MNP5, use **`AT%C1`**.

Software compression

Enabling PPP compression can further increase throughput. Since PPP is implemented in a host computer (and not modem hardware), there are more aggressive compression algorithms available that can increase throughput by consuming slightly more CPU and RAM. The three most popular PPP compression algorithms are ***BSD Compression***, ***Van Jacobson***, and ***deflate***.

BSD Compression

The BSD Compression protocol (***bsdcomp***) is outlined in RFC1977. The algorithm is the same that is used in the ubiquitous UNIX compress command. It is freely and widely distributed and has the following features. It supports Automatic optimisation and disabling of compression when doing so would be ineffective. Since it has been widely used for many years, it is stable and well supported by virtually all PPP implementations.

To enable PPP in pppd 2.4.3, add the following directive to your command line (or options file): **`compress 15,15`**

VAN Jacobson (VJ)

Van Jacobson (VJ) header compression, detailed in RFC1144, is specifically designed to improve performance over serial links.

VJ compression reduces the 40 byte TCP/IP packet header to 3-4 bytes by tracking the state of TCP session on both sides of the link. Only the differences in header changes are sent in future transmissions. Doing this drastically improves performance and saves on average of 36 bytes per packet. VJ compression is included in most versions of PPP. To enable Van Jacobson compression, add this to your PPP configuration: **`vj-max-slots 16`**

Deflate

The deflate compression algorithm (RFC1979) is based on the Lempel-Ziv LZ77 compression algorithm. The GNU project and the Portable Network Graphics working group have done extensive work to support the patent-free status of the deflate algorithm. Most PPP implementations support the deflate compression algorithm.

To enable deflate compression, use this PPP option: **`deflate 15,15`**

Bandwidth accounting

Bandwidth accounting is the process of logging bandwidth usage on a per-IP or per-user basis. This can make it very easy to derive statistics about how much bandwidth individual users and machines are consuming. By using accounting techniques, you can generate hard statistics that can be used to enforce quotas or limits.

Squid bandwidth accounting

If you use authentication on your Squid cache, it can be desirable to perform bandwidth reporting or accounting broken down per user. If you intend to introduce quotas, or if you want to charge your users for access, you will have to keep records of each user's utilisation. There are many ways to do this by applying some simple scripts. This example is a quick recipe for basic bandwidth accounting. Feel free to adapt it to fit your own organisation.

To begin, you will obviously need a Squid server configured for user authentication (page **186**). You will also need a server with AMP (Apache, Mysql, PHP, and Perl) installed. If your network is relatively small, these can be installed on the same box as the Squid server. For busier environments, you should install these components on a separate machine. You will also need to install Perl (with the Perl DBI interface) on the Squid server. If you wish to implement this without the user front end, you only need Perl and a MySQL database.

First, download the example scripts from *http://bwmo.net/*. On the MySQL server, create the database tables by following the instructions provided in the **`mysql-setup.txt`** file included in the archive.

Next, copy the **`dolog.pl`** script to the MySQL server. This script will parse the log entries, and enter a summary of the user and site information into the database. Edit it to reflect the user name and password needed to connect to the MySQL database. Schedule a cron job to run the script every half hour or so.

On the Squid server, you will need to run the script **`squidlog.pl`**. You should insert the correct values for the MySQL server network address, user name, and password, as well as the correct path to the Squid log file. This script should be put in a cron job to run at some suitable interval (for example, every 15 or 30 minutes).

Finally, on the web server, make sure Apache is configured to use PHP with the LDAP extension installed. Extract and place the contents of the tar file into your web server's document directory. Edit **`config.php`** to include the correct values for the LDAP authentication source, and then visit the directory in your web browser.

You should now be able to log in as a user and view your browsing history. Even if you do not use the web application to give your users access to the database, you will still have a database that you can use for your own needs. This provides a perfect base to implement quotas, generate a report of the top 100 users and web sites visited, and so on. You can use any SQL tool you like (such as phpMyAdmin) to view the data, and can make simple scripts to provide reports and even take action when certain thresholds are crossed. When used in conjunction with a redirector (page **184**), you can automatically notify users when their bandwidth consumption reaches a specified limit.

Bandwidth accounting with BWM tools

As mentioned earlier, BWM Tools is a full-featured firewall, shaping, logging, and graphing system. It can be integrated with RRDtool to generate live graphs, including bandwidth utilisation and statistics.

Setting up of BWM Tool flows for reporting purposes is very simple. To log data every 5 minutes, add this to your configuration file:

```
<traffic>
    <flow name="smtp_inbound" max-rate="16384" report-timeout="300">
        inbound;
    </flow>
    <flow name="smtp_outbound" max-rate="8192" report-timeout="300">
        outbound;
    </flow>
</traffic>
```

If you want to log directly to an RRD file, you can set it up like this:

```
<traffic>
    <flow name="smtp_inbound" max-rate="16384" report-timeout="300"
report-format="rrd">
```

```
        inbound;
    </flow>
    <flow name="smtp_outbound" max-rate="8192" report-timeout="300"
report-format="rrd">
        outbound;
    </flow>
</traffic>
```

You can now use RRDtool (page **83**) or an integrated package (such as Cacti, page **84** or Zabbix, page **89**) to display graphs based on the data files generated above.

Linux interface bandwidth accounting with RRDtool

Using a simple Perl script with RRDtool integration, you can monitor interface bandwidth usage and generate live graphs by polling an interface directly on a Linux server.

You will need this script:

http://www.linuxrulz.org/nkukard/scripts/bandwidth-monitor

Edit the script and change **`eth0`** to match the interface you wish to monitor. You create the RRD files by running the script with the **`--create-rrd`** switch:

```
bandwidth-monitor --create-rrd
```

This will create the file **`/var/log/bandwidth.rrd`**. To start the bandwidth monitor, run this command:

```
bandwidth-monitor --monitor
```

The script will keep running and will log traffic statistics every five minutes. To view live statistic graphs, download this cgi and put it into your web server's **`cgi-bin`** directory:

http://www.linuxrulz.org/nkukard/scripts/stats.cgi

Now visit that script in your web browser to see the flow statistics for the interface being monitored.

VSAT optimisation

(Note: This portion was adapted from the VSAT Buyer's Guide, and is used with permission. This guide is also released under a Creative Commons license, and is full of valuable information that is useful when comparing satellite communications systems. For the full guide, see *http://ictinafrica.com/vsat/*).

VSAT is short for Very Small Aperture Terminal, and is often the only viable connectivity option for very remote locations. There are a whole host of technical considerations you will need to make when buying a VSAT. Most of them involve making trade-offs among the technical characteristics that give you what you want and what you can afford. The common considerations you may be forced to make are:

- Whether to use inclined orbit satellites
- Whether to use C or Ku band
- Whether to use shared or dedicated bandwidth

The technology you choose will necessarily need to balance operating costs with performance. Here are some points to keep in mind when choosing a VSAT provider.

Use of inclined orbit satellite

The price of bandwidth on inclined orbit satellites is usually much lower since these satellites are nearing their end of life. The downside is that it requires a dish with tracking capabilities that can be very expensive. The high capital costs associated with the expensive antenna can be offset by lower operating costs, but only if you are purchasing large amounts of bandwidth. You should therefore make sure that you carefully consider both your capital and operating costs over the period you intend to operate the VSAT. Of course, remember to ascertain the exact remaining life of the satellite, when making this consideration. If you decide to opt for inclined orbit capacity, caution is advised as the service can be down for a while in the event that you are running mission critical applications.

C band, Ku band, and Ka band

One of the big decisions you are likely to encounter when buying a VSAT is whether to use C band or Ku band. In order to enable you to arrive at an informed decision, we have briefly presented the advantages and disadvantages of each band.

Advantages of using C band

- **C band is less affected by rain.** If you happen to live in a high rain-fall area such as the tropics and your VSAT applications are “mission critical," in other words, you want your system available all the time, you can opt for C band over Ku band. However, this does not exclude the use of Ku band systems in the tropics especially for home TV and Internet access since these are not mission critical applications and the consumers can live with a few hours of intermittent service.

- **C band systems have been around longer than Ku band systems and thus rely on proven technology.** However, Ku band systems seem to be overtaking C band systems as the preferred technology in the home consumer business in the last few years. Note that Ku band dishes are more likely to be smaller and therefore cheaper for any given application, because of Ku band's higher frequencies. You should also bear in mind that Ku band bandwidth prices are higher than C band prices and therefore any savings on capital costs could be offset by higher operating costs.

- **C band satellite beams have large foot prints.** The global beam for C band covers almost a third of the earth's surface. If you are looking for single satellite hop operation (e.g. for real time applications such as VoIP or video conferencing) to connect locations far apart from one another, you may be forced to choose the wider coverage C band beams. However, the latest satellites launched have large Ku band beams covering entire continents. You should also note that two beams on the satellites can be connected through a method called "cross strapping" thus allowing two or more locations covered by two separate beams to be connected in a single hop.

Disadvantages of C band

- **C band requires the use of larger dishes.** They can be quite cumbersome to install and are more expensive to acquire and transport.
- **C band systems share the same frequency bands as allocated to some terrestrial microwave systems.** As such, care must be taken when positioning C band antennas in areas where terrestrial microwave systems exist (for example TV or radio stations). For this reason, C band satellite transponder power is deliberately limited during the satellite's design and manufacture according to sharing criteria laid down by the ITU, leading to a requirement for larger dishes on the ground.

Advantages of Ku band

- **Ku band systems require smaller dishes.** Because of their higher satellite transponder power and higher frequencies, they can use smaller, cheaper antennas on the ground and therefore reduce startup and transport costs.
- **The smaller Ku Band dishes can be easily installed on almost any surface.** For example, they can be installed on the ground, mounted on roofs, or bolted to the side of buildings. This is an important consideration for areas with limited space.

Disadvantages of Ku band

- **Ku band systems are more affected by rainfall.** Because of their higher operating frequencies, they are usually considered unsuitable for mission critical applications in the tropics, unless specific measures are taken to provide for the added rain attenuation. For example, this can be overcome by using larger dishes. This drawback has also been slightly offset by the higher power satellites being manufactured today. As noted above, Ku band systems are gaining popularity even in the tropics for home use where users can survive a few hours of intermittent service a month.
- **Ku band satellite systems usually have smaller beams covering a small surface of the earth.** Therefore if you intend to cover two locations a large distance apart, within a single hop or with a broadcast system, you may need to consider C band systems.
- **Ku band bandwidth is more expensive that C band bandwidth.** As noted above, the savings in capital cost you gain using Ku band's smaller antennas may be negated by the higher operating costs imposed by high bandwidth prices.

Advantages of Ka band

- **Ka band dishes can be much smaller than Ku band dishes.** This is due to the even higher Ka band frequencies and higher satellite power. The smaller dishes translate to lower start up costs for equipment.

Disadvantages of Ka band

- **The higher frequencies of Ka band are significantly more vulnerable to signal quality problems caused by rainfall.** Therefore, Ka band VSATs are usually unsuitable for mission critical or high availability systems in the tropical and sub-tropical regions without the provision of measures to combat adverse weather conditions.
- **Ka-band systems will almost always require tracking antennas.** This adds to complexity and startup costs.
- **Ka band bandwidth is more expensive than C band or Ku band bandwidth.**
- **The Ka band is currently unavailable over Africa.**

Shared vs. dedicated bandwidth

It is critical for you to decide whether you will accept shared or dedicated bandwidth. Shared bandwidth refers to bandwidth that is shared with other customers of your service provider. Dedicated bandwidth is "committed" solely to you. Shared bandwidth is obviously cheaper than dedicated bandwidth because you are also sharing the cost of the bandwidth among other users. Unfortunately, some service providers pass off shared bandwidth as dedicated bandwidth and charge you rates equivalent to those for dedicated bandwidth. You therefore have to be clear about what you are buying.

Shared bandwidth is desirable when you will not be using all the bandwidth all the time. If your primary applications will be email and web surfing and you do not have many users, e.g. a community telecentre, then shared bandwidth may well work for you. However, if you have a large volume of users accessing the system throughout the day, or if you intend to run real time applications such as telephony or video conferencing, then you will need dedicated bandwidth.

There is one instance when you should consider shared capacity even when you have heavy users and real time applications. This is the situation in which you own the entire network. You would essentially be buying a chunk of dedicated bandwidth and then sharing its capacity among your network. The reasoning behind this is that if all VSATs are part of the same network, with the same profile of user, then you are likely to have instances when capacity would be unused. For instance, not all the VSATs in your network may be making voice calls or participating in video conferencing all the time. This method is especially suited to global organisations with offices in different time zones.

There are three key metrics you will need to consider when purchasing shared bandwidth:

The Contention Ratio

Contention is a term that comes from terrestrial Internet systems such as ***Digital Subscriber Link*** (***DSL***) and refers to sharing. The contention ratio is the number of users sharing the bandwidth. Obviously, the more users sharing the bandwidth, the less bandwidth you get if they are all online. For instance, if you are sharing bandwidth with a capacity of 1 Mbps among 20 customers (contention ratio of 20:1), then your maximum connection speed when all the customers are using the bandwidth is 50 kbps, equivalent to a dial up modem connection. If, however, the contention ratio is 50:1, or 50 customers sharing the connection, then your maximum speed when all customers are using the system is 20 kbps. As you can imagine, how much of the 1 Mbps promised by the service provider you actually get depends on the contention ratio. Contention is also called “over booking” or “over selling” capacity.

Committed Information Rate (CIR)

Even with shared bandwidth capacity, your service provider may guarantee you certain minimum capacity at all times. This guaranteed capacity is the CIR. In our example above using a contention ratio of 20:1, this CIR would be 50 kbps, even though you are quoted a bandwidth capacity of 1 Mbps.

Bursting capacity

Bursting refers to the ability of a VSAT system to utilise capacity above and beyond its normal allocation. Bursting is only possible when you purchase shared bandwidth. If your service provider has implemented bursting, a portion or all of the shared bandwidth capacity will be pooled. For instance, several portions of 1 Mbps may be pooled together. When other customers are not using their capacity, you may be able to ***burst***, or use more than your allocated capacity. Note that bursting also only occurs when there is ‘free’ or available capacity in the pool. The amount of additional or burst capacity to which any VSAT station sharing the pool is entitle to is limited to a set maximum, usually less than the total pool capacity to ensure that there is always capacity available for other VSAT stations.

In summary, when purchasing shared capacity, you should ask your service provider to specify the contention ratio, your CIR, and how much bursting capacity you can get.

TCP/IP factors over a satellite connection

A VSAT is often referred to as a ***long fat pipe network***. This term refers to factors that affect TCP/IP performance on any network that has relatively large bandwidth, but high latency. Most Internet connections in Africa and other parts of the developing world are via VSAT. Therefore, even if a university gets its connection via an ISP, this section might apply if the ISP's connection is via VSAT. The high latency in satellite networks is due to the long distance to the satellite and the constant speed of light. This distance adds about 520 ms to a packet’s round-trip time (RTT), compared to a typical RTT between Europe and the USA of about 140 ms.

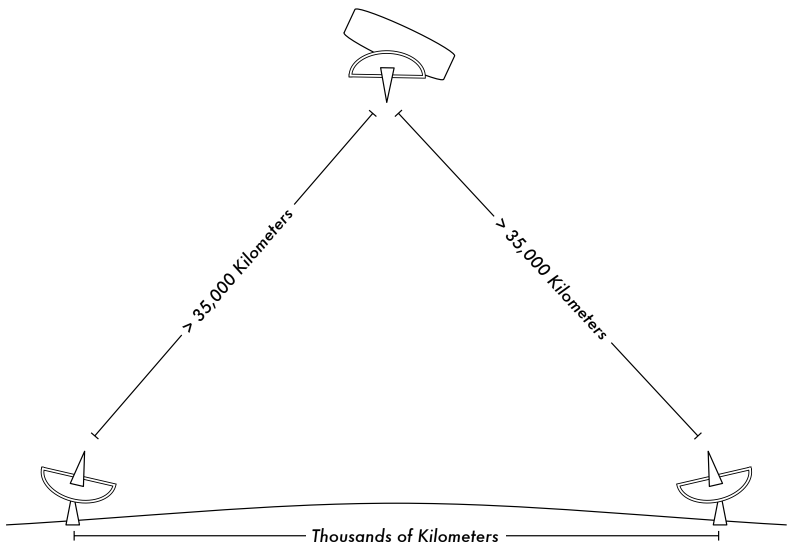


Figure 6.8: Due to the speed of light and long distances involved, a single ping packet can take more than 520 ms to be acknowledged over a VSAT link.

The factors that most significantly impact TCP/IP performance are **long RTT**, **large bandwidth delay product**, and **transmission errors**.

Generally speaking, operating systems that support modern TCP/IP implementations should be used in a satellite network. These implementations support the RFC1323 extensions:

- The ***window scale*** option for supporting large TCP window sizes (larger than 64KB).
- ***Selective acknowledgment*** (***SACK***) to enable faster recovery from transmission errors.
- Timestamps for calculating appropriate RTT and retransmission timeout values for the link in use.

Long round-trip time (RTT)

Satellite links have an average RTT of around 520ms to the first hop. TCP uses the slow-start mechanism at the start of a connection to find the appropriate TCP/IP parameters for that connection. Time spent in the slow-start stage is proportional to the RTT, and for a satellite link it means that TCP stays in slow-start mode for a longer time than would otherwise be the case. This drastically decreases the throughput of short-duration TCP connections. This is can be seen in the way that a small website might take surprisingly long to load, but when a large file is transferred acceptable data rates are achieved after awhile.

Furthermore, when packets are lost, TCP enters the congestion-control phase, and owing to the higher RTT, remains in this phase for a longer time, thus reducing the throughput of both short- and long-duration TCP connections.

Large bandwidth-delay product

The amount of data in transit on a link at any point of time is the product of bandwidth and the RTT. Because of the high latency of the satellite link, the bandwidth-delay product is large. TCP/IP allows the remote host to send a certain amount of data in advance without acknowledgment. An acknowledgment is usually required for all incoming data on a TCP/IP connection. However, the remote host is always allowed to send a certain amount of data without acknowledgment, which is important to achieve a good transfer rate on large bandwidth-delay product connections. This amount of data is called the ***TCP window size***. The window size is usually 64KB in modern TCP/IP implementations.

On satellite networks, the value of the bandwidth-delay product is important. To utilise the link fully, the window size of the connection should be equal to the bandwidth-delay product. If the largest window size allowed is 64KB, the maximum theoretical throughput achievable via satellite is (window size) / RTT, or 64KB / 520 ms. This gives a maximum data rate of 123KB/s, which is 984 Kbps, regardless of the fact that the capacity of the link may be much greater.

Each TCP segment header contains a field called ***advertised window***, which specifies how many additional bytes of data the receiver is prepared to accept. The advertised window is the receiver's current available buffer size. The sender is not allowed to send more bytes than the advertised window. To maximise performance, the sender should set its send buffer size and the receiver should set its receive buffer size to no less than the bandwidth-delay product. This buffer size has a maximum value of 64KB in most modern TCP/IP implementations.

To overcome the problem of TCP/IP stacks from operating systems that don't increase the window size beyond 64KB, a technique known as ***TCP acknowledgment spoofing*** can be used (see Performance Enhancing Proxy, below).

Transmission errors

In older TCP/IP implementations, packet loss is always considered to have been caused by congestion (as opposed to link errors). When this happens, TCP performs congestion avoidance, requiring three duplicate ACKs or slow start in the case of a timeout. Because of the long RTT value, once this congestion-control phase is started, TCP/IP on satellite links will take a longer time to return to the previous throughput level. Therefore errors on a satellite link have a more serious effect on the performance of TCP than do errors on low latency links. To overcome this limitation, mechanisms such as ***Selective Acknowledgment*** (***SACK***) have been developed. SACK specifies exactly those packets that have been received, allowing the sender to retransmit only those segments that are missing because of link errors.

The Microsoft Windows 2000 TCP/IP Implementation Details White Paper states

> "*Windows 2000 introduces support for an important performance feature known as Selective Acknowledgment (SACK). SACK is especially important for connections using large TCP window sizes.*"

SACK has been a standard feature in Linux and BSD kernels for quite some time. Be sure that your Internet router and your ISP's remote side both support SACK.

Implications for universities

If a site has a 512 Kbps connection to the Internet, the default TCP/IP settings are likely sufficient, because a 64 KB window size can fill up to 984 Kbps. But if the university has more than 984 Kbps, it might in some cases not get the full bandwidth of the available link due to the "long fat pipe network" factors discussed above. What these factors really imply is that they prevent a single machine from filling the entire bandwidth. This is not a bad thing during the day, because many people are using the bandwidth. But if, for example, there are large scheduled downloads at night, the administrator might want those downloads to make use of the full bandwidth, and the "long fat pipe network" factors might be an obstacle. This may also become critical if a significant amount of your network traffic routes through a single tunnel or VPN connection to the other end of the VSAT link.

Administrators might consider taking steps to ensure that the full bandwidth can be achieved by tuning their TCP/IP settings. If a university has implemented a network where all traffic has to go through the proxy (enforced by network layout), then the only machines that make connections to the Internet will be the proxy and mail servers.

For more information, see *http://www.psc.edu/networking/perf_tune.html* .

Performance-enhancing proxy (PEP)

The idea of a Performance-enhancing proxy is described in RFC3135 (see *http://www.ietf.org/rfc/rfc3135*), and would be a proxy server with a large disk cache that has RFC1323 extensions, among other features. A laptop has a TCP session with the PEP at the ISP. That PEP, and the one at the satellite provider, communicate using a different TCP session or even their own proprietary protocol. The PEP at the satellite provider gets the files from the web server. In this way, the TCP session is split, and thus the link characteristics that affect protocol performance (long fat pipe factors) are overcome (by TCP acknowledgment spoofing, for example). Additionally, the PEP makes use of proxying and pre-fetching to accelerate web access further.

Such a system can be built from scratch using Squid, for example, or purchased "off the shelf" from a number of vendors. Here are some useful links to information about building your own performance enhancing proxy.

- "Enabling High Performance Data Transfers," Pittsburgh Supercomputing Center: *http://www.psc.edu/networking/projects/tcptune/*
- RFC3135, "Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations."
- ***PEPsal***, a Performance Enhancing Proxy implementation released under the GPL: *http://sourceforge.net/projects/pepsal/*

Resources

- Adzapper: *http://adzapper.sourceforge.net/*
- Authentication in Squid, *http://www.squid-cache.org/Doc/FAQ/FAQ-23.html*
- Cache heirarchies with Squid, *http://squid-docs.sourceforge.net/latest/html/c2075.html*
- DansGuard: *http://dansguardian.org/*
- dnsmasq caching DNS and DHCP server, *http://thekelleys.org.uk/dnsmasq/doc.html*
- Enhancing International World Wide Web Access in Mozambique Through the Use of Mirroring and Caching Proxies, *http://www.isoc.org/inet97/ans97/cloet.htm*
- Fluff file distribution utility, *http://www.bristol.ac.uk/fluff/*
- Lawrence Berkeley National Laboratory's TCP Tuning Guide, *http://dsd.lbl.gov/TCP-tuning/background.html*
- Linux Advanced Routing & Traffic Control HOWTO, *http://lartc.org/lartc.html*
- Microsoft Internet Security and Acceleration Server, *http://www.microsoft.com/isaserver/*
- Microsoft ISA Server Firewall and Cache resource site, *http://www.isaserver.org/*

- Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations, *http://www.ietf.org/rfc/rfc3135*
- PF: The OpenBSD Packet Filter FAQ, *http://www.openbsd.org/faq/pf/*
- Pittsburgh Supercomputing Center's guide to Enabling High Performance Data Transfers, *http://www.psc.edu/networking/perf_tune.html*
- SquidGuard: *http://www.squidguard.org/*
- Squid web proxy cache, *http://squid-cache.org/*
- TCP Tuning and Network Troubleshooting by Brian Tierney, *http://www.onlamp.com/pub/a/onlamp/2005/11/17/tcp_tuning.html*
- ViSolve Squid configuration manual, *http://www.visolve.com/squid/squid24s1/contents.php*
- The VSAT Buyer's guide, *http://ictinafrica.com/vsat/*
- Wessels, Duane. *Squid: The Definitive Guide*. O'Reilly Media (2004). *http://squidbook.org/*

7
Case Studies

You may have noticed that there is not a single definitive set of instructions on "how to fix your slow Internet connection" anywhere in this book. This is because every environment has different needs and resources, and the techniques used for bandwidth management will vary greatly depending on your local environment. To put this into perspective, here are some real life examples of how the methods and techniques presented in this book have solved real world bandwidth management problems.

KENET, Kenya

Established in 1999, The Kenya National Education Network (KENET) connects educational institutions and research centres with the goal of broadcasting knowledge throughout the country. Currently, there are 13 members directly connected to the main node in Nairobi and 40 additional members participating in the network via Kenya's backbone Telco. The main node consists of a 3 Mbps uplink via a leased line, and a 3 Mbps downlink via VSAT.

Clearly, available bandwidth is limited. Members typically range from 64Kbps to 960 Kbps of bandwidth usage, many of whom connect to KENET via leased lines from the local Telco. These leased lines then terminate at KENET on E1 lines with 2 Mbps capacity. In addition, some of the members have their own VSAT downlinks and only uplink via KENET.

Most sites do not have skilled staff managing the network, and are often unaware of the dangers and costs involved when bandwidth is mismanaged.

Problems

Initially, certain networks were set up improperly due to a lack of trained staff. For example, one network consisting of 20 machines was operating without a caching proxy server. Servers and routers were often poorly configured. The major consequence of this neglect was that the bandwidth became widely mismanaged – usually clogged with peer-to-peer traffic, spam, and viruses.

In turn, this led to KENET IP addresses being blacklisted on the Internet, making services and sites completely unreachable. The already low capacity bandwidth became completely unusable for most institutions.

Analysis

An analysis of each institution's traffic behaviour was made using MRTG (traffic grapher) and FlowC (graphical packet analyser). This established a baseline so that changes to the network could be monitored for their effects. In addition to investigating the problematic institutions' network configurations, the software in use at each site was examined.

The minimum skill level required of staff at participating institutions was determined, and the possibility of replacing some software with FOSS (Free and Open Source Software) was researched.

Solutions

A regular training program for staff was started in 2004. Training topics included network management, security, and monitoring using FOSS tools. Training sessions also educated staff to interpret traffic graphs and detect anomalies.

On the networks, aggressive Access Control Lists (ACLs) were installed on routers to restrict access to include only approved services. Custom servers were set-up by KENET to address specific problems at each institution. While the configuration details varied between locations, a decision was made to standardise the network on a uniform platform: Firewalled FreeBSD. This solution proved to be stable, secure, reliable, and FREE! Uniform solutions were chosen for each software component as well: MTA (Qmail), Spam Filter (Spamassassin), Mail Antivirus (ClamAV), Proxy (Squid), and BSD traffic shaping.

Here are the results of applying good bandwidth management techniques to three sites within KENET.

Site One: firewall & proxy server

The institution at Site One had 960 Kbps of available bandwidth. This institution's bandwidth needed to accommodate approximately 300 computers connected to Internet, plus two Mail and two DNS servers. Although a proprietary proxy server and firewall were in place (Microsoft Internet Security and Acceleration Server, ISA), the firewall was compromised and used as a platform for sending spam.

Once the firewall was compromised, network performance slowed dramatically and the institution's IP address was blacklisted. Uplink traffic consisted mainly of 700 Kbps of spam mail originating from the firewall. The ISA proxy server was replaced with a FreeBSD 5.4 server, configured and installed to serve as firewall and proxy (using native IPF firewall tools and Squid).

The results of this change were immediate and dramatic: Internet speed perceptibly improved as students and staff reported better performance. The network became efficient, with a maximum of only 400Kb uplink traffic. With monitoring in place, bandwidth graphs easily showed legitimate patterns and traffic. Finally, with the spam problem under control, the institution's IP address was removed from Internet blacklists.

Site Two: proxy & mail server

The institution at Site Two had 128Kbps of bandwidth, connecting approximately 50 computers networked over two sites. As with Site One, the proxy and mail server was compromised, allowing spam, peer-to-peer traffic, and viruses to penetrate. Uplink traffic was completely filled with 128Kbps of spam email.

Since the uplink was flooded, the network slowed dramatically and the institution's IP addresses were blacklisted. Again, the solution was to set up a Squid proxy on a firewalled FreeBSD server. Qmail, SpamAssassin, and ClamAV were installed on same server. ClamAV and SpamAssassin were configured to check incoming and outgoing mail viruses and spam.

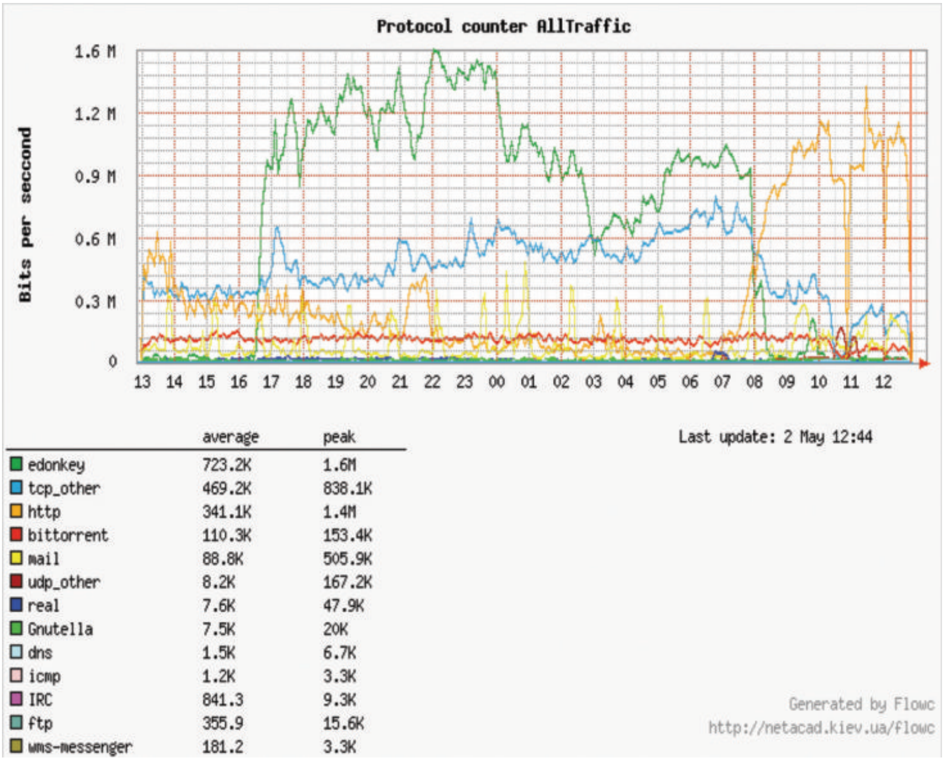
As a result, viruses and spam were not transmitted, so the uplink traffic was no longer clogged. The overall Internet speed improved, and the monitoring graphs showed clean (and expected) traffic. The institution's IPs were also removed from Internet blacklists.

This server has functioned problem-free for more than 10 months and requires virtually no maintenance. Clam Antivirus updates itself every night, so no administrative interaction was required to obtain current virus signatures.

Site Three: FOSS traffic shaper

A more advanced traffic shaper was installed at this KENET site. Here, a firewall is used that is implemented with PF using ALTQ on FreeBSD. It can queue and assign priority to specific traffic types, and is implemented using FOSS.

The system is a 2 GHz Pentium IV, with 256 MB RAM, 4 NICs and 40 GB storage running FreeBSD. Even with these modest hardware requirements, the system easily handles 3 Mbps of traffic, both up and down. By using FlowC, traffic utilisation is easily broken down by application.



	average	peak
🟩 edonkey	723.2K	1.6M
🟦 tcp_other	469.2K	838.1K
🟧 http	341.1K	1.4M
🟥 bittorrent	110.3K	153.4K
🟨 mail	88.8K	505.9K
🟥 udp_other	8.2K	167.2K
🟦 real	7.6K	47.9K
🟩 Gnutella	7.5K	20K
⬜ dns	1.5K	6.7K
⬜ icmp	1.2K	3.3K
🟪 IRC	841.3	9.3K
🟩 ftp	355.9	15.6K
🟫 wms-messenger	181.2	3.3K

Figure 7.1: Note the dramatic shift from eDonkey to HTTP traffic at 08:00.

Notice the change in the graph when the shaper was activated at 8:00. With this FOSS server in place, efficiency in use of available bandwidth can be verified. Peer-to-peer traffic is under control, browsing is faster as HTTP now has priority, and the traffic graphs all show clean and expected traffic patterns.

--Kevin Chege

Aidworld in Accra, Ghana

In 2006, Aidworld spent three months in Ghana optimising two web portals for use over the kind of Internet connections found in research and higher education institutions there. As part of this project, we visited with members of many Ghanaian institutions to ask about their experiences using the Internet. As a result of these conversations, we gained a general understanding of the issues concerning bandwidth management and the effects of using bandwidth management strategies.

We learned that larger organisations were able to afford to employ a number of skilled network professionals to manage their network. Smaller organisations did not always have this luxury. Many institutes employ either solely part-time network administrators or none at all. The second condition is true of institutes where the network was installed by its users; people who have neither the time, nor the training, to manage such networks. Most organisations have no budget for antivirus software. Usually, Windows updates are often not installed, automatic updating is not enabled, and service packs are not applied. Existing bandwidth problems are compounded by computers infected with worms (network viruses). The computers could have avoided infection, if automatic updating and virus detection had been enabled.

As a result of bandwidth management issues, many of the institutes we visited had little or no Internet access during regular business hours. Staff members often modified their work schedules to compensate, working long, inconvenient hours in order to download important documents at night, or in the early morning.

It is often assumed that problems with Internet access and speed can only be solved by adding bandwidth. However, in our experience, we've found this is not necessarily the case. Using bandwidth management strategies are more effective than simply adding bandwidth. In the case of a network that is overloaded by virus traffic, additional bandwidth will have very little effect on the user's experience.

This case study demonstrates the need for antivirus and malware policy as a central aspect of BMO policy and the benefits of enforcing them.

The Institute for Scientific and Technological Information (INSTI) is based in Accra, the capital of Ghana. It provides information services, such as a library and Internet access, to research institutes under the Council for Scientific and Industrial Research (CSIR) throughout Ghana. INSTI acts as a hub for several organisations, providing them with access to the Internet through its own network. It also provides access to visiting students and researchers through an Internet cafe. INSTI has approximately 50 staff librarians, researchers and administrators. It has about 50 computers, 20 of which comprise the Internet cafe.

At one time, the user's Internet experience of at INSTI was very slow, particularly during the day. For instance Benjamin, a librarian, would regularly have to come into work at 5:00 or 6:00 am in order to download journal articles because those were the only times that access was fast enough or available. This situation worsened until access to the Internet was stopped altogether.

Diagnosing the problem began with investigating the network traffic. It was noticed that all of the network activity lights on the switches, firewall router, and ADSL router were blinking constantly: all signs of a very high load. The traffic graphs on the ADSL router showed that the outbound bandwidth used was much higher than the connection could support, and remained so constantly. On the other hand, incoming bandwidth use was zero. That was unusual.

It was suspected that there was a problem with the ADSL router. Rebooting the router would allow Internet access, but only for a short time. The router was apparently crashing every few minutes. When unplugged from its main network, the router stopped crashing. This condition suggested an underlying cause of unusual network traffic.

The firewall (IPcop) was unable to determine which local machines were generating the traffic, or what kind of traffic it was. By inserting a laptop, configured as a transparent bridge, between the firewall and the rest of the network, it became possible to see the Internet addresses of the machines sending traffic, rather than the external address of the firewall after it had subjected them to ***network address translation*** (***NAT***). Without the internal address, it would not be possible to identify which machine was sending the traffic.

When we used a packet sniffer (tcpdump) to look at the network traffic, it was immediately apparent that most of it was UDP packets. These were sent by several local machines to a vast number of remote IP addresses, destined for one or two well-known ports. The above conditions are a classic signature of Internet worms. Most of the infected machines were in the Internet cafe. Without strictly enforcing antivirus, anti-spyware, or update policy within the Internet cafe, several of these machines had become badly infected and were flooding the network with virus traffic.

To remedy the situation, antivirus and anti-spyware software (SpyBot S&D) were installed on all machines. They were also configured for automatic Windows updates. Having made these changes, the outgoing network traffic almost reached zero, and incoming was well within the capacity of the link. Web pages loaded quickly, ping response times dramatically went down, and the router stopped crashing. It became possible to download articles during the day which meant that the staff no longer needed to work unsocial hours in order to use the Internet.

--Alan Jackson

BMO in the UK

The following are two case studies derived from published information about Internet access in British universities. These case studies serve as useful reference points for the implementation of successful bandwidth management strategies.

JANET, UK

The Joint Academic Network (JANET) is the UK's education and research network. It connects UK universities to each other, and to the Internet. It has equivalents in many countries, such as KENET in Kenya. It serves over 18 million users, according to its website. JANET is operated and developed by a government funded organisation, the United Kingdom Education and Research Networking Association (UKERNA).

The Problem

JANET is a large network with high demand due to the number of students who gain easy, free, high speed access to it as part of their university course. They have limited funding, and in the past were not able to supply enough Internet bandwidth to meet the demand. This resulted in poor performance at peak times, and frequent outages.

The Solutions

JANET's bandwidth management strategy has two parts: an acceptable use policy (AUP), and technical measures to enforce that policy. The policy says that JANET may be used "for any purpose that is legal, that is socially acceptable to the community JANET serves, and that does not cause degradation of the performance of the network beyond that which might reasonably be expected." Degradation of network performance is exactly what bandwidth management aims to prevent.

JANET's acceptable use policy also prohibits its use for illegal file sharing, denial of service attacks, and spamming, which are some of the main bandwidth hogs on unmanaged networks. The policy gives UKERNA staff the right to shut down a JANET member who cannot or will not manage his or her own use of the bandwidth effectively, or who causes problems for JANET or its members.

A schematic map of JANET shows that it has two connections to the Internet on opposite sides of a large "ring" around the country. If that connection were to become completely full, the resulting congestion would make Internet access slow and unreliable for all JANET members. JANET has chosen to make the connections to "external links" (i.e., the Internet) equal or larger in capacity than the connections to the regional networks, which in turn serve the universities. This should mean that institutions' own connections to JANET are a bottleneck that restricts their bandwidth to the Internet and to each other, and no one institution can flood the connection.

Institutions will find that this bottleneck gives them the power and responsibility to manage the bandwidth use on their own connection. UKERNA recommends that "all JANET connected organisations should formulate a local AUP and ask staff and students to sign a declaration to confirm that they will abide by its rules" and "carry out their own internal monitoring of their network connection." A case study of how Blackburn College manages its own bandwidth over its JANET connection is given below.

JANET runs a system called Netsight. It monitors and records performance information about their network, including link status and bandwidth usage graphs. Netsight can detect and "highlight abnormal traffic levels on a site's access link that may be a result of illegal activity." UKERNA also recommends that "organisations record sufficient information about the use of their networks and maintain tools to be able to investigate and deal with problems."

JANET also offers a caching service for the Domain Name System (DNS). It is used to convert website names into addresses on the network. Caching makes DNS queries faster, more reliable and reduces the bandwidth used by DNS.

The Joint Information Systems Committee, one of the members and supporters of JANET, operates a web cache service that is used by many other members, including some that peer their own caches with it for greater efficiency. A 1996 report about this service says:

> "*The UK academic network has always offered high-performance connections within the country but comparatively slow international links. The desire to make best use of the scarce international bandwidth led to an early interest in web caching and... a national academic cache was set up... in 1995. The cache... now uses six separate computers located at sites in Canterbury and Leeds.*"

The cache has a hit rate of between 55% and 60% (documents served from the cache rather than the original website). They claim that it is one of the highest hit rates for any cache in the world, and close to the theoretical maximum. They also say that "caches act as bandwidth multipliers. For every megabyte of requests arriving at the national cache, only 400 kilobytes of traffic are passed on to further networks."

In summary, JANET employs an acceptable use policy, user bandwidth limiting, and network monitoring to manage their network and bandwidth. They offer some services to their members which help the members to monitor, control, and reduce their own bandwidth use.

More information

- About JANET, *http://www.ja.net/about/index.html*
- About UKERNA, *http://www.ja.net/about/ukerna/ukerna.html*
- National Web Cache Service, *http://www.jisc.ac.uk/index.cfm?name=acn_caching*
- JANET Acceptable Use Policy (AUP), *http://www.ja.net/services/publications/service-documentation/supportmanual/policies.html*
- JANET Backbone Schematic, *http://www.ja.net/about/topology/janetbackboneschematic.pdf*
- JANET Security, *http://www.ja.net/services/publications/service-documentation/supportmanual/security.html*
- About JISC, *http://www.jisc.ac.uk/*
- JANET DNS Cache Service, *http://www.ja.net/services/network-services/resolver/index.html*

Blackburn College, UK

Blackburn College of Further Education, in north-west England, has over 10,000 full-time students. They have 1800 PCs, laptops, and printers. Their network connects to JANET for access to the Internet and to other universities.

Blackburn College incorporates disciplinary procedures to encourage users to use bandwidth wisely and in moderation. This is achieved by a combination of policy and monitoring. Technical measures to control bandwidth use are limited, but may increase in the future.

The Problem

Being a member of JANET, Blackburn College is required to comply with the Acceptable Use Policy (AUP) of JANET. This means that they have to be able to respond to complaints from JANET about abusive traffic, and track down the offender, or risk being cut off entirely from JANET.

The bandwidth use on Blackburn College's connection to JANET is approaching saturation. Due to careful monitoring they are aware of this fact before it will become a problem. Rather than upgrading their connection, they are now preparing to reduce their usage.

The Solution

As recommended by UKERNA, Blackburn College has an IT policy that covers their network and Internet connection. The college also monitors and keeps sufficient records of network usage. In the event of an investigation, they will be able to provide detailed reporting to authorities.

The IT policy includes all the terms of the JANET acceptable use policy. It makes it very clear that network traffic is monitored, that "action will be taken against users transgressing the usage codes," and that users should have no expectation of privacy.

This monitoring system gives Blackburn College the power to track down a user responsible for any problems on their network, and the IT policy gives them the ability to apply disciplinary sanctions on the user. Users are motivated to effectively manage their own bandwidth use, or else they are disconnected.

Blackburn also uses restrictive technological methods to reduce their bandwidth usage. For example, all web access from the student network and from most users on the staff network, must go through a proxy server rather than a direct connection. The proxy server denies access to specific web sites, and includes a proxy cache that can reduce bandwidth use. Blackburn filters inbound and outbound IP traffic on the border router using access control lists, on the basis of protocols and port numbers.

The College runs two caching servers, one for the staff network and one for the student network. "At the time of the case study the staff cache had a hit rate of 45% of which 25% was without any validation" (a process where the cache server makes a small request to the original web server to check that the cached copy is still valid, which takes some time and bandwidth). "The student cache had a hit rate of 40% of which 30% was without any validation." The caching servers therefore reduce their bandwidth usage by approximately 40%.

The bandwidth of the College's connection to JANET, at the time of the case study, was 4 Mbps. This was the only link on the network that was approaching saturation (maximum capacity), and risked congestion as a result. They have started a long-term collaboration with JANET's Bandwidth Management and Advisory Service (BMAS) to investigate options for reducing and managing traffic on this link.

Blackburn College will be investigating systems for monitoring the network to identify applications and protocols. They will be looking to improve their filtering, both at the gateway router (OSI layer 3) and on the proxy server (layer 7). They are continuing to investigate pre-caching of popular websites such as Microsoft.com during network off-peak times. Finally, they are investigating the effectiveness of bandwidth shaping and throttling using various products, including their existing Cisco router.

In summary, Blackburn College has a well developed and established IT policy. It is implemented using monitoring and packet filtering, which may be regarded as part of an organisational bandwidth management strategy. Their bandwidth management operations are largely reactive, manual, and policy-based, but this is expected to change as a result of their collaboration with JANET BMAS.

More information

- Blackburn College of Further Education

  http://www.blackburn.ac.uk

- Blackburn College Case Study

  http://www.ja.net/services/network-services/bmas/good-practice/part3.html

- JANET Bandwidth Management and Advisory Service

  http://www.ja.net/services/network-services/bmas/

These case studies show us that bandwidth management is necessary to get good performance from most networks. Policy and technical measures are both useful, and work best in combination, when technology is used to implement policy.

--Alan Jackson

Malawi

At the University of Malawi's Mahatma Campus in Blantyre, there are several academic and research institutions. The largest institution is the College of Medicine. It has a central campus as well as departments located in wards at a nearby hospital. In addition, there are several Malaria research facilities.

Most of these institutions are connected to the College of Medicine, using a VSAT link located there for Internet access. Previously, these institutions were interconnected by means of wireless links. The low capacity of the links, coupled with frequency disturbances (and disturbances from tress and other objects blocking the free line of sight), meant that Internet access was rather poor in most places. As a result, the 512Kb/128Kb down/up-link was rarely fully utilised. My task, together with local IT staff, was to replace the existing network, and address bandwidth management issues once the local network no longer acted as a bottleneck. We chose to replace the wireless links with fibre optic cabling.

When it came to bandwidth management, we chose to focus on three items. Firstly, we wanted to shape the Internet traffic, in part to ensure that each institution received their purchased bandwidth. Secondly, we wanted to authenticate users before they could access the Internet. At the College of Medicine, most users access the network via Windows machines connected to a Samba Domain Controller, with an LDAP database for authentication. At other sites, Active Directory was used as the domain controller. By authenticating users, we wanted to create separate bandwidth classes for staff, researchers, and students at the College of Medicine. Additionally, we hoped to prevent unauthorised users from having direct access to our bandwidth.

We decided to use tc and iptables to shape traffic, and RRDtool to graph the results. The authentication for users on Windows machines connected to the Samba Domain Controller was to be handled by a Perl script that checked which users were logged in, and opened and closed slots in the iptables firewall. For laptop users, authentication would be performed against the same LDAP database as is used by the domain controller.

How did it turn out? In setting up the new network, we chose a switched topology with VLANs for each subnet and routing done by a central routing switch. This introduced quite a few new problems. In the old network, each subnet had a gateway that filtered traffic and performed NAT. As it turned out, many machines were infected with viruses. When we removed the gateways, the viruses' rapid propagation led to congestion in the fibre network. The upstream link had an average of 100% usage, which went down to 60% when we started to filter out traffic.

In light of these problems, we decided to scrap the switched topology, opting instead for a routed network in which each subnet has its own Linux/OSPF router gateway. At each gateway, traffic is redirected to Dansguardian, before being passed to a central Squid web cache. We are currently doing some basic traffic shaping with tc and the HTB queue, while also trying to find settings that are appropriate for the VSAT link with its long delay. We are also planning additional security measures - that will automatically locate infected hosts and block them until they are clean.

To date, our case has taught us that bandwidth management requires an integrated approach. We needed to control the virus situation and prevent infected hosts from using up Internet bandwidth. We have also learned the importance of communication and policies. For example, we are performing content filtering, so someone must decide what material is appropriate. Also, there must be an efficient way for the user to alert the administrator when a useful site is blocked.

--David Blomberg, KTH

One Bellevue Center

Infospace, Inc. is a medium-sized company based in Bellevue, Washington, USA. Infospace provides the behind-the-scenes technology necessary for ring tones, mobile games, and WAP portals. To alleviate its growing pains, Infospace quickly acquired an additional lease property approximately one block from its main office. A 10 Mbps metro Ethernet service was obtained as a private Layer 2 link between the two buildings. A Netscreen 50 Firewall/VPN device was connected to the metro ethernet in each building.

While a 10 Mbps link may sound like a lot, the demand quickly outgrew the capacity engineered between the two offices. One hundred and fifty employees, each with multiple computers and VoIP telephone handsets, were scheduled to move into these offices. Security requirements included a five camera high-resolution security system, to be streamed to the physical security department located in the main building.

While VoIP conversations generally worked fine, the five security cameras consumed almost 9 Mbps of the 10 Mbps connection, and intermittently caused interruption. Internet, file sharing, and other network services came to a crawl. Network engineers were faced with a serious contention problem. They considered doubling or tripling the line rate to correct the problem. This would have increased the cost of the circuit by up to 200%. Also, it was discovered that the network provider used aggressive packet dropping (not a highly favored rate limiting method) once the data rate exceeded 10 Mbps.

However, an engineer suggested a low cost alternative: combine user education, Quality of Service, and traffic shaping while modifying several simple controls in the video security system.

First, traffic shaping was put on both Netscreens to set a maximum rate of 10 Mbps on both interfaces. This immediately solved the problem of packet drops and retries. Second, an additional traffic shaping policy provisioned 10 Mbps to VoIP services. Third, all other traffic was given a slightly lower priority, yet allowed to borrow from any of the 10 Mbps provisioned not in use. Next, network engineers discovered that the video security system actually had video bit rate controls that could be easily changed. The system was configured in such a way that no more than 1 Mbps would be used for video streaming. Security was also informed that opening more than one camera at once would degrade network performance, and so they did not use the full 1 Mbps unless it was necessary to do so.

Users were also informed that bandwidth was limited, and to be careful with the activities they performed during business hours. This helped to make network access more fair for everyone.

All of these measures were successful in managing bandwidth within the office environment without the need for adding more capacity.

--Casey Halverson

Carnegie Mellon University

In early September of 2001, the members of the Carnegie Mellon Network Group anxiously awaited the return of the students to the campus. We knew that upon their arrival we would see a dramatic increase in our network traffic across our border router. We knew that it would be bad, indeed we were in the process of installing a new gigabit Ethernet connection to replace our OC-3, and our expectations were fulfilled. Immediately our router started dropping ATM cells. This made the situation horrible. When we dropped one 53 byte cell out of the 30 or so needed to transport a 1500 byte packet, the sending host was then required to retransmit the entire packet. This positive feedback was unbearable.

Workaround #1: Best effort rate limiting

In order to survive until our gigabit connection was ready for production, we installed committed access rate limiters on our border router. We had to experiment to find the maximum rate we could allow without dropping cells. We found that our OC-3 could only carry about 75 Mbps before it would start to drop cells. We were still dropping packets, but now they were dropping before going across the ATM link. Application latency increased dramatically, but at least the network was now usable.

Getting more than you paid for

A few months later, we had our gigabit Ethernet link up and running. We did not have to worry about oversubscribing the physical line. Our next problem was more political in nature. Carnegie Mellon and a few other Universities all contributed to the Pittsburgh Supercomputing Center's network group to manage the "Gigapop." The PSC in turn managed a link to the Abilene high speed research network, as well as a number of commodity Internet links, via a number of tier one ISPs. Each school was then allocated a certain percentage of the overall commodity bandwidth. At the time, there were no hard limits on the maximum bandwidth. Instead, problems were dealt with personally with a phone call or email. Sure enough, it wasn't long before we heard from them, telling us that we were pushing a lot more traffic than we were paying for. It was now the spring of 2002, and we needed to come up with a new plan.

Workaround #2: Fun with rate limiting

When we installed our gigabit connection, we divided it into two Virtual LANs (VLANs) using 802.1q. One of the VLANs peered with the Internet2 / Abilene routers, and the other one peered with the commodity Internet routers. Since our bandwidth to Internet2 was not a problem, and we were required to support researchers with large bandwidth needs, we needed to constrain only traffic headed to the commodity routers. Our first thought was to use the same rate limiters we used on the ATM link. We were disappointed to find that our hardware did not support egress rate limiters. Quality of Service (QoS) would not help us, because there was no congestion on our routers.

We brainstormed a number of ideas, and our final solution was "interesting." If we could only limit traffic coming into an interface, we needed to create a point in our network where all commodity traffic would go into a single network port. Through a complex combination of VLANs, a fibre link between two ports on our border router, and redistributing Internet2 BGP routes into a new, separate OSPF process between our core and border routers, we managed to create an interface that we could rate limit.

It was not the perfect solution. We were not discriminating between different types of traffic, so every packet played Russian roulette as it attempted to pass the gauntlet of random drop. It is the kind of thing a network engineer would only do where it was an emergency and spending money was not an option. We knew that we were only buying some time to research a long term supportable solution.

More problems with packet drops

Summer came and went in 2002. During that time we dropped below our rate limiting threshold, but it was just the calm before the storm. When the students returned for the Fall session, our bandwidth usage immediately hit our thresholds and stayed there. People complained that they could not check email from home. Even SSH connections were nearly unusable. We decided to analyse the traffic usage from the previous day, and were surprised at the results. On September 4th, nine hosts were responsible for 21% of that days bandwidth utilisation. On that same day, 47% of the traffic was easily classifiable as peer-to-peer. 18% of the traffic was HTTP. Out of the remaining traffic, we believe that 28% of it consisted of port-hopping peer-to-peer traffic. Machines on our network were acting as servers to clients outside of our campus. That was what was causing us such pain. A new solution was needed.

Requirements and considerations

We needed a long term solution, not a workaround. Here is what we needed to consider:

- The solution must not impact mission critical services (e.g. www.cmu.edu, email, ssh).
- Traffic to Internet2 must not be affected.
- We could not rely on TCP ports to punish services, since the peer-to-peer clients would hop randomly between ports.
- We needed to constrain our commodity usage to match our allowed limits.

At the time, we already knew that some P2P applications were using port hopping to get around firewalls, access-lists, and rate limiting. Indeed, we foresaw that eventually the arms race between service providers and authors of P2P software would lead to encrypting the payload and piggybacking on port 80 or 443, which could obviously never be blocked. Instead of punishing "bad traffic," we decided it would be easier to white-list "good traffic" and then make a best-effort attempt at delivery for all other packets. At first we tried to apply our rules using QoS policy, but our routers were not able to enforce them. It was apparent that any new technical solution would require a capital expense.

Researching hardware rate limiters

Our next step was to evaluate a middle-box packet rate limiter. We did a head-to-head comparison of Allot's NetEnforcer and Packeteer's Packetshaper. We found the NetEnforcer to be a better match for our requirements. We liked the method the NetEnforcer used to shape traffic: it takes advantage of TCP window size and buffering, it fairly distributes bandwidth among flows, it was easier to configure, and most importantly, it had better throughput performance when demand for bandwidth equaled the limit.

Final solution or new workaround?

In October of 2002, we put the NetEnforcer in-line with traffic and used the classification data to help develop a policy. When we returned from holiday break in January, we put the policy in place. We used a per-flow, class based, fair bandwidth queueing policy. We had 5 classes of traffic, from high priority to low:

1. Network Critical (routing protocols)
2. Interactive (SSH and telnet) - limited per-flow, if bandwidth went over a certain limit, excess would be put into best effort queue
3. IMAP, HTTP, SMTP and other well-known ports
4. Non-classified traffic
5. P2P traffic, classified by port number

This policy resulted in fewer complaints to our help desk about accessing campus services remotely, and users' experiences seemed better. We were, however, still pushing traffic out of our network at the upper limit. We had some concerns regarding this solution:

- Per-host fair queueing was not possible. A user with 100 flows was getting 100 times the bandwidth of a user with one flow, so abuse was trivial and rampant.
- ssh -X forwarding performance was poor, making remote work difficult.
- There was high latency for UDP gaming services (which is a valid concern for a college student in the dorms).
- The demand for bandwidth was still high – socially, nothing had changed.

Application layer analysis to the rescue

In February, new software for our NetEnforcer arrived, and with it came layer 7 classification of a number of P2P services. At this time we put a hard limit on the amount of egress bandwidth these applications could consume. Our bandwidth graphs were finally showing some fluctuation below the hard limit, but the demand was still too close for comfort. We felt that we had reached the technical limit on solving our bandwidth problems. It was time for a new approach.

Social engineering

After many meetings, both in hallways and in conference rooms, we formalised our next steps. We found that we needed to change human behaviour and not network behaviour. To that end, we decided to initiate a dialogue with our campus community. Instead of the blanket email notices we had previously tried, we came up with a set of guidelines that included messages targeted to the owners of specific machines that were violating our policy. We had not tried this before because we did not have the technical systems in place to accurately measure per host bandwidth usage. We were also unsure of how the community would react to what some might see as an unfair or unneeded restriction of "their" connection to the Internet.

But in the end, our main reason for not trying to change user behaviour is that we did not know how easy it would be.

The campus bandwidth usage guidelines

We established a daily per-host bandwidth usage limit of 1 gigabyte of data per day for traffic leaving the campus out through the commodity Internet link. As before, we did not want to limit traffic going out across our research network link. Hosts on the wireless network were given a more stringent limit of 250 Megabytes per day. We previously had hosts transferring multiple gigabytes of data in a single day over their wireless link. We published these guidelines and requested comments from the campus community. We also included an exemption for specific hosts. The exception clause in our policy states:

> "*Owners of servers or services who exceed the usage guideline and feel they have legitimate cause to do so, must contact Computing Services. In rare cases an exemption may be granted. In all cases, solutions for fair use of the Commodity Link bandwidth will favor those services that support the research and educational goals of the university.*"

Our goal was to help the users follow the rules. To do this, we came up with a system for enforcing them that gave the users a chance to change their ways. We wanted the overall experience of the users to be positive. To accomplish this, our enforcement system worked this way:

- If a host exported more than 10 gigabytes of data over a 5 day period, we sent out an initial email message that informed the owner of the quota and how much they actually used. We requested that they provide information regarding their usage, if they felt it was legitimate, otherwise they should decrease their bandwidth consumption. If they needed help, we gave them our help desk's contact information.
- We then waited two days before taking any further action. On the third day we would start checking their daily usage and comparing it to the quota. If they ever exported 2 gigabytes or more in a single day, we would send a warning message. If the initial notification was the "good cop" message, this one played the role of "bad cop." The email warned the user that they must either provide a reason why the host should be exempted from the guidelines or decrease their bandwidth usage. If they did not comply, we would disconnect their host from the network. This would be treated like any other network abuse issue.

- We would then wait two more days to give them a chance to change their usage behaviour. After that, if they exceeded more than 2 gigabytes in a single day without contacting us to explain the situation, we would carry out our disconnection.

Human effort

Initially, the system was carried out manually. We had an automated system that would generate a daily report of all hosts and how much bandwidth they used, sorted by top usage. We already had a system in place that could tie every MAC address with a specific user or group. Then it was just a matter of going through the list by hand, updating a database of hosts in the various warning states, and sending the email messages. This took about two hours per day for one engineer. In the first week, we sent out 49 initial inquiry messages and 11 warnings. In March, we sent out 155 initial messages, 26 warning messages, and disconnected 5 hosts.

You may have have noticed that we were acting on numbers that were twice that of the policy. We thought it would be easier to find the sweet spot of per host limits if we started enforcing at a higher level and worked down to the official quota. Indeed, it was good that we did. On the first day of enforcement, we dropped our Internet usage from 120 Mbps to 90 Mbps. We were no longer dropping packets with our hard limits, and we were convinced that we had found a solution that worked. Our next step was to spend the resources to develop a software solution that would automate the entire system.

Positive results

We received very few negative responses related to the guidelines. Most users were completely unaware of their bandwidth usage. Many had installed peer-to-peer applications that ran in the background, unbeknownst to them. Some users discovered trojan FTP servers running on their hosts. Some of my favorite email response were "I don't understand--what's bandwidth?? How do I reduce it? What am I doing wrong? What's going on?!" and "I have no idea what bandwidth is or how one goes about using it too much."

By the summer of that year, we sent a total of 489 initial inquiry messages and 102 warnings. We revoked network access to 29 hosts and exempted 27. Over 39 machines were found to be compromised.

Conclusion

This system worked very well for us. There were two systems we had in place before we started working on the problem that proved to be invaluable. One of them was a system that allowed users to register their hosts on the network.

This self-service system provided the database that tied a MAC address to a specific user. It is called NetReg (available at *http://www.net.cmu.edu/netreg/*). Another system we had in place was an Argus monitoring infrastructure (*http://www.qosient.com/argus/*). This is a set of hosts that collects network flow data from routers. This was used to generate the "top talker" network usage reports. Both of these packages are freely available from their developers. Both of them require a decent system administrator to get up and running.

The big lesson we learned was that users were willing to change their behaviour, and that without their cooperation, technical solutions alone could not solve the problem. It was worth the effort to work with them to manage the bandwidth at our organisation. If you would like more information, here are some interesting links:

- Carnegie Mellon Network Bandwidth Usage Guideline: Wired network *http://www.cmu.edu/computing/documentation/policies_bandwidth/bandwidth.html*
- Carnegie Mellon Network Bandwidth Usage Guideline: Wireless network *http://www.cmu.edu/computing/documentation/policies_wirelessbw/wireless_bandwidth.html*
- How Computing Services Monitors and Enforces Network Bandwidth Usage *http://www.cmu.edu/computing/documentation/bandwidth_howenforced/index.html*
- Internet2 Joint Techs presentation resource page: "Successful Bandwidth Management at Carnegie Mellon," *http://www.net.cmu.edu/pres/jt0803/*

--Peter John Hill

8
The Future

"Predicting the future is always difficult and is no less so when the Internet is involved," wrote Dr. Jon Knight from Loughborough University in an article published in 2003. "Protocols and technologies appear and achieve widespread use at an alarming rate and much of what is now commonplace was unheard of or only in research laboratories five years ago." Although there are these difficulties in prediction, the development and expansion of current technologies can be analysed for their future bandwidth management implications.

Bandwidth consuming technologies

Among these rapidly-growing Internet technologies is peer-to-peer (P2P) file-sharing, which allows users from around the world to connect to each other and trade items such as music and video files. This was initially made popular in 1999 by the online service Napster, which subsequently lost a high-profile legal battle over copyright violations. However, P2P technologies now have increasing support from large, well-known companies. For example, Warner Bros. announced in 2006 that they would sell some of their films and TV shows using the popular BitTorrent file-sharing system.

This may have an interesting effect on bandwidth management and usage at institutions around the world. For example, many schools and universities have banned or limited the use of P2P programs on their networks, partly because they consume large amounts of bandwidth, and also because they are often used to trade copyrighted files illegally. With the rise in the number of legal, paid-for services that use P2P technologies, the justification for banning file-sharing entirely is no longer so clear.

The streaming of audio and video is becoming increasingly popular, and uses large amounts of bandwidth in comparison to traditional text-only media. Voice over IP (VoIP) allows telephone calls to be made over the Internet at relatively low cost, and could foreseeably replace current telecommunication systems. Companies such as Skype have established well-known services in this area. Downloading live television broadcasts from Web sites (such as the BBC's) is also becoming common. In addition to causing a heavy load on the network, streaming technologies need a stable, reliable connection. As institutional VoIP usage is likely to increase to the point of being an essential service, this will place a great demand for effective bandwidth management.

A definite trend is continuing towards multimedia websites, which contain bandwidth-hungry images, video, animations, and interactive content. This puts increasing demands on Internet connections, resulting in slower downloads and decreased usability on saturated connections. Bandwidth management technologies will have to adapt to this change, employing new caching technologies to reduce bandwidth demand and improve user experience.

Although policy may ban or discourage a certain type of Internet use, in practice it can be difficult to completely enforce that policy purely by technical means. For example, an institution may have a policy of limiting the use of file sharing. However, determined users can tunnel file sharing over other protocols like HTTP, which is difficult or impossible for the institution to detect.

Trends in developing countries

As the number of Internet users in underdeveloped parts of Africa and Asia expands, there will also be a growing need to provide more local services. It will become more important for copies of large files (e.g. open-source software) to be stored on servers closer to users, which will enhance the speed and reliability of downloads. This technique, called "mirroring," is already widely used in the developed world, but there are no known public mirrors of popular software on the African continent.

Even today, 59% of African universities have limited or no bandwidth management, according to a report by ATICS (p.47). As small Local Area Networks (LANs) are created and extended, it is likely that more and more people will become de facto network administrators, despite having little or no training. If this training shortage problem is not addressed, the situation can only become worse.

Organisations with tighter budgets, such as universities, colleges, and research institutes, will continue to suffer with respect to the private sector in terms of the speed and reliability of their Internet connections. Unfortunately, these are the same institutions which cannot afford the best systems administrators, and where the public benefit that would arise from unfettered access to information is greatest.

As the knowledge of cheaper phone calls over the Internet spreads in developing countries, coupled with gradually increasing bandwidth and gradually spreading networks, users will start to demand faster and better Internet connections. This will erode the revenue of national monopoly telecom providers, who will seek greater legal protection of their monopolies. Hard battles will continue to be fought over industry deregulation in developing countries.

Newer bandwidth management software and network hardware, with better support for guaranteed network Quality of Service (QoS), will spread beyond the best equipped networks. Thus administrators will find themselves under pressure to implement these systems on their networks.

Conversely, shared bandwidth connections such as ADSL will continue to grow in popularity, at the expense of guaranteed bandwidth connections such as leased lines. While these seem cheaper, and often offer better performance on average, shared connections make it very difficult to manage bandwidth since the usable bandwidth is unknown and constantly varying. While Service Level Agreements (SLAs) can bind an ISP to provide a specified minimum level of performance, these agreements can come at a significant cost, particularly in areas that have little competition between providers.

New software

Aidworld and KENET are currently developing a simple open source toolkit which will provide affordable, reliable bandwidth management. While commercial solutions already exist, they are expensive and targeted at large organisations. The toolkit is designed for use in any institution regardless of size, bandwidth or staff experience.

BC Router, a project in development at Leuven University, is designed to provide fair bandwidth for all users on a network. It does this by providing an equitable share of bandwidth for each user at the packet level - and therefore across all protocols and all uses. With a per-user allocated bandwidth allowance, people who abuse the network will be the ones most affected since any increase in traffic will just consume their bandwidth share. This project is already live within the university, but at the time of writing is not yet ready for general release.

Other mini-distributions are beginning to evolve that integrate bandwidth management tools with a simple configuration interface. The development of tools such as IPCop (*http://www.ipcop.org/*), m0n0wall (*http://m0n0.ch/wall/*), and SmoothWall (*http://www.smoothwall.org/*) are bringing sophisticated traffic shaping and firewall tools into the hands of less experienced network administrators, and can be run on conventional PC hardware. In time, this should help to make good bandwidth management techniques part of every standard Internet connection.

In closing

The ultimate vision of the Internet is a truly ubiquitous network where information flows freely to wherever humans can make use of it. It is likely that we will eventually build a network where there is sufficient bandwidth for all. But until this is achieved, we must continue to carefully manage network resources, ensuring that this limited resource is equitably accessible to everyone.

The technologies that allow us to effectively manage Internet resources are evolving as quickly as the Internet. New techniques and tools are being developed every day that help us to squeeze even more performance out of our overburdened network connections. Be sure to check in at our web site (*http://bwmo.net/*) for more resources and future updates to this work.

Appendix A
Resources

Links

This list includes most of the URLs referenced in this book. For more online resources and further reading, see our website at *http://bwmo.net/*

Anti-virus & anti-spyware tools

- AdAware, *http://www.lavasoftusa.com/software/adaware/*
- Clam Antivirus, *http://www.clamav.net/*
- Spychecker, *http://www.spychecker.com/*
- xp-antispy, *http://www.xp-antispy.de/*

Benchmarking tools

- Bing, *http://www.freenix.fr/freenix/logiciels/bing.html*
- DSL Reports Speed Test, *http://www.dslreports.com/stest*
- The Global Broadband Speed Test, *http://speedtest.net/*
- iperf, *http://dast.nlanr.net/Projects/Iperf/*
- ttcp, *http://ftp.arl.mil/ftp/pub/ttcp/*

Content filters

- AdZapper, *http://adzapper.sourceforge.net/*
- DansGuard, *http://dansguardian.org/*
- Squidguard, *http://www.squidguard.org/*

DNS & email

- Amavisd-new, *http://www.ijs.si/software/amavisd/*
- BaSoMail, *http://www.baso.no/*
- BIND, *http://www.isc.org/sw/bind/*
- dnsmasq, *http://thekelleys.org.uk/dnsmasq/*
- DJBDNS, *http://cr.yp.to/djbdns.html*
- Exim, *http://www.exim.org/*
- Free backup software, *http://free-backup.info/*
- Life with qmail, *http://www.lifewithqmail.org/*
- Macallan Mail Server, *http://macallan.club.fr/*
- MailEnable, *http://www.mailenable.com/*
- Pegasus Mail, *http://www.pmail.com/*
- Postfix, *http://www.postfix.org/*
- qmail, *http://www.qmail.org/*
- Sendmail, *http://www.sendmail.org/*

File exchange tools

- DropLoad, *http://www.dropload.com/*
- FLUFF, *http://www.bristol.ac.uk/fluff/*

Firewalls

- IPCop, *http://www.ipcop.org/*
- L7-filter, *http://l7-filter.sourceforge.net/*
- Linux iptables HOWTO, *http://www.linuxguruz.com/iptables/howto/*
- m0n0wall, *http://m0n0.ch/wall/*
- Netfilter, *http://www.netfilter.org/*

- Network Address Translation HOWTO : *http://www.netfilter.org/documentation/HOWTO/NAT-HOWTO.html*
- Packet Filtering HOWTO: *http://www.netfilter.org/documentation/HOWTO/packet-filtering-HOWTO-7.html*
- PF: The OpenBSD Packet Filter, *http://www.openbsd.org/faq/pf/*
- Smoothwall, *http://www.smoothwall.org/*
- Shorewall, *http://shorewall.net/*
- ZoneAlarm, *http://www.zonelabs.com/*

Flow monitors

- EtherApe, *http://etherape.sourceforge.net/*
- Flowc, *http://netacad.kiev.ua/flowc/*
- iptraf, *http://iptraf.seul.org/*
- MRTG, *http://oss.oetiker.ch/mrtg/*
- NeTraMet, *http://www.auckland.ac.nz/net/NeTraMet/*
- RRDtool, *http://oss.oetiker.ch/rrdtool/*

Internet authorities

- APNIC, http://www.apnic.net/
- AfriNIC, *http://www.afrinic.net/*
- ARIN, http://www.arin.net/
- IANA, *http://www.iana.org/*
- LACNIC, *http://lacnic.net/*
- RIPE, *http://www.ripe.net/*

Log parsers

- adcfw-log, *http://adcfw-log.sourceforge.net/*
- ADMLogger, *http://aaron.marasco.com/linux.html*
- Analog, *http://www.analog.cx/*
- AWStats, *http://awstats.sourceforge.net/*
- Calamaris, *http://cord.de/tools/squid/calamaris/*
- IPTables log analyzer, *http://www.gege.org/iptables/*
- isoqlog, *http://www.enderunix.org/isoqlog/*

- Logwatch, *http://www.logwatch.org/*
- Sawmill, *http://www.sawmill.net/*
- Webalizer, *http://www.mrunix.net/webalizer/*

Mirroring tools

- Curl, *http://curl.haxx.se/*
- HTTrack, *http://www.httrack.com/*
- rsync, *http://rsync.samba.org/*
- wget, *http://www.gnu.org/software/wget/*

Policy

- Carnegie Mellon Network Bandwidth Usage Guideline: Wired network *http://www.cmu.edu/computing/documentation/policies_bandwidth/bandwidth.html*
- Carnegie Mellon Network Bandwidth Usage Guideline: Wireless network *http://www.cmu.edu/computing/documentation/policies_wirelessbw/wireless_bandwidth.html*
- Educause collation on Acceptable/Responsible Use Policies, *http://www.educause.edu/content.asp?page_id=645&PARENT_ID=110&bhcp=1*
- Examples of Internet Acceptable Use Policies, *http://ndsl.lib.state.nd.us/AcceptableUseExp.html*
- Illegal software and film downloads exhaust university computer networks, *http://www.hs.fi/english/article/1101978960379*
- INASP Policy Development Workshop, *http://www.inasp.info/training/bandwidth/bmo-pdw/*
- JANET Acceptable Use Policy (AUP), *http://www.ja.net/services/publications/service-documentation/supportmanual/policies.html*
- Policy and rules on Internet and Email use, from The University of Cape Town, *http://www.icts.uct.ac.za/modules.php?name=News&file=print&sid=633*
- The SANS institute policy template page: *http://www.sans.org/resources/policies/#template*
- Tech Republic: A framework for e-mail and Internet usage policies for your enterprise, *http://articles.techrepublic.com.com/5102-6299-1033914.html*
- University of KwaZulu-Natal's ELECTRONIC COMMUNICATIONS POLICY, *http://www.nu.ac.za/itd/policies/ecommunications.pdf*

Protocol analysers

- tcpdump, *http://www.tcpdump.org/*
- SoftPerfect Network Scanner, *http://www.softperfect.com/*
- WinDump, *http://www.winpcap.org/windump/*
- Wireshark, *http://www.wireshark.org/*

Protocol tuning

- Lawrence Berkeley National Laboratory's TCP Tuning Guide, *http://dsd.lbl.gov/TCP-tuning/background.html*
- Pittsburgh Supercomputing Center's guide to Enabling High Performance Data Transfers, *http://www.psc.edu/networking/perf_tune.html*
- Swedish University Computer Network TCP tuning parameters guide, *http://proj.sunet.se/E2E/tcptune.html*
- TCP Tuning and Network Troubleshooting by Brian Tierney, *http://www.onlamp.com/pub/a/onlamp/2005/11/17/tcp_tuning.html*
- TXQueueLen Investigation into IP Performance, *http://www.hep.ucl.ac.uk/~ytl/tcpip/linux/txqueuelen/*

Proxies & caches

- Automatic Proxy Configuration protocol, *http://wp.netscape.com/eng/mozilla/2.0/relnotes/demo/proxy-live.html*
- JANET DNS Cache Service, *http://www.ja.net/services/network-services/resolver/index.html*
- Microsoft ISA Server, *http://www.isaserver.org/*
- National Web Cache Service, *http://www.jisc.ac.uk/index.cfm?name=acn_caching*
- PEPsal, *http://sourceforge.net/projects/pepsal/*
- Squid, *http://squid-cache.org/*
- Squid cache wiki, *http://wiki.squid-cache.org/*
- Squid documentation, *http://www.visolve.com/squid/*
- Squid for Windows, *http://www.acmeconsulting.it/SquidNT/*
- Squid User's Guide, *http://www.deckle.co.za/squid-users-guide/*

Realtime monitoring tools

- Nagios, *http://nagios.org/*
- Zabbix, *http://www.zabbix.org/*

Security

- GotRoot, *http://gotroot.com/*
- JANET Security*,* *http://www.ja.net/services/publications/service-documentation/supportmanual/security.html*
- Linux security and admin software, *http://www.linux.org/apps/all/Networking/Security_/_Admin.html*
- ModSecurity, *http://www.modsecurity.org/*
- ngrep, *http://ngrep.sourceforge.net/*
- nmap, *http://insecure.org/nmap/*
- Snort, *http://snort.org/*

Spam fighting tools

- denysoft_greylist, *http://www.openfusion.com.au/labs/dist/denysoft_greylist*
- DomainKeys, http://antispam.yahoo.com/
- DSPAM, *http://dspam.nuclearelephant.com/*
- exim-greylist, *http://johannes.sipsolutions.net/Projects/ex*
- Gld Greylists on Postfix, *http://www.gasmi.net/gld.html*
- gps - greylist policy service for Postfix, *http://mimo.gn.apc.org/gps/*
- Greylisting with Exim & MySQL, *http://theinternetco.net/projects/exim/greylist*
- Greylisting with Exim & Postgres, *http://raw.no/personal/blog/tech/Debian/*
- Mail relay testing tool, *http://www.abuse.net/relay.html*
- milter-greylist, *http://hcpnet.free.fr/milter-greylist/* http://home.teleport.com/~nb6z/
- Open Relay Database, *http://www.ordb.org/*
- policyd for Postfix, *http://policyd.sourceforge.net/*
- postgrey, *http://isg.ee.ethz.ch/tools/postgrey/*
- qgreylist, *http://www.jonatkins.com/page/software/*

- Smart Sendmail filters, *http://smfs.sourceforge.net/*
- SpamAssassin, *http://spamassassin.apache.org/*
- Spam Filtering for Mail Exchangers, *http://www.tldp.org/HOWTO/Spam-Filtering-for-MX/*
- SPF: Sender Policy Framework, *http://www.openspf.org/*
- SPF mail filter, *http://www.acme.com/software/spfmilter/*
- SPF support in Postfix, *http://www.linuxrulz.org/nkukard/postfix/*
- SQLgrey, *http://sqlgrey.sourceforge.net/*
- tumgreyspf, *http://www.tummy.com/Community/software/tumgreyspf/*

Spot check tools

- MyTraceRoute, *http://www.bitwizard.nl/mtr/*
- ntop, *http://www.ntop.org/*

Traffic shaping tools

- BWM Tools, *http://bwm-tools.pr.linuxrulz.org/*
- WonderShaper, *http://lartc.org/wondershaper/*
- The Linux Advanced Routing and Traffic Control HOWTO, *http://lartc.org/*

Trending tools

- Argus, *http://www.qosient.com/argus/*
- Cacti, *http://www.cacti.net/*
- SmokePing, *http://oss.oetiker.ch/smokeping/*

Very low bandwidth

- Das Packet Radio-Portal, *http://www.packetzone.de/*
- Introduction to Packet Radio, *http://www.choisser.com/packet/*
- loband, *http://www.loband.org/*
- TEK, *http://tek.sourceforge.net/*
- www4mail, *http://www.www4mail.org/*

More information

- AidWorld, *http://www.aidworld.org/*

- Enhancing International World Wide Web Access in Mozambique Through the Use of Mirroring and Caching Proxies, *http://www.isoc.org/inet97/ans97/cloet.htm*
- FreeBSD Handbook, *http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/*
- Freely available ISO standards, *http://standards.iso.org/ittf/PubliclyAvailableStandards/*
- Guide to IP Layer Network Administration with Linux, *http://linux-ip.net/html/*
- How Computing Services Monitors and Enforces Network Bandwidth Usage *http://www.cmu.edu/computing/documentation/bandwidth_howenforced/index.html*
- ICTP, *http://www.ictp.it/*
- ICTP Workshop on Optimization Technologies for Low-Bandwidth Networks, *http://cdsagenda5.ictp.it/full_display.php?smr=0&ida=a05228*
- INASP, *http://www.inasp.info/*
- INASP Network Traffic Monitoring and Analysis Workshop, *http://www.inasp.info/training/bandwidth/bmo-ntmw/*
- Internet2 Joint Techs presentation resource page: "Successful Bandwidth Management at Carnegie Mellon," *http://www.net.cmu.edu/pres/jt0803/*
- JANET Bandwidth Management Review, *http://www.ja.net/services/network-services/bmas/papers/review/BMAS_Bandwidth_Management_Review.htm*
- Linux Advanced Routing and Traffic Control HOWTO, *http://lartc.org/*
- Linux networking wiki, *http://linux-net.osdl.org/*
- MTA comparison, *http://shearer.org/MTA_Comparison*
- Optimising Internet Bandwidth in Developing Country Higher Education, *http://www.inasp.info/pubs/bandwidth/*
- Planet Malaysia blog on bandwidth management, *http://planetmy.com/blog/?p=148*
- The VSAT Buyer's Guide, IDRC, 2005 *http://ictinafrica.com/vsat/*
- Wessels, Duane. *Squid: The Definitive Guide*. O'Reilly Media (2004). *http://squidbook.org/*
- Wireless Networking in the Developing World, *http://wndw.net/*

Wikipedia entries

Wikipedia has a wealth of information about Internet protocols. As with any wiki, information should always be verified by checking with other sources. These entries are an excellent starting place for learning more about how the Internet works.

- *http://en.wikipedia.org/wiki/Bandwidth_management*
- *http://en.wikipedia.org/wiki/Colocation_centre*
- *http://en.wikipedia.org/wiki/Ethernet*
- *http://en.wikipedia.org/wiki/Internet_protocol_suite*
- *http://en.wikipedia.org/wiki/Network_traffic_measurement*
- *http://en.wikipedia.org/wiki/OSI_model*
- *http://en.wikipedia.org/wiki/Synchronous_optical_networking*
- *http://en.wikipedia.org/wiki/TCPIP*
- *http://en.wikipedia.org/wiki/Wide_area_network*

Relevant RFCs

While some RFCs are approved by the IETF and become official Internet standards, others are simply proposals or technical background on network engineering challenges. Many of these become de facto standards even without official approval.

The RFCs listed here are mentioned in this book, and are a good starting point for learning more about the various Internet protocols. You can view RFCs online at *http://rfc.net/*.

- RFC1144: *Compressing TCP/IP Headers for Low-Speed Serial Links*
- RFC1323: *TCP Extensions for High Performance*
- RFC1518: *An Architecture for IP Address Allocation with CIDR*
- RFC1918: *Address Allocation for Private Internets*
- RFC1928: *SOCKS Protocol Version 5*
- RFC1977: *PPP BSD Compression Protocol*
- RFC1979: *PPP Deflate Protocol*
- RFC2186: *Internet Cache Protocol (ICP), version 2*

- RFC2821: *Simple Mail Transfer Protocol*
- RFC3135: *Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations*

Appendix B
Squid ACL Primer

This is a crash course on Squid ***Access Control Lists*** (***ACLs***) in Squid. For more background on how to use Squid in your organisation, see chapter six, **Performance Tuning**.

There are two types of ACL objects in Squid: ***ACL elements*** and ***ACL rules***. Both are used together to implement access control on your Squid cache. ACLs are specified in your **`squid.conf`** file. When you are finished making changes to your ACLs, you should run **`squid -k reconfigure`**. This causes Squid to reload the config file and use your new settings.

ACL elements

ACL elements are the basic building blocks of the access control features of Squid. They let you define elements that identify a request, such as IP addresses, port numbers, host names or user names.

The basic syntax of a ACL element is:

```
acl name type value1 value2 value3 ...
```

It is important to note that elements use OR logic to process the values. This means that as soon a match is found, the element will return true. It is possible to refer to the same element name on multiple lines, as well as on the same line. For example:

```
acl myinternalip 10.1.0.0/255.255.0.0 10.2.0.0/255.255.0.0
```

...is the same as:

```
acl myinternalip 10.1.0.0/255.255.0.0
acl myinternalip 10.2.0.0/255.255.0.0
```

ACL elements can also be contained in external files. This makes it easy to manage a long list of values.

```
acl myinternalip "/etc/mynets"
```

This directs Squid to read **`/etc/mynets`** to populate the **`myinternalip`** element. Each value should be on its own line in the **`/etc/mynets`** file.

Some other useful ACL elements are:

- **`src`**. This will match the IP address of the client making the request.

```
acl aclname src [ip-address/netmask]
```

- **`dst`**. This refers to the web server's IP address. When using this element in an ACL, Squid performs a DNS lookup on the host specified in the HTTP request header. This can add a delay to ACL processing as Squid waits for the DNS response.

```
acl aclname dst [ip-address/netmask]
```

- **`dstdomain`**. This matches the domain of the site requested by the client. A leading **`.`** before the domain will cause all subdomains to be matched as well.

```
acl aclname dstdomain [domain-name]
```

- **`dstdom_regex`**. This is similar to **`dstdomain`**, but uses a regular expression to match the domain.

```
acl aclname dstdom_regex [pattern]
```

- **`time`**. This matches the time and day of the week.

```
acl aclname time [day] [hh:mm-hh:mm]
```

The day is a one-letter abbreviation as follows: **`S`** (Sunday), **`M`** (Monday), **`T`** (Tuesday), **`W`** (Wednesday), **`H`** (Thursday), **`F`** (Friday), or **`A`** (Saturday). The last parameter specifies a range of time, and is specified in 24-hour format. For example, this would match Monday through Friday, 8:00am to 6:00pm:

```
acl business_hours time MTWHF 8:00-18:00
```

- **`url_regex`**. This searches the entire URL for the regular expression specified. Note that these regular expressions are case-sensitive by default. To make them case-insensitive, use the **`-i`** option just before the pattern.

```
acl aclname url_regex [pattern]
```

- **`urlpath_regex`**. This performs regular expression pattern matching on the URL, but without the protocol and host name. Note that as with **`url_regex`**, these regular expressions are case-sensitive by default unless you use the **`-i`** switch.

```
acl aclname urlpath_regex [pattern]
```

- **`port`**. This matches the destination port address used by the web server.

```
acl aclname port [number]
```

- **`proxy_auth`**. Matches an authenticated proxy user. Note that **`proxy_auth`** requires an external authentication program (which is specified by the **`authenticate_program`** tag). Multiple user names are separated by spaces.

```
acl aclname proxy_auth [user names]
```

  You can use the magic term **`REQUIRED`** instead of an explicit user name to accept any valid user name.

- **`proxy_auth_regex`**. Similar to the proxy_auth element, but uses a regular expression to match the user name. Remember to use the -i switch if you wish to ignore case when matching.

```
acl aclname proxy_auth_regex [pattern]
```

ACL rules

ACL rules combine ACL elements to control access to certain features of Squid. It is important to remember is that the ACL rules use AND logic. This means that all elements on line need to evaluate to true in order for the rule to execute. The rules are processed from the top down. As soon as a rule matches, the rule is executed and all subsequent rules using the same rule type will be ignored.

It is a very good idea to keep ACL rules of the same type together. It can be very easy to get confused if you have, for example, your **`http_access`** rules scattered all over your **`squid.conf`** file.

There are several ACL rule types. Here are some of the most commonly used rules.

- **`http_access`**. Allow or deny http access based on a previously defined element.

```
http_access [allow|deny] [aclname]
```

You can precede any element with a **`!`** to match anything that is **not** on the list. If none of the access lines match, then the default is to do whatever is the opposite of the last line in the list. It is a good idea to insert a **`deny all`** or **`allow all`** entry at the end of your access lists to avoid confusion.

- **`icp_access`**. If your Squid cache is configured to serve ICP replies (when using hierarchical caches), you should use the **`icp_access`** list. In most cases, you should only allow ICP requests from your neighbor caches.

```
icp_access [allow|deny] [aclname]
```

- **`redirector_access`**. This access list determines which requests are sent to one of the redirector processes. If you do not specify a **`redirector_access`** line, all requests go through the redirectors. You can use this list to prevent certain requests from being rewritten.

```
redirector_access [allow|deny] [aclname]
```

- **`delay_access`**. This access list rule determines if a client should be sent to a delay pool. Note that the client is directed to the first delay pool that matches.

```
delay_access [delaypoolnumber] [allow|deny] [aclname]
```

Examples

Here are some examples of how to perform common ACL tasks in Squid.

Allow only local clients

Almost all Squid installations need to restrict access based on the client's IP address. This is one of the best ways to protect your cache from abuse and bandwidth theft. This example will only permit clients from **`MyNet`** to access the cache.

```
acl MyNet src 10.2.1.0/24 10.2.2.0/24
http_access allow MyNet
http_access deny All
```

Deny a list of sites

Add the list of sites you wish to block to the file specified by **BadSites**, with one site per line.

```
acl BadSites dstdomain "/usr/local/squid/etc/badsites"
http_access deny BadSites
http_access allow MyNet
http_access deny All
```

Block a few clients by IP address

```
acl BadPc src 10.1.2.4
http_access deny BadPc
http_access allow MyNet
http_access deny All
```

Allow access to the bad sites only after hours

```
acl workhours acl workhours time MTWHF 08:00-17:00
acl BadSites dstdomain "/usr/local/squid/etc/badsites"
http_access deny workhours BadSites
http_access allow MyNet
http_access deny All
```

Block certain users regardless of their IP address

```
acl Authenticated proxy_auth REQUIRED
acl BadUsers proxy_auth Richard John
http_access deny BadUsers
http_access allow MyNet
http_access deny All
```

Direct certain users to a delay pool

```
acl Authenticated proxy_auth REQUIRED
acl SlowUsers proxy_auth Richard John
http_access allow MyNet
http_access deny All

delay_access 2 allow SlowUsers
delay_access 2 deny # This so no other users match this pool
delay_access 1 allow all
```

For more information about ACLs, see the online Squid documentation at *http://www.deckle.co.za/squid-users-guide/.*

Glossary

0 - 9

802.11. While 802.11 is a wireless protocol in its own right, 802.11 is often used to refer to a family of wireless networking protocols used mainly for local area networking. Three popular variants include 802.11b, 802.11g, and 802.11a. See also ***Wi-Fi.***

95th percentile. A billing method used by calculating the highest traffic rate for a given period, discarding the highest 5%. Compare with ***flat rate*** and ***actual usage***.

A

ACCEPT. The netfilter target that permits packets to pass. Packet matching stops immediately once the ACCEPT target is met. Compare with ***DROP***, ***LOG***, and ***REJECT***.

Acceptable Use Policy see ***AUP.***

ACL (***Access Control List***). A set of rules that must be matched before access is granted to a system. IP addresses, user names, and port numbers are frequently used in ACLs.

ACL elements. In Squid, ACL elements define a list of attributes (such as source IP, MAC address, user name, or browser type) to be later matched by rules. Together, elements and rules define the resources that are permitted by the Squid cache.

ACL rules. In Squid, ACL rules take some action (usually to permit or deny access) by comparing the request with various ACL elements.

actual usage. A billing method used by calculating the total number of bytes transferred in a given time period (usually one month). Compare with ***flat rate*** and ***95th percentile***.

Address Resolution Protocol see ***ARP.***

address space. A group of IP addresses that all reside within the same logical subnet.

ADSL (***Asymmetric Digital Subscriber Line***) see ***DSL***.

advertised window. The portion of a TCP header that specifies how many additional bytes of data the receiver is prepared to accept.

adzapper. A Squid redirector that intercepts advertisements and replaces them with smaller static graphic files. Available from *http://adzapper.sourceforge.net/.*

ALTQ (***Alternate Queuing***). ALTQ is a packet scheduler used to shape network traffic on BSD systems.

Analog (*http://www.analog.cx/*). A popular web and cache server log reporting tool.

AND logic. A logical operation that only evaluates as true if all of the items being compared also evaluate as true. See also ***OR logic***.

Application firewalls. A special kind of network firewall that can approve or deny traffic based on a high level analysis of the application protocol being used. For example, a web application firewall can inspect the contents of HTTP packets to determine whether a particular connection should be permitted.

Application Layer. The topmost layer in the OSI and TCP/IP network models, where applications can exchange data without regard for underlying network layers.

Argus. An open source network monitoring tool used for tracking flows between hosts. Argus is short for ***Audit Record Generation and Utilization System***. Available from *http://www.qosient.com/argus* .

ARP (***Address Resolution Protocol***). ARP is a protocol widely used on Ethernet networks to translate IP addresses into MAC addresses.

ARQ (***Automatic Repeat Request***). ARQ provides radio link layer error recovery on HF networks by managing retransmission requests.

Asymmetric Digital Subscriber Line (***ADSL***) see ***DSL***.

AT command set. A common set of commands used by modems to select operating parameters and control a data connection. Originally developed by the Hayes corporation in the early 1980s.

Audit Record Generation and Utilization System see ***Argus.***

AUP (***Acceptable Use Policy***). A formal set of rules that defines how a network connection may be used. ISPs often provide service contingent upon adherence to an AUP.

authenticating cache servers. A caching web proxy that requires credentials (such as a user name and password) is an authenticating cache server. Authentication enables the ability to implement quotas, billing, and other services based on the usage patterns of individuals.

automatic proxy configuration. A technique used to automatically configure web browsers to detect and make use of a web proxy.

Automatic Repeat Request see ***ARQ.***

AWStats. A popular web and cache server log reporting tool available from *http://awstats.sourceforge.net/*

B

bands. A queue used for prioritised delivery of network traffic. In the QoS implementation in Linux, a packet's TOS bits determine the band that is used for delivery. See also ***PRIO***, ***QoS***, and ***TOS***.

Bandwidth. A measure of frequency ranges, typically used for digital communications. The word bandwidth is also commonly used interchangeably with ***capacity*** to refer to the theoretical maximum data rate of a digital communications line.

benchmarking. Testing the maximum performance of a service or device. Benchmarking a network connection typically involves flooding the link with traffic and measuring the actual observed throughput, both on transmit and receive.

Berkeley Internet Name Domain see ***BIND***.

BGAN (***Broadband Global Access Network***). One of several standards used for satellite Internet access. See also ***DVB-S*** and ***VSAT***.

BIND (***Berkeley Internet Name Domain***). BIND is probably the most common implementation of DNS used on the Internet. It is published by the Internet Systems Consortium, and is available from *http://www.isc.org/sw/bind/.*

bits per second see ***bps***.

Bonding. A method used for combining the throughput of two or more network connections.

bps (***bits per second***). A measure of capacity of a digital communications line. Often, bps is used in conjunction with the common prefixes K (kilo), M (mega), or G (giga). For example, a T1 line may be said to provide a 1.544 Mbps connection.

branch node. When using HTB, a branch node refers to a class that contains other child nodes. See also ***leaf node***.

bridge. A network device that connects two networks together at the Data Link layer. Bridges do not route packets at the Network Layer. They simply repeat packets between two link-local networks. See also ***router*** and ***transparent bridging firewall***.

Broadband Global Access Network see ***BGAN***.

broadcast address. On IP networks, the broadcast IP address is used to send data to all hosts in the local subnet. On Ethernet networks, the broadcast MAC address is used to send data to all machines in the same collision domain.

BSD Compression (***bsdcomp***). A common data compression algorithm used for compressing PPP headers. See also ***deflate*** and ***VJ***.

burst. To temporarily use a data rate above the agreed rate. In VSAT systems using shared bandwidth, bursting allows for a temporary increase in the maximum available throughput by "borrowing" from customers whose lines are idle.

by-the-bit see ***actual usage***.

C

Cable modem. A device that implements DOCSIS, allowing two-way network communications over consumer cable television lines.

Cache. A local copy of data that takes a long time to compute or retrieve. Network operations can be sped up by keeping a local cache of network data, such as web pages or DNS requests, and serving the local copy on subsequent requests.

cache digests. A very compact summary of the objects available in a cache. By using cache digests, inter-cache communications can be significantly reduced, increasing performance.

cachemgr. A command-line diagnostic interface that displays the status of a running Squid cache.

caching web proxy. A server that makes web requests on behalf of clients, and saves a local copy of the retrieved data to increase performance and decrease Internet utilisation.

Cacti (*http://www.cacti.net/*). A popular web-based monitoring tool written in PHP.

Calamaris. A very powerful web cache log analyser. Available from *http://cord.de/tools/squid/calamaris*

capacity. The theoretical maximum amount of traffic provided by a digital communications line. Often used interchangeably with ***bandwidth***.

captive portal. A mechanism used to transparently redirect web browsers to a new location. Captive portals are often used for authentication or for interrupting a user's online session (for example, to display an AUP).

CBQ (***Class Based Queueing***). A very popular and complex queuing discipline used for traffic shaping.

chains. One of the many phases of packet evaluation used by the Linux netfilter firewall system. Chains contain rules that determine the fate of every packet passing through the system. See also ***netfilter***, ***rules***, and ***tables***.

CIDR (***Classless Inter-Domain Routing***). CIDR was developed to improve routing efficiency on the Internet backbone by enabling route aggregation and network masks of arbitrary size. CIDR replaces the old class-based addressing scheme. See also ***Class A, B, and C networks***.

CIDR notation. A method used to define a network mask by specifying the number of bits present. For example, the netmask 255.255.255.0 can be specified as /24 in CIDR notation.

Class A, B, and C networks. For some time, IP address space was allocated in blocks of three different sizes. These were Class A (about 16 million addresses), Class B (about 65 thousand addresses), and Class C (255 addresses). While CIDR has replaced class-based allocation, these classes are often still referred to and used internally in organisations using private address space. See also ***CIDR***.

Class Based Queueing see ***CBQ***.

Classless Inter-Domain Routing see ***CIDR***.

collision. On an Ethernet network, a collision occurs when two devices connected to the same physical segment attempt to transmit at the same time. When collisions are detected, devices delay retransmission for a brief, randomly selected period.

colocation facility (***colo***). A service that provides hosting close to the Internet backbone. This may be provided as virtual space on a shared server, or as physical room for server equipment. ISPs often offer colo services to their customers.

connectionless. A network protocol (such as UDP) that requires no session initiation or maintenance. Connectionless protocols typically require less overhead than session oriented protocols, but do not usually offer data protection or packet reassembly. See also ***session oriented***.

connection tracking see ***stateful inspection***.

content filtering. Selectively allowing information to flow through a network based on the actual contents of the data. This may include virus scanners, spam filters, advertising blockers, or web proxy filters.

contention ratio. The ratio of customers to the available bandwidth. If an ISP provides a 1 megabit service and sells access to twenty customers, the contention ratio is 20:1.

cron job. A Unix facility that allows timed and repeated execution of programs.

curl (*http://curl.haxx.se/*). A command line tool for downloading web pages.

D

DansGuardian . A Squid redirector that provides web content filtering. Available at *http://dansguardian.org/*

Data Link Layer. The second layer in both the OSI and TCP/IP network models. Communications at this layer happen directly between nodes. On Ethernet networks, this is also sometimes called the MAC layer.

Data Over Cable Service Interface Specification see ***DOCSIS***.

default gateway. When a router receives a packet destined for a network for which it has no explicit route, the packet is forwarded to the default gateway. The default gateway then repeats the process, possibly sending the packet to its own default gateway, until the packet reaches its ultimate destination.

default route. A network route that points to the default gateway.

deflate. A compression algorithm used by PPP to reduce the size of packet headers. See also ***bsdcomp*** and ***VJ***.

delay pools. A packet shaping method used by Squid to prioritise data delivery.

Denial of Service see ***DoS***.

deny by default. A firewall policy that only allows traffic that is explicitly permitted. This is widely considered to be more secure than filtering only undesirable traffic.

DHCP (***Dynamic Host Configuration Protocol***). A protocol used by hosts to automatically determine their IP address.

dial on demand. A network connection that is only made when required. Dial on demand is often used with dial-up connections.

Digital Subscriber Line see ***DSL***.

Digital Video Broadcast see ***DVB-S***.

DNS Black List see ***DNSBL***.

DNS caching. By installing a DNS server on your local LAN, DNS requests for an entire network may be cached locally, improving response times. This technique is called DNS caching.

DNSBL (***DNS Black List***). A spam prevention technique that rejects inbound mail based on the originating IP address.

Dnsmasq. An open source caching DNS and DHCP server, available from *http://thekelleys.org.uk/*

DOCSIS (***Data Over Cable Service Interface Specification***). DOCSIS is the protocol spoken on cable modem networks that provides two-way data communications.

DomainKeys. A spam fighting technique developed by Yahoo! designed to verify the authenticity of the sender, as well as the integrity of the message.

DoS (***Denial of Service***). An attack on network resources, usually achieved by flooding a network with traffic or exploiting a bug in an application or network protocol. When the source of these attacks is distributed across a large number of machines, it is called a ***Distributed Denial of Service attack*** (***DDoS***).

download manager. A program that keeps track of downloaded files, often claiming to improve download speeds as well. Many download managers implement a peer-to-peer protocol to improve performance, which can cause significant impact on network utilisation. See also ***peer-to-peer***.

DROP. This netfilter target immediately discards the packet in question and stops any further processing. Compare with ***ACCEPT***, ***LOG***, and ***REJECT***.

DSL (***Digital Subscriber Line***). A family of related high speed network technologies implemented using standard telephone lines. The most common form is ADSL (Asymmetric Digital Subscriber Line), which provides faster download speeds than upload speeds. Another version is SDSL (Symmetric Digital Subscriber Line), which provides matching upload and download speeds, but usually at significantly greater cost. DSL can provide much greater capacity than dial-up, but has a limited installation range.

DSL modem. A device used to provide DSL service over traditional telephone lines.

DVB-S (***Digital Video Broadcast***). One of several standards used for satellite Internet access. See also ***BGAN*** and ***VSAT***.

Dynamic Host Configuration Protocol see ***DHCP***.

E

edge. The place where one organisation's network meets another. Edges are defined by the location of the external router, which often acts as a firewall.

equal cost routing. A technique used for aggregating network links in a round-robin fashion.

EtherApe. An open source network visualisation tool. Available at *http://etherape.sourceforge.net/*

Ethereal see ***Wireshark***.

EuroDOCSIS. The European version of the DOCSIS cable modem specification.

Exim (*http://www.exim.org/*). Exim is a popular email server (MTA) designed for flexibility and ease of administration.

external traffic. Network traffic that originates from, or is destined for, an IP address outside your internal network, such as Internet traffic.

F

far-side scrubbing. An optimisation technique where content filtering takes place at your ISP before it is sent across your Internet connection.

fibre optic see ***optical fibre***.

Fibre To The Home (***FTTH***) and ***Fibre To The Premises*** (***FTTP***). Very high speed Internet service provided via optical fibre. FTTH and FTTP are currently available in limited areas in only a few countries.

filter. The default table used in the Linux netfilter firewall system is the filter table. This table is used for determining traffic that should be accepted or denied.

firewall. A router that accepts or denies traffic based on some criteria. Firewalls are one basic tool used to protect entire networks from undesirable traffic. See also ***personal firewall***.

Flat rate billing. A billing method where a predetermined rate is charged for service, regardless of the amount of bandwidth used. Compare with ***95th percentile*** and ***actual usage***.

FLUFF. A distributed download system developed by the University of Bristol. More information is available at *http://www.bristol.ac.uk/fluff/*

flush. To remove all entries in a routing table or netfilter chain.

forwarding. When routers receive packets that are destined for a different host or network, they send the packet to the next router closest to its ultimate destination. This process is called forwarding.

forwarding loops. A routing misconfiguration where packets are forwarded cyclically between two or more routers. Catastrophic network failure is prevented by using the TTL value on every packet, but forwarding loops need to be resolved for proper network operations.

frame relay. A digital communications technology used for wide-area networks in cases where leased lines, DSL, or other wired network connections are impractical.

FTTH see ***Fibre To The Home***.

FTTP see ***Fibre To The Premises***.

G

Globally routable IP addresses. An address issued by an ISP or RIR that is reachable from any point on the Internet. In IPv4, there are approximately four billion possible IP addresses, although not all of these are globally routable.

greylists. A spam fighting technique where incoming emails are automatically deferred for a short amount of time. Greylists get their name from the combined use of whitelists and blacklists.

H

Hayes AT command set see ***AT command set***.

HF (***High-Frequency***). Radio waves from 3 to 30 MHz are referred to as

HF. Data networks can be built on HF that operate at very long range, but with very low data capacity.

Hierarchical Token Buckets see ***HTB***.

High-Frequency see ***HF***.

hop. Data that crosses one network connection. A web server may be several hops away from your local computer, as packets are forwarded from router to router, eventually reaching their ultimate destination.

HTB (***Hierarchical Token Buckets***). A class-based queuing discipline used for traffic shaping.

HTTrack (*http://www.httrack.com*). An open source offline browser utility used to make a local copy of websites.

hub. An Ethernet networking device that repeats received data on all connected ports. See also ***switch***.

I

IANA (***Internet Assigned Numbers Authority***). The organisation that administers various critical parts of Internet infrastructure, including IP address allocation, DNS root nameservers, and protocol service numbers.

ICMP (***Internet Control Message Protocol***). A Network Layer protocol used to inform nodes about the state of the network. ICMP is part of the Internet protocol suite. See also ***TCP/IP***.

ICP (***Internet Cache Protocol***). A high performance protocol used to communicate between web caches.

inbound traffic. Network packets that originate from outside the local network (typically the Internet) and are bound for a destination inside the local network. See also ***outbound traffic***.

infecting. The process where a network virus spreads from machine to machine. Viruses can sometimes be stopped by firewalls, and should be eliminated from your network using anti-virus software.

Integrated Services Digital Network see ***ISDN***.

interception caching see ***transparent caching***.

Internet Cache Protocol see ***ICP***.

Internet Control Message Protocol see ***ICMP***.

Internet Protocol see ***IP***.

Internet protocol suite. The family of communication protocols that make up the Internet. Some of these protocols include TCP, IP, ICMP, and UDP. Also called the ***TCP/IP protocol suite***, or simply ***TCP/IP***.

Intrusion Detection System (***IDS***). A program that watches network traffic, looking for suspicious data or behaviour patterns. An IDS may make a log entry, notify a network administrator, or take direct action in response to undesirable traffic.

IP (***Internet Protocol***). The most common network layer protocol in use. IP defines the hosts and networks that make up the global Internet.

IPF and ***IPFW***. Two of the three popular firewall implementations used in BSD. See also ***PF***.

iproute2. The advanced routing tools package for Linux, used for traffic shaping and other advanced techniques. Available from *http://linux-net.osdl.org/*

iptables. The primary command used to manipulate netfilter firewall rules.

ISDN (***Integrated Services Digital Network***). A network connection using digital signaling over the traditional telephone network.

ISM band. ISM is short for Industrial, Scientific, and Medical. The ISM band is a set of radio frequencies set aside by the ITU for unlicensed use.

Isoqlog. An open source MTA log processing and reporting tool. (*http://www.enderunix.org/isoqlog/*)

K

known good. In troubleshooting, a known good is any component that can be replaced to verify that its counterpart is in good, working condition.

L

l7-filter. An open source application layer firewall application. L7-filter can catch traffic based on the high level protocol being used, regardless of the source or destination port numbers used. This processing usually requires significant CPU resources. *http://l7-filter.sourceforge.net/*

LAN (***Local Area Network***). A network (typically Ethernet) used within an organisation. The part of a network that exists just behind an ISP's router is generally considered to be part of the LAN. See also ***WAN***.

Latency. The amount of time it takes for a packet to cross a network connection. It is often (somewhat incorrectly) used interchangeably with Round Trip Time (RTT), since measuring the RTT of a wide-area connection is trivial compared to measuring the actual latency. See also ***RTT***.

leaf node. When using HTB, a leaf node refers to a class that contains no child nodes. See also ***branch node***.

lease time. In DHCP, IP addresses are assigned for a limited period of time, known as the lease time. After this time period expires, clients must request a new IP address from the DHCP server.

leased line. A dedicated physical connection between two locations, usually leased from the local telephone company.

link-local. Network devices that are connected to the same physical segment communicate with each other directly, and are said to be link-local. A link-local connection cannot cross a router boundary without using some kind of encapsulation, such as tunneling or a VPN.

listen. Programs that accept connections on a TCP port are said to listen on that port.

Local Area Network see ***LAN***.

LOG. This netfilter target writes the packet to the system log and continues processing rules. See also ***ACCEPT***, ***DROP***, and ***REJECT***.

Log analysis. Computer logs can be read by humans, but are often more useful when processed by a log analyser. These tools can distill a large number of events into aggregated reports and trends, and can notify a human immediately when emergency conditions occur.

long fat pipe. A network connection (such as VSAT) that has high capacity and high latency. In order to achieve the best possible performance, TCP/IP must be tuned to traffic on such links.

M

MAC (***Media Access Control***) ***layer***. See ***Data Link Layer***.

MAC table. A network switch must keep track of the MAC addresses used on each physical port, in order to efficiently distribute packets. This information is kept in a table called the MAC table.

Mail Delivery Agent see ***MDA***.

Mail Transfer Agent see ***MTA***.

Mail User Agent see ***MUA***.

malware. Software such as a virus or keylogger that performs undesirable actions on a computer, often without the user's knowledge. See also ***trojan horse***, ***virus***, and ***worm***.

managed. Networking hardware that provides an administrative interface, port counters, SNMP, or other interactive features is said to be managed.

masquerading. A form of Network Address Translation used by Linux routers.

master browser. On Windows networks, the master browser is the computer that keeps a list of all the computers, shares and printers that are available in **Network Neighborhood** or **My Network Places**.

match condition. In netfilter, a match condition specifies the criteria that determine the ultimate target for a given packet. Packets may be matched on MAC address, source or destination IP address, port number, data contents, or just about any other property.

MDA (***Mail Delivery Agent***). A program that delivers an email to a storage device (usually a hard disk on an email server). This may be implemented in the MTA itself, or using an external program such as procmail. See also ***MTA*** and ***MUA***.

Media Access Control see ***MAC***

message types. Rather that port numbers, ICMP traffic uses message types to define the type of information being sent. See also ***ICMP***.

Microsoft Windows Server Update Services see ***WSUS***.

milter. An email filter specification supported by Sendmail and Postfix.

Mirroring. Making a complete local copy of a web site or other online resource. Bandwidth can be saved by directing users to the local copy, rather than allowing them to access the original site directly.

modem. Short for modulator / demodulator, the term modem was once used to refer to any interface between a computer and an analog network connection (such as a telephone line). In recent years, it has come to represent any device that bridges a network to Ethernet.

monitor port. On a managed switch, one or more monitor ports may be defined that receive traffic sent to all of the other ports. This allows you to connect a traffic monitor server to the port to observe and analyse traffic patterns.

MRTG (***Multi Router Traffic Grapher***). An open source tool used for graphing traffic statistics. Available from *http://oss.oetiker.ch/mrtg/*

MTA (***Mail Transfer Agent***). A program that transports email between networks. Servers run an MTA such as Sendmail or Postfix to accept mail for a domain. See also ***MDA*** and ***MUA***.

mtr (***My TraceRoute***). A network diagnostic tool used as an alternative to the traditional traceroute program. *http://www.bitwizard.nl/mtr/.* See also ***traceroute*** / ***tracert***.

MUA (***Mail User Agent***). A program that retrieves and displays email message. Thunderbird and Outlook are examples of MUAs. See also ***MDA*** and ***MTA***.

Multi Router Traffic Grapher see ***MRTG***.

My TraceRoute see ***mtr***.

N

Nagios (*http://nagios.org/*) A real-time monitoring tool that logs and notifies a system administrator about service and network outages.

NAT (***Network Address Translation***). NAT is a networking technology that allows many computers to share a single, globally routable IP address. While NAT can help to solve the problem of limited IP address space, it creates a technical challenge for two-way services, such as Voice over IP.

nat. The table used in the Linux netfilter firewall system to configure Network Address Translation.

negative-cached. In addition to normal responses, network failures can also be cached for increased performance. Squid will cache failures (such as **404 Not Found** responses) to client requests to prevent redundant retrieval of nonexistent pages. Caching DNS servers will also cache negative responses (replies to nonexistent host names) for the amount of time defined in the domain's zone file.

NetBIOS. A session layer protocol used by Windows networking for file and printer sharing. See also ***SMB***.

netfilter. The packet filtering framework in modern Linux kernels is known as netfilter. It uses the iptables command to manipulate filter rules. *http://netfilter.org/*

netmask (***network mask***). A netmask is a 32-bit number that divides the 16 million available IP addresses into smaller chunks, called subnets. All IP networks use IP addresses in combination with netmasks to logically group hosts and networks.

NeTraMet. An open source network flow analysis tool available from *http://www.auckland.ac.nz/net/*

network address. The lowest IP number in a subnet. The network address is used in routing tables to specify the destination to be used when sending packets to a logical group of IP addresses.

Network Address Translation see ***NAT***.

network etiquette. Generally accepted guidelines of behaviour that are considered to be polite to other network users. Conserving bandwidth, making sure your computer is virus-free, and refraining from sending spam email are considered good network etiquette.

Network Layer. The third layer of the OSI and TCP/IP network models, where IP operates and Internet routing takes place.

network mask see ***netmask***.

ngrep. An open source network security utility used to find patterns in data flows. Available from *http://ngrep.sourceforge.net/*

nmap. An open source network security utility used to scan networks and hosts to probe available services. *http://insecure.org/nmap/*

Norton Personal Firewall. A commercial personal firewall program published by Symantec. See also ***firewall***, ***personal firewall***, and ***Zone Alarm***.

ntop. A network monitoring tool that provides extensive detail about connections and protocol use on a local area network. *http://www.ntop.org/*

O

open relay. An email server that accepts mail delivery to any domain when sent from any location is called an open relay. Spammers make extensive use of open relays to hide the origin of their traffic. See also ***spam***.

Optical fibre. Communication cables made from glass that provide very high bandwidth and very low latency across very long distances. See also ***FTTH***, ***FTTP***, and ***SDH***.

OR logic. A logical operation that evaluates as true if any of the items being compared also evaluate as true. See also ***AND logic***.

OSI network model. A popular model of network communications defined by the ISO/IEC 7498-1 standard. The OSI model consists of seven interdependent layers, from the physical through the application. See also ***TCP/IP network model***.

outbound traffic. Network packets that originate from the local network and are bound for a destination outside the local network (typically somewhere on the Internet). See also ***inbound traffic***.

P

pac see ***Proxy Auto Configuration (.pac) file***

Packet filter. A firewall that operates at the Internet layer by inspecting source and destination IP addresses, port numbers, and protocols. Packets are either permitted or discarded depending on the packet filter rules.

packets. On IP networks, messages sent between computers are broken into small pieces called packets. Each packet includes a source, destination, and other routing information that is used to route it to its ultimate destination. Packets are reassembled again at the remote end by TCP (or another protocol) before being passed to the application.

partition. A technique used by network hubs to limit the impact of computers that transmit excessively. Hubs will temporarily remove the abusive computer (partition it) from the rest of the network, and reconnect it again after some time. Excessive partitioning indicates the presence of an excessive bandwidth consumer, such as a peer-to-peer client or network virus.

peer-to-peer. Any of several popular programs (such as BitTorrent, Gnutella, KaZaA, or eDonkey2000) used for file sharing. A peer-to-peer program turns a user's computer into both a client and a server, where information is exchanged directly with everyone else who is also running the program. Peer-to-peer programs consume considerable bandwidth, both inbound and outbound. See also ***download manager***.

PEPsal. An open source performance enhancing proxy used for improving TCP performance on links with different characteristics (such as VSAT and VPNs). Available from *http://sourceforge.net/projects/pepsal/*

personal firewall. An application used on client computers to provide a small measure of protection from network attacks.

PF. One of three firewall implementations used in BSD (along with IPF and IPFW). See also ***IPF*** and ***IPFW***.

pfifo_fast. The default queuing discipline used on Linux network interfaces. It defines three bands of priority that are used according to the Type Of Service (TOS) bits present in a given packet. See also ***qdisc***, ***QoS***, and ***TOS***.

Physical Layer. The lowest layer in both the OSI and TCP/IP network models. The physical layer is the actual medium used for communications, such as copper cable, optic fibre, or radio waves.

ping. A ubiquitous network diagnostic utility that uses ICMP echo request and reply messages to determine the round trip time to a network host. Ping can be used to determine the location of network problems by "pinging" computers in the path between the local machine and the ultimate destination.

Point-to-Point Protocol see ***PPP***.

policy. In netfilter, the policy is the default action to be taken when no other filtering rules apply. For example, the default policy for any chain may be set to ACCEPT or DROP.

port counters. Managed switches and routers provide statistics for each network port called port counters. These statistics may include inbound and outbound packet and byte counts, as well as errors and retransmissions.

Postfix (***http://www.postfix.org/***). A popular email server (MTA) designed as a more secure alternative to Sendmail.

PPP (***Point-to-Point Protocol***). A network protocol typically used on serial lines (such as a dial-up connection) to provide IP connectivity.

Presentation Layer. The sixth layer of the OSI networking model. This layer deals with data representation, such as MIME encoding or data compression.

PRIO. A queuing discipline used with Linux QoS to prioritise traffic according to Type Of Service (TOS) bits present in the packet. See also ***qdisc***, ***QoS***, and ***TOS***.

PRIQ (***Priority Queueing***). A queuing discipline used to implement QoS on BSD systems. See also ***CBQ***, ***qdisc***, and ***QoS***.

private address space. A set of reserved IP addresses outlined in RFC1918. Private address space is frequently used within an organisation, in conjunction with Network Address Translation (NAT). The reserved private address space ranges include 10.0.0.0/8, 172.16.0.0/12, and 192.168.0.0/16. See also ***NAT***.

protocol analyser. A diagnostic program used to observe and disassemble network packets. Protocol analysers provide the greatest possible detail about individual packets.

protocol stack. A set of network protocols that provide interdependent layers of functionality. See also ***OSI network model*** and ***TCP/IP network model***.

Proxy Auto Configuration (.pac) file. A file used in conjunction with a web server to automatically provide proxy information to a web browser.

proxy server. A network server that makes requests on behalf of client computers. Requests may or may not be cached locally, and the server may require authentication credentials. The Squid web cache is one example of a proxy server.

public good. A resource that can be consumed by an individual in arbitrarily large amounts, irrespective of the contribution made by that individual to conserving or renewing that resource.

Q

qdisc (***queuing discipline***). An algorithm that controls when and how the interface is allowed to send packets. See also ***ALTQ***, ***CBQ***, ***HTB***, ***pfifo_fast***, ***PRIO***, and ***PRIQ***.

qmail (*http://www.qmail.org/*). A popular email server (MTA) designed for security and speed.

QoS (***Quality of Service***). QoS allows you to prioritise the delivery of traffic based on some criteria, such as the type of service or originating network. Since packets are already sent as quickly as possible, QoS techniques only help when a communications line approaches saturation. See also ***TOS***.

queuing discipline see ***qdisc***.

R

Really Simple Syndication see ***RSS***.

Realtime monitoring. A network monitoring tool that performs unattended monitoring over long periods, and notifies administrators immediately when problems arise.

Redirector. A feature of the Squid web proxy that allows an administrator to intercept a user's browser and redirect them to different web content. This feature is used to implement captive portals, bandwidth enforcement pages, advertisement blocking, etc.

regex (***Regular Expression***). A pattern matching language used to determine if a given string matches a particular pattern. Many programs use a form of regular expressions to match various kinds of input. For example, Squid can use regex matches to determine if a requested URL fits a particular pattern (such as your organisation's domain name).

Regional Internet Registrar see ***RIR***.

Regular Expression see ***regex***.

REJECT. The netfilter target that returns an ICMP error message on matched packets. This is considered more polite, but less secure, than using the DROP target. Packet matching stops immediately once the REJECT target is met. Compare with ***ACCEPT***, ***DROP***, and ***LOG***.

RFC (***Request For Comments***). RFCs are a numbered series of documents published by the Internet Society that document ideas and concepts related to Internet technologies. Not all RFCs are actual standards, but many are either approved explicitly by the IETF, or eventually become de facto standards. RFCs can be viewed online at *http://rfc.net/.*

RIR (***Regional Internet Registrar***). The 4 billion available IP addresses are administered by the IANA. The space has been divided into large subnets, which are delegated to one of the five regional Internet registries, each with authority over a large geographic area.

robot exclusion standard (***robots.txt***). A convention used to limit the impact of automatic web crawlers (spiders) on a web server. Well-behaved web page retrieval software will only visit pages permitted by the robots.txt file. This can significantly reduce the load on your web server and Internet connection.

Round Robin Database see ***RRD***.

Round Trip Time see ***RTT***.

router. A device that forwards packets between different networks. The process of forwarding packets to the next hop is called ***routing.***

routing table. A list of networks and IP addresses kept by a router to determine how packets should be forwarded. If a router receives a packet for a network that is not in the routing table, the router uses its default gateway. Routers operate at the Network Layer. See also ***bridge*** and ***default gateway***.

RRD (***Round Robin Database***). A database that stores information in a very compact way that does not expand over time. This is the data format used by RRDtool and other network monitoring tools.

RRDtool. A suite of tools that allow you to create and modify RRD databases, as well as generate useful graphs to present the data. RRDtool is used to keep track of time-series data (such as network bandwidth, machine room temperature, or server load average) and can display that data as an average over time. RRDtool is available from *http://oss.oetiker.ch/rrdtool/*

RSS (***Really Simple Syndication***). A format used for providing news feeds. Anything that can be broken down into discrete items (such as news stories, wiki posts, or blog entries) can be syndicated with RSS. Rather than using a web browser, users collect RSS feeds using an RSS browser.

rsync (*http://rsync.samba.org/*). An open source incremental file transfer utility used for maintaining mirrors.

RTT (***round trip time***). The amount of time it takes for a packet to be acknowledged from the remote end of a connection. Frequently confused with ***latency***.

rules. Entries used in netfilter chains to match, manipulate, and determine the ultimate fate of a packet. See also ***chains***, ***netfilter***, and ***tables***.

S

SACK (***Selective acknowledgment***). A mechanism used to overcome TCP inefficiencies on high latency networks, such as VSAT.

Sawmill. A commercial log processing and reporting tool. Available from *http://www.sawmill.net/*

SDH (***Synchronous Digital Hierarchy***). A popular Data Link Layer protocol used on fibre optic networks.

SDSL (***Symmetric Digital Subscriber Line***) see ***DSL***.

Selective acknowledgment see ***SACK***.

Sender Policy Framework see ***SPF***.

Sendmail. The oldest open source email server still in wide use. Available from *http://www.sendmail.org/*

Server Message Block see ***SMB***.

Service Level Agreement see ***SLA***.

Session Layer. Layer five of the OSI model, the Session Layer manages logical connections between applications.

session oriented. A network protocol (such as TCP) that requires initialisation before data can be exchanged, as well as some clean-up after data exchange has completed. Session oriented protocols typically offer error correction and packet reassembly, while connectionless protocols do not. See also ***connectionless***.

SFQ (***Stochastic Fairness Queuing***). A fair queueing algorithm designed to require fewer calculations than other algorithms while being almost perfectly fair. Rather than allocate a separate queue for each session, it uses an algorithm that divides traffic over a limited number of queues using a hashing algorithm. This assignment is nearly random, hence the name "stochastic." See also ***ALTQ***, ***CBQ***, and ***HTB***.

Shorewall (*http://shorewall.net/*). A configuration tool used for setting up netfilter firewalls without the need to learn iptables syntax.

Simple Mail Transfer Protocol see ***SMTP***.

Simple Network Management Protocol see ***SNMP***.

site-wide web cache. While all modern web browsers provide a local data cache, large organisations can improve efficiency by installing a site-wide web cache, such as Squid. A site-wide web cache keeps a copy of all requests made from within an organisation, and serves the local copy on subsequent requests. See also ***Squid***.

SLA (***Service Level Agreement***). A document that describes the precise level of network service that will be provided, including technical support, minimum uptime statistics, emergency contact procedures, and liability for unforeseen service outages. ISPs typically provide SLAs to customers at different rates depending on the level of service requested.

SMB (***Server Message Block***). A network protocol used in Windows networks to provide file sharing services. See also ***NetBIOS***.

SmokePing. A latency measurement tool that measures, stores and displays latency, latency distribution and packet loss all on a single graph. SmokePing is available from *http://oss.oetiker.ch/smokeping/*

SMTP (***Simple Mail Transfer Protocol***). The protocol used to exchange email between MTAs.

SNMP (***Simple Network Management Protocol***). A protocol designed to facilitate the exchange of management information between network devices. SNMP is typically used to poll network switches and routers to gather operating statistics.

Snort (*http://www.snort.org/*). A very popular open source intrusion detection system. See also ***IDS***.

SOCKS proxy. A generic application proxy server used for improving site security. There are many free and commercial SOCKS servers and clients available. Most web browsers have support for SOCKS proxies. See RFC1928.

spam. Unsolicited and undesirable communications, usually in the form of email messages, news group postings, or blog comments. Spam messages often include advertising or attempt to involve the recipient some kind of fraudulent activity. Spam wastes bandwidth, causes frustration in users, and the sending of it has been made illegal in many parts of the world.

spammers. People who engage in sending spam in an effort to attack, annoy, enrage, exploit, extort, swindle, or steal. Keep them out of your networks.

Speed. A generic term used to refer to the responsiveness of a network connection. A "high-speed" network should have low latency and more than enough capacity to carry the traffic of its users. See also ***bandwidth***, ***capacity***, and ***latency***.

SPF (***Sender Policy Framework***). A technique used to fight email forgery, which is often used in scam emails. SPF allows you to verify that mail apparently from a particular domain (say, a financial institution or government office) was sent from an authorised mail server. SPF verifies the path that email took to arrive at your MTA, and can discard email that originated at unauthorised MTAs before the message is transmitted, thus saving bandwidth and reducing spam.

split horizon DNS. A technique used to serve different answers to DNS requests based on the source of the request. Split horizon is used to direct internal users to a different set of servers than Internet users.

Spot check tools. Network monitoring tools that are run only when needed to diagnose a problem. Ping and traceroute are examples of spot check tools.

Squid. A very popular open source web proxy cache. It is flexible, robust, full-featured, and scales to support networks of nearly any size. *http://www.squid-cache.org/*

Squidguard. A Squid redirector that provides web content filtering. *http://www.squidguard.org/*

stateful inspection. Firewall rules that are aware of the the state associated with a given packet. The state is not part of the packet as transmitted over the Internet, but is determined by the firewall itself. New, established, and related connections may all be taken into consideration when filtering packets. Stateful inspection is sometimes called connection tracking.

Stochastic Fairness Queueing see ***SFQ***.

subnet. An IP network that is broken down into smaller groups of hosts through the use of netmasks.

subnet mask see ***netmask***.

swarming. Another name for peer-to-peer activity. See ***peer-to-peer***.

switch. A network device that provides a temporary, dedicated connection between communicating devices. See also ***hub***.

Symmetric Digital Subscriber Line see ***DSL***.

Synchronous Digital Hierarchy see ***SDH***.

T

tables. Groups of netfilter chains that define the type of operation to be done (such as filter or nat). See also ***chains, netfilter***, and ***rules***.

target. In netfilter, the action to be taken once a packet matches a rule. Some possible netfilter targets include ***ACCEPT***, ***DROP***, ***LOG***, and ***REJECT***.

TBF (***Token Bucket Filter***). An algorithm used to throttle traffic to a particular rate. It is the algorithm used by delay pools in Squid, and can be used as a queuing discipline. See also ***ALTQ***, ***CBQ***, ***HTB***, ***pfifo_fast***, ***PRIO***, and ***PRIQ***.

TCP (***Transmission Control Protocol***). A session oriented protocol that operates at the Transport Layer, providing packet reassembly, congestion avoidance, and reliable delivery. TCP is an integral protocol used by many Internet applications, including HTTP and SMTP. See also ***UDP***.

TCP window size. The TCP parameter that defines how much data that may be sent before an ACK packet is returned from the receiving side. For instance, a window size of 3000 would mean that two packets of 1500 bytes each will be sent, after which the receiving end will either ACK the chunk or request retransmission.

TCP/IP. See ***Internet protocol suite***.

TCP/IP network model. A popular simplification of the OSI network model that is used with Internet networks. The TCP/IP model consists of five interdependent layers, from the physical through the application. See also ***OSI network model***.

tcpdump. A popular open source packet capture and analysis tool available at *http://www.tcpdump.org/.* See also ***WinDump*** and ***Wireshark***.

thrashing. Excessive hard disk use that occurs when a system has insufficient RAM, and must continually swap processes out to disk.

Throughput. The actual amount of information flowing through a network connection, disregarding protocol overhead.

Time To Live see ***TTL***.

Token Bucket Filter see ***TBF***.

TOS (***Type Of Service***). TOS bits may be assigned to a packet to determine QoS characteristics. The TOS bits determine whether a packet should be prioritised to minimise delay, maximise throughput, maximise reliability, minimise monetary cost, or some combination of these. Applications request the appropriate TOS bits when transmitting packets. See also ***QoS***.

traceroute / tracert. A ubiquitous network diagnostic utility often used in conjunction with ping to determine the location of network problems. The Unix version is called traceroute, while the Windows version is tracert. Both use ICMP echo requests with increasing TTL values to determine which routers are used to connect to a remote host, and also display latency statistics. Another variant is tracepath, which uses a similar technique with UDP packets. See also ***mtr***.

Transmission Control Protocol see ***TCP***.

transparent bridging firewall. A firewall technique that introduces a bridge that selectively forwards packets based on firewall rules. One benefit of a transparent bridging firewall is that it does not require an IP address. See also ***bridge***.

transparent cache. A method of implementing a site-wide web cache that requires no configuration on the web clients. Web requests are silently redirected to the cache, which makes the request on behalf of the client. Transparent caches cannot use authentication, which makes it impossible to implement traffic accounting at the user level. See also ***site-wide web cache***, ***Squid***.

Transport Layer. The third layer of the OSI and TCP/IP network models, which provides a method of reaching a particular service on a given network node. Examples of protocols that operate at this layer are ***TCP*** and ***UDP***.

Trending. A type of network monitoring tool that performs unattended monitoring over long periods, and plots the results on a graph. Trending tools allow you to predict future behaviour of your network, which helps you plan for upgrades and changes.

trojan horse. A type of malware that claims to perform some useful function while secretly performing some other task (such as sending spam email or infecting a system with a virus). See also ***malware***, ***virus***, and ***worm***.

trunking. A feature of network switches that support VLAN tagging. A trunked connection can carry traffic from multiple VLANs on a single physical cable, and extend the reach of a VLAN to other network devices. See also ***VLAN***.

TTL (***Time To Live***). A TTL value acts as a deadline or emergency brake to signal a time when the data should be discarded. In TCP/IP networks, the TTL is a counter that starts at some value (such as 64) and is decremented at each router hop. If the TTL reaches zero, the packet is discarded. This mechanism helps reduce damage caused by routing loops. In DNS, the TTL defines the amount of time that a particular zone record should be kept before it must be refreshed. In Squid, the TTL defines how long a cached object may be kept before it must be again retrieved from the original website.

tunnel. A form of data encapsulation that wraps one protocol stack within another. This is often used in conjunction with encryption to protect communications from potential eavesdroppers, while eliminating the need to support encryption within the application itself. Tunnels are often used conjunction with VPNs. See also ***VPN***.

Type Of Service see ***TOS***.

U

UBE (***Unsolicited Bulk Email***). Another term for ***spam***.

UDP (***User Datagram Protocol***). A connectionless protocol that operates at the Transport Layer. UDP does not provide error correction or packet reassembly, but it requires less overhead than TCP connections. UDP is an integral protocol used by many Internet applications, including DNS, VoIP, streaming media, and gaming. See also ***TCP***.

Unsolicited Bulk Email see ***UBE***.

User Datagram Protocol see ***UDP***.

V

Van Jacobson see ***VJ***.

vector. A place where malware enters a computer or network. Vectors are security holes that should be fixed whenever they are found.

versionitis. The chaos that often ensues when multiple people attempt to make changes to the same document. Unless the changes are managed intelligently, the result may be several different copies of the same document, each with incompatible changes.

Very Small Aperture Terminal see ***VSAT***.

Virtual LAN see ***VLAN***.

Virtual Private Network see ***VPN***.

virus. A type of malware that exploits security holes to cause a computer to perform an undesirable task (such as sending spam email, deleting files, or infecting other systems). See also ***malware***, ***trojan horse***, and ***worm***.

VJ (***Van Jacobson***). A type of PPP header compression defined in RFC1144. See also ***bsdcomp*** and ***deflate***.

VLAN (***Virtual LAN***). A logical network that can coexist with, and be isolated from, other VLANs on the same physical medium. VLANs are normally implemented by network switching hardware, and so it makes no difference to a computer whether it is connected to a LAN or a VLAN. See also ***trunking***.

VoIP (***Voice over IP***). A technology that provides telephone-like features over an Internet connection. Examples of popular VoIP clients include Skype, Gizmo Project, MSN Messenger, and iChat.

VPN (***Virtual Private Network***). A tool used to join two networks together over an untrusted network (such as the Internet). VPNs are often used to connect remote users to an organisation's network when travelling or working from home. VPNs use a combination of encryption and tunneling to secure all network traffic, regardless of the application being used. See also ***tunnel***.

VSAT (***Very Small Aperture Terminal***). One of several standards used for satellite Internet access. VSAT is the most widely deployed satellite technology used in Africa. See also ***BGAN*** and ***DVB-S***.

W

WAN (***Wide Area Network***). Any long distance networking technology. Leased lines, frame relay, DSL, fixed wireless, and satellite all typically implement wide area networks. See also ***LAN***.

web application firewall. A firewall that understands HTTP data, and can make packet delivery decisions based on that data. For example, a web application firewall may refuse to allow requests that post spam or virus data to a public forum.

Web Proxy Auto Discovery see ***WPAD***.

Webalizer. A popular open source web and cache server log reporting tool. *http://www.mrunix.net/webalizer/*

webmail. An MUA implemented as a web application. Hotmail, Gmail, and Yahoo! mail are examples of webmail applications. Webmail uses considerably more bandwidth than traditional email services.

wget. An open source command line tool for downloading web pages. *http://www.gnu.org/software/wget/*

Wi-Fi. A marketing brand owned by the Wi-Fi Alliance that is used to refer to various wireless networking technologies (including 802.11a, 802.11b, and 802.11g).

Wide Area Network see ***WAN***.

wiki. A web site that allows any user to edit the contents of any page. One of the most popular public wikis is *http://www.wikipedia.org/*

window scale. A TCP enhancement defined by RFC1323 that allows TCP window sizes larger than 64KB.

WinDump. The Windows version of tcpdump. It is available from *http://www.winpcap.org/windump/*

Wireless Fidelity see ***Wi-Fi***.

wireshark. A free network protocol analyser for Unix and Windows. *http://www.wireshark.org/*

worm. A type of malware similar to a virus that attempts to spread copies of itself to as many hosts as possible. Worms differ from viruses in that they do not contain an intentionally malicious payload, but their presence can consume bandwidth and cause other problems. See also ***malware***, ***trojan horse***, and ***virus***.

WPAD (***Web Proxy Auto Discovery***). A protocol that provides a number of methods for generating a URL that refers to a Proxy Auto Configuration file. See also ***Proxy Auto Configuration (.pac) file***.

WSUS (***Microsoft Windows Server Update Services***). A server used to replace the standard Microsoft update site. By directing clients to a local WSUS mirror, tremendous amounts of bandwidth can be saved as the clients automatically update their Windows components.

Z

ZoneAlarm. A commercial personal firewall program available from *http://www.zonelabs.com/*. See also ***Norton Personal Firewall***, ***personal firewall***.

Colophon

Lead copy editor: **Lisa Chan**, *http://www.cowinanorange.com/*
Supporting copy edit: **Kimberly Thomas**, kimthomas73@earthlink.net
Illustrations and cover: **Jessie Heaven Lotz**, *http://jessieheavenlotz.com/*
Technical review: **Roger Weeks** (*http://www.oreillynet.com/pub/au/1280*)
and **Richard Lotz** (*rlotz@seattlewireless.net*)

Produced by Hacker Friendly LLC, Seattle, WA, USA.
http://hackerfriendly.com/